

Safe and Efficient Robot Action Choice Using Human Intent Prediction in Physically-
Shared Space Environments

by

Catharine L. R. McGhan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2014

Doctoral Committee:

Associate Professor Ella M. Atkins, Chair
Assistant Professor James W. Cutler
Professor Ilya V. Kolmanovsky
Assistant Professor Emily Mower Provost

© Catharine L. R. McGhan 2014

For my mother, who explained what an engineer is.

For my brother, my first and closest friend.

For my father, frustrating, but still loving.

For those who encouraged me, when I doubted myself.

For everyone else, who might find this of use.

And for myself, because I can.

Acknowledgments

I'd like to thank my advisor, Ella Atkins, who has been both my advisor and my mentor since way back when I first started my college career at the University of Maryland in 2000. She had been exceedingly kind, patient, and helpful over these many years. For being willing to share her knowledge and experience with me, and for giving me a chance in the first place, I will be forever grateful. I would also like to thank the undergraduate research students who were a part of the Undergraduate Research Opportunities Program and helped develop our safe robot manipulator platform – Jeremy Green, Kevin Matzen, and Ryan Wolcott – and Gabriel Arroyo, who helped conduct human subject experiments the following summer. I'd also like to thank both the graduate and undergraduate student volunteers from the Department of Aerospace Engineering at the University of Michigan who participated in the human subject testing that gave us the reference data for our Chapter 3 findings. I would also like to thank Ali Nasir, who helped me to understand Markov Decision Processes and their use, and who was willing to bounce ideas back and forth with me, helping me create the original MDP software for policy calculation, and also in the determination of the MDP HIP formulation and representation. Ali, your help has been invaluable! Thanks also to Justin Bradley, Derrick Yeo, Ryan Eubank, and Aaron Hoskins for being good friends and colleagues over the years – you've really helped me keep my spirits up, to keep the 'imposter syndrome' at bay, and to focus on the main prize. Finally, a big thank-you goes out to my mother, Judy, who went above and beyond the call of duty by proofreading this manuscript from start to finish. She didn't have to do it, but she did it anyway, for me. So believe me when I say: anything that somehow escaped both her and my advisor that might possibly have *dared* to remain in error is all on me!

Table of Contents

Dedication	ii
Acknowledgments	iii
List of Figures	viii
List of Tables.....	xiii
List of Appendices.....	xv
List of Symbols	xvi
Abstract.....	xxiii
Chapter 1 Introduction	1
1.1 Motivation.....	2
1.2 Problem Statement	9
1.3 Research Objectives	13
1.4 Approach.....	14
1.5 Contributions.....	15
1.6 Innovations.....	16
1.7 Outline	16
Chapter 2 Background	18
2.1 Introduction.....	18
2.2 Robotic Manipulation.....	19
2.2.1 Kinematics and dynamics	19
2.2.2 Manipulator Trajectory Generation.....	24
2.3 Task Planning & Scheduling.....	26

2.3.1	Deterministic planning	26
2.3.2	Probabilistic planning	28
2.4	Space Robotics and Manipulation	31
2.5	Human-Robot Interaction (HRI)	33
2.5.1	Human Modeling.....	38
2.5.2	Robot Decision-Making with Integrated Human Models	39
2.5.3	Human Subject Experiments.....	40
Chapter 3 Experiments on Human-Robot Operation in a Shared Workspace		42
3.1	Introduction.....	42
3.2	Test Environment	43
3.2.1	Test conductor interface	45
3.2.2	Test subject interface	46
3.3	MM-Arm Hardware and Control	47
3.4	Human Subject Experiments.....	52
3.4.1	Test Methodology	52
3.4.2	Assumptions and Constraints.....	55
3.4.3	Test Matrix.....	55
3.5	Test Metrics.....	60
3.6	Results.....	62
3.6.1	Learning Curve.....	63
3.6.2	Paired Complementary Test Comparisons – Robot-as-Subordinate.....	67
3.6.3	Task Category Comparisons	69
3.6.4	Task Completion Times.....	73
3.6.5	Subject Reaction.....	75
3.7	Preliminary Conclusions.....	76

Chapter 4 System Architecture with Feedback for Human-Robot Interaction	79
4.1 Introduction	79
4.2 Motivating Example	80
4.3 System Architecture	81
Chapter 5 Human Intent Prediction	90
5.1 Introduction	90
5.2 Markov Decision Process (MDP) Formulation for Human Intent Prediction	91
5.2.1 States and Actions	92
5.2.2 Transition Probability Function	95
5.2.3 Rewards	100
5.3 Metrics for Performance Evaluation	102
5.4 Case Studies	103
5.4.1 Encoding Pre-existing Script(s) within a Markov Decision Process	104
5.4.2 Case Study #1 – EVA space repair example, deterministic system.....	108
5.4.3 Stochastic HIP modeling	119
5.4.4 Case Study #2 – IVA scenario, stochastic system	119
5.4.5 Inclusion of action-recognition input $a_{n_h+1}^i$ for one-step predictive lookahead	158
5.5 Conclusions and Discussion	159
5.5.1 Future work: evaluation and comparison against other methods.....	160
5.5.2 Future work: handling of model uncertainty	161
5.5.3 Future work: simulated human vs. human matching models	162
5.5.4 Future work: computation of action history length	163
Chapter 6 Robot Planning for Optimal Human-Robot Interaction	164
6.1 Introduction	164

6.2	Markov Decision Process (MDP) for Robot Action Choice (RAC)	165
6.2.1	States and Actions	166
6.2.2	Transition Probabilities.....	171
6.2.3	Rewards	178
6.3	Metrics for RAC MDP Performance Evaluation	183
6.4	Case Studies	184
6.4.1	Encoding Zone Information within an RAC MDP state space.....	184
6.4.2	Case Study #1 – IVA scenario, with and without human state input.....	194
6.5	Conclusions and Discussion	214
6.5.1	Feedback of RAC into HIP	215
6.5.2	Comparison of primarily-scripted HIP+RAC to A*, POMDP, or other methods	216
6.5.3	Markov chains for progression of robot action choice.....	218
6.5.4	Differing choice of R_2 algorithm.....	218
6.5.5	Impact of allowing reactive controller to handle conflict resolution ‘intelligently’	219
6.5.6	Similar state spaces, same or different transition probability and reward functions	220
6.5.7	Explicit zone calculations and mappings.....	220
6.5.8	Relaxation of assumption of perfect HIP information	220
6.5.9	Relaxation of fixed-base assumption	221
Chapter 7	Conclusions and Future Research Directions	222
7.1	Summary and Conclusions	222
7.2	Future Work.....	223
Appendices	225
Bibliography	245

List of Figures

Figure 1-1: Astronaut working in the International Space Station’s Kibo laboratory	3
Figure 1-2: EVA spacewalk finishing repairs on a torn solar array	3
Figure 1-3: Summary Timeline for Two Astronauts on EVA1, Flight STS-135 [3]	4
Figure 1-4: Partial Timeline for “Install COLTS” and “SSRMS Setup” Tasks on EVA1, Flight STS-135 [3].....	5
Figure 2-1: Markov Chain Model	28
Figure 2-2: Hidden Markov Model	28
Figure 2-3: MDP Representation	30
Figure 2-4: POMDP Representation.....	30
Figure 3-1: Hardware Subsystems for Human-Robot Experiments.....	43
Figure 3-2: Software Infrastructure	45
Figure 3-3: Test Conductor Interface Keyboard Bindings (number keys = button-pushing actions, yellow keys = drink soda, pink keys = eat chip)	46
Figure 3-4: Sample Math Problem Display (with Blue Waypoint Target in Foreground)	47
Figure 3-5: Workspace Setup with MM-arm; buttons b1, b2, and b3 are indicated to the Test Subject by Blue Reflectors	48
Figure 3-6: Test Scenarios	58
Figure 3-7: Selected TLX Load Source Ratings Relative to Baseline: Subject 5, Test Set 1	64
Figure 3-8: TLX Load Source Ratings for Baseline Cases Over All Test Subjects	65
Figure 3-9: Correctness Rate for Math Problems: Across All Subjects, Test Set 1	66
Figure 3-10: Selected TLX Load Source Ratings Relative to Baseline by Task Type, Subject	70

Figure 3-11: Comparing Correctness Rates between Test Groupings Across All Subjects	71
Figure 3-12: Selected TLX Load Source Ratings Relative to Baseline Across All Subjects and Tests by Task Type (no overtasking cases)	72
Figure 3-13: Task Completion Times, Across-All-Tests per Subject	74
Figure 4-1: General 3T Architecture for Space HRI with Feedback, System-Level	82
Figure 4-2: 3T Architecture with Decomposed Human Intent Prediction (HIP) and Robot Action Choice (RAC)	84
Figure 4-3: Timing of Intent Updates as Used by Robot Action Choice (RAC)	88
Figure 5-1: General transition cases for HIP, with no in-progress action supplied	97
Figure 5-2: State evolution for the optimal MDP policy, case 4c (7 goals, $n_h=0$), starting from $s^i = \{\text{no goals set}\}$, for $s^i = \{g_1^i, g_2^i, g_{31}^i, g_{32}^i, g_{33}^i, g_{34}^i, g_4^i\}$	117
Figure 5-3: State evolution for the optimal MDP policy, case 4b (4 goals, $n_h=4$), starting from $s^i = \{\text{no goals set, all actions in history } toolbox_retrieval (a_1)\}$, for $s^i = \{g_1^i, g_2^i, g_3^i, g_4^i, a_1^i, a_2^i, a_3^i, a_4^i\}$	118
Figure 5-4: State Transition Diagram and transition matrix for work_motivation only, $n_h=0$	124
Figure 5-5: State Transition Diagram and transition matrix for button_1_inactive only, $n_h=0$	124
Figure 5-6: State Transition Diagram and transition matrices for blood_sugar_level and hydration_level only, $n_h=0$	125
Figure 5-7: Finite State Machine Diagram for case 5a Representation, fully-connected (not all links labeled), $n_h=1$	126
Figure 5-8: State/policy-action progression outcomes, column 1 from Table 5-13 (blood_sugar_level g_1 vs. work_motivation g_3), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$	131
Figure 5-9: State/policy-action progression outcomes, column 2 from Table 5-13 (blood_sugar_level g_1 vs. work_motivation g_3), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$	132
Figure 5-10: State/policy-action progression outcomes, column 1 from Table 5-14 (blood_sugar_level g_1 vs. hydration_level g_2), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$	134

Figure 5-11: State/policy-action progression outcomes, column 2 from Table 5-14
(blood_sugar_level g_1 vs. hydration_level g_2), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 135

Figure 5-12: State/policy-action progression outcomes, column 1 from Table 5-15
(blood_sugar_level g_1 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 138

Figure 5-13: State/policy-action progression outcomes, column 2 from Table 5-15
(blood_sugar_level g_1 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 139

Figure 5-14: State/policy-action progression outcomes, column 1 from Table 5-16
(work_motivation g_3 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 140

Figure 5-15: State/policy-action progression outcomes, column 2 from Table 5-16
(work_motivation g_3 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 141

Figure 5-16: State/policy-action progression outcomes, column 1 from Table 5-17
(blood_sugar_level g_1 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 144

Figure 5-17: State/policy-action progression outcomes, column 2 from Table 5-17
(blood_sugar_level g_1 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 145

Figure 5-18: State/policy-action progression outcomes, column 3 from Table 5-17
(blood_sugar_level g_1 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 146

Figure 5-19: State/policy-action progression outcomes, column 1 from Table 5-18,
($n_h=0$), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$ 150

Figure 5-20: State/policy-action progression outcomes, column 2 from Table 5-18,
($n_h=1$), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, full diagram (low-resolution overview) 151

Figure 5-21: State/policy-action progression outcomes, column 2 from Table 5-18,
($n_h=1$), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (left-most)..... 152

Figure 5-22: State/policy-action progression outcomes, column 2 from Table 5-18,
($n_h=1$), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (left-center)..... 153

Figure 5-23: State/policy-action progression outcomes, column 2 from Table 5-18,
($n_h=1$), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (center) 154

Figure 5-24: State/policy-action progression outcomes, column 2 from Table 5-18,
($n_h=1$), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (right-center) 155

Figure 5-25: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (right of right-center)	156
Figure 5-26: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (right-most)	157
Figure 6-1: Algorithm for calculating $p(H^j H^i)$	174
Figure 6-2: Algorithm for calculating likelihood of increase for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$...	181
Figure 6-3: Algorithm for calculating $r_{23}(a_k, a^i, H^i)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$	182
Figure 6-4: Zone partitioning using Definition #1, 2-D case, overhead view; fixed-base frames for agents are shown; zones are designated orange = occupiable by both, green = reachable by robot only, blue = reachable by human only, grey = non-occupiable.....	187
Figure 6-5: Action-zone partitioning using Definition #2, 2-D case, overhead view, stationary arm positions	189
Figure 6-6: Action-zone partitioning using Definition #2, 2-D case, overhead view, example with possible-conflict case	191
Figure 6-7: Algorithm for calculating $p(g_1^j g_1^i, a_k)$	199
Figure 6-8: Algorithm for calculating $p(b^j G^i, F^i, G^j, F^j, b^i, a_k)$	200
Figure 6-9: Algorithm for calculating $p(H^j H^i)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$	201
Figure 6-10: Algorithm for calculating $p(H^j H^i)$ for $H^i = \{^H a_{obs}^i\}$	201
Figure 6-11: Algorithm for calculating $p(f_z^j H^i, f_z^i, a_k, H^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$ and $H^i = \{^H a_{obs}^i\}$	202
Figure 6-12: Algorithm for calculating $p(f_z^j f_z^i, a_k)$ for $H^i = \emptyset$	203
Figure 6-13: Algorithm for calculating $p(d^j H^i, H^j, a^j, b^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$ and $H^i = \{^H a_{obs}^i\}$	204
Figure 6-14: Algorithm for calculating $noconflict(H^i, a, H^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$	204
Figure 6-15: Algorithm for calculating $noconflict(H^i, a, H^j)$ for $H^i = \{^H a_{obs}^i\}$	205
Figure 6-16: Algorithm for calculating a flag representing danger increase potential for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$	207

Figure 6-17: Algorithm for calculating a flag representing danger increase potential for $H^i = \{^H a_{obs}^i\}$ 208

Figure 6-18: Algorithm for calculating $r_{23}(a_k, a^i, H^i)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$209

Figure 6-19: Algorithm for calculating $r_{23}(a_k, a^i, H^i)$ for $H^i = \{^H a_{obs}^i\}$ 209

List of Tables

Table 3-1: MM-arm D-H parameters	49
Table 3-2: MM-arm poses	51
Table 3-3: Test session 1: distribution of task category combinations.....	59
Table 3-4: Expected relationship of test result data in test set 1	63
Table 3-5: Exhibited statistically-significant learning curves: test set 1 comparison	65
Table 3-6: Trends for correctness rates: objective data	67
Table 3-7: Trends for overall workload: TLX data	69
Table 5-1: Mission-goal transition probabilities for HIP	98
Table 5-2: First example representation, computer work then eat chip	105
Table 5-3: First example representation, computer work and eat chip (in any order) ...	105
Table 5-4: Second example representation, computer work then eat chip	106
Table 5-5: Third example representation, computer work then eat chip	107
Table 5-6: Domain Representation of actions a_k^i	109
Table 5-7: Domain Representation of goal-objectives	110
Table 5-8: Memory requirements, case 4a (1 goal, $n_h=8$)	112
Table 5-9: MDP policy illustrating action-history use, case 4b (4 goals, $n_h=4$)	115
Table 5-10: MDP policy illustrating action-history use, case 4c (7 goals, $n_h=0$)	115
Table 5-11: Domain representation of actions a_k^i	121
Table 5-12: Domain representation of goal-objectives	121
Table 5-13: Impact of reward weightings, eat_chip (a_1) / blood_sugar_level (g_1) vs. computer_work (a_3) / work_motivation (g_3).....	130
Table 5-14: Impact of reward weightings, eat_chip (a_1) / blood_sugar_level (g_1) vs. drink_soda (a_2) / hydration_level (g_2)	133
Table 5-15: Impact of reward weightings, eat_chip (a_1) / blood_sugar_level (g_1) vs. push_button (a_4) / button_1_inactive (f_1).....	136

Table 5-16: Impact of reward weightings, computer_work (a_3) / work_motivation (g_3) vs. push_button (a_4) / button_1_inactive (f_1)	137
Table 5-17: Impact of transition probabilities, eat_chip (a_1) / blood-sugar_level (g_1) vs. computer_work (a_3) / work_motivation (g_3).....	142
Table 5-18: Impact of transition probabilities, $n_h=0$ through $n_h=2$	147
Table 5-19: Impact of transition probabilities, $n_h=0$ and $n_h=3$	148
Table 6-1: Use of Human State Information in RAC	172
Table 6-2: Zone partitioning using Definition #2, robot's zones as related to button 1 (${}^R b_1$), button 2 (${}^R b_2$), and unstow position (${}^R u_1$)	191
Table 6-3: Zone partitioning using Definition #2, human's zones as related to button 1 (${}^H b_1$) and button 2 (${}^H b_2$).....	192
Table 6-4: Domain Representation of human actions ${}^H a_x^i, \left(\left\{ {}^H a_{obs}^i, {}^H a_{HIP}^i \right\} \right)$	192
Table 6-5: Domain Representation of (robot) actions a_k	193
Table 6-6: Human zone partitioning using Definition #2.....	194
Table 6-7: Robot zone partitioning using Definition #2.....	194
Table 6-8: Domain Representation of human actions ${}^H a_x^i, \left(\left\{ {}^H a_{obs}^i, {}^H a_{HIP}^i \right\} \right)$	195
Table 6-9: Domain Representation of g_z^i, f_z^i goal-objectives	195
Table 6-10: Domain Representation of (robot) actions a_k	196
Table 6-11: Collision spread according to policy, robot action a_k , in-progress action ..	211
Table 6-12: Collision spread according to policy, robot action a_k , future-predicted action	211
Table 6-13: Collision spread versus value, robot action a_k , in-progress action;	213

List of Appendices

Appendix A.....	226
MichiganMan(ipulator) Arm Characteristics	226
Specific Measured D-H parameters of the Michigan Manipulator	227
Forward Kinematics Equations	228
Inverse Kinematics Equations (numerical solution methods for generalized solutions / position-only waypoint).....	228
MM-Arm Dynamics & Singularity Identification.....	237
Appendix B	241
Task Timelines for Test Sets.....	241

List of Symbols

${}^A P$	translation vector, location in frame A
${}^A P_{BORG}$	translation vector, locates frame B's origin with respect to frame A
${}^A R_B$	rotation matrix from frame B to frame A
${}^A T_B$	4x4 transformation matrix
\hat{X}_i, \hat{Z}_i	unit vectors attached to manipulator arm linkages (see Craig [1])
a_{i-1}	distance from \hat{Z}_{i-1} to \hat{Z}_i measured along \hat{X}_{i-1}
α_{i-1}	angle from \hat{Z}_{i-1} to \hat{Z}_i measured about \hat{X}_{i-1}
d_i	distance from \hat{X}_{i-1} to \hat{X}_i measured along \hat{Z}_i
θ_i	(variable) angle from \hat{X}_{i-1} to \hat{X}_i measured about \hat{Z}_i
r_{xy}	parameter in a rotation matrix ${}^A R_B$ at row x column y
p_x	parameter in a translation vector ${}^A P$
${}^A J(\Theta)$	Jacobian matrix with respect to frame A
${}^A v$	6x1 vector with Cartesian velocity with respect to frame A
${}^A v$	3x1 linear velocity vector of the tooltip with respect to frame A
${}^A \omega$	3x1 angular velocity vector of the tooltip with respect to frame A

Θ	$n \times 1$ vector of joint angles of the manipulator (with n joints)
${}^A \mathfrak{F}$	6×1 force vector with respect to frame A
${}^A F$	3×1 (linear) force vector of the tooltip with respect to frame A
${}^A N$	3×1 (angular) moment vector of the tooltip with respect to frame A
τ	$n \times 1$ vector of joint torques of the manipulator
DI	danger index
$f_D(s)$	distance factor
$f_V(v)$	velocity factor
$f_I(I_{CP})$	inertia factor
k_D, k_V	constant values used to calculate the danger index
D_{min}	minimum allowable distance between human and robot
D_{max}	distance between human and robot at which safety is assured
V_{min}	minimum relative velocity
V_{max}	maximum relative velocity
I_{cp}	effective inertia of the critical point
I_{max}	maximum safe value of robot inertia
v	approach velocity
s	distance from critical point to nearest point on person
S_i	Markov Chain state (value)
O_i	Hidden Markov Model observation (value)
S	set of possible states s^i

s^i	Markov Decision Process state i
A	set of available actions a_k
a_k	action-choice
$R(s^i, a_k)$	set of state-dependent rewards for performing action a_k at state s^i
$T(s^i, a_k, s^j)$	set of all transition probabilities $p(s^j/s^i, a_k)$
$p(s^j/s^i, a_k)$	the probability of a state s^j outcome, given a_k executed from a state s^i
$\pi(s^i)$	optimal policy for state s^i
n_s	number of states in S
n_a	number of actions in A
$V(s^i)$	value of state s^i
λ	discount factor
A_i	(PO)MDP action (value)
R_i	(PO)MDP reward (value)
o^i	observation vector at POMDP state S^i
S^i	POMDP vector of all possible states s^i and beliefs o^i associated with them
r_i	NASA TLX rating
W_i	NASA TLX weight for load source i
R_i	NASA TLX adjusted ratings
R_W	NASA TLX weighted rating, overall workload
T_f	arrival time to reach goal pose from the initial pose
T_s	initial time of goal selection

t	time
N	number of joint angle divisions/waypoints, constant value
$^H a_{HIP}$	next intended human action, predicted future intent
$^H a_{obs}$	currently-observed human action, in-progress
n_h	length of action-history (number of actions)
$a_{n_h+1}^i$	human's current or in-progress action in state s^i
G^i	set of mission goal states in state s^i
F^i	set of high-priority goal states in state s^i
A^i	abbreviated action-history of observed actions in state s^i
n_g	number of mission goals in G^i
g_z^i	mission goal z in G^i
n_f	number of high-priority goals in F^i
f_z^i	high-priority (interruptive) goal z in F^i
a_k^i	action k in abbreviated action-history A^i
n_z^i	minimum number of actions to complete goal g_k^i for a sequence in A^i
p_{kz}	probability of action a_k not transitioning a mission-goal g_z^i
n_{bk}^i	number of nonzero probabilities in tensor $T(s^i, a_k, s^j)$ for a_k and state s^i
n_{Tk}^i	number of state transitions s^j with non-zero probability for $T(s^i, a_k, s^j)$
n_b^i	total number of possibilities of transition minus the number of actions
ϵ_{kz}	small nonzero probability of transitioning to states not defined by p_{kz}

n_{ϵ}^i	number of low-likelihood states transitions from state s^i
$R(s^i)$	reward function for state s^i
α_z	reward weight for mission goal g_z^i
β_z	reward weight for high-priority goal f_z^i
r_x	reward function x
k_z	cost associated with f_z^i
γ	discount factor
p	probability
λ_{kz}	transition probability weight for impact of action a_k on mission goal g_z^i
g_z	label for value of mission goal g_z^i
f_z	label for value of high-priority goal f_z^i
H^i	set of human companion states in state s^i
R^i	set of robot states in state s^i
d^i	discretized danger index attribute in s^i
${}^R A$	set of available robot actions
${}^R n_a$	number of robot actions in ${}^R A$
${}^H A_i$	set of human actions in H^i
${}^H A$	set of available human actions
${}^H a_{obs}^i$	human's in-progress action output by the observer module in state s^i
${}^H a_{HIP}^i$	human's action output from the HIP policy in state s^i

a^i	robot's current action in R^i
b^i	robot's current action-status in R^i
${}^R Z_i$	set of robot zones (zone data) in state s^i
${}^R Z_k^i$	robot zone k in ${}^R Z_i$
${}^R Z$	set of robot zones
${}^R n_z$	number of robot zones in ${}^R Z$
${}^H Z_i$	set of human zones (zone data) in state s^i
${}^H Z_k^i$	human zone k in ${}^H Z_i$
${}^H Z$	set of human zones
${}^H n_z$	number of human zones in ${}^H Z$
D	set of all discretized danger index values
D_{max}	maximum value of d^i in D
DI_x	threshold value x of danger index
$p_{H a_x^i}$	probability that ${}^H a_x^i$ will not transition
w_x	reward weight for reward function R_x
w_{xy}	weight on reward function r_{xy}
r_{xy}	reward function xy
w_d	weighted penalty for binary d^i
a_y	label for value of robot action
${}^A b_m$	label for button location m assigned to agent A
${}^A u_m$	label for unstowed position (location) m assigned to agent A

z_x	label for zone x
${}^I x, {}^I y$	inertial coordinate system axes
${}^A a z_x$	action-zone of agent A

Abstract

Emerging robotic systems are capable of autonomously planning and executing well-defined tasks, particularly when the environment can be accurately modeled. Robots supporting human space exploration must be able to safely interact with human astronaut companions during intravehicular and extravehicular activities. Given a shared workspace, efficiency can be gained by leveraging robotic awareness of its human companion. This dissertation presents a modular architecture that allows a human and robotic manipulator to efficiently complete independent sets of tasks in a shared physical workspace without the robot requiring oversight or situational awareness from its human companion. We propose that a robot requires four capabilities to act safely and optimally with awareness of its companion: sense the environment and the human within it; translate sensor data into a form useful for decision-making; use this data to predict the human's future intent; and then use this information to inform its action-choice based also on the robot's goals and safety constraints. We first present a series of human subject experiments demonstrating that human intent can help a robot predict and avoid conflict, and that sharing the workspace need not degrade human performance so long as the manipulator does not distract or introduce conflict. We describe an architecture that relies on Markov Decision Processes (MDPs) to support robot decision-making. A key contribution of our architecture is its decomposition of the decision problem into two parts: human intent prediction (HIP) and robot action choice (RAC). This decomposition is made possible by an assumption that the robot's actions will not influence human intent. Presuming an observer that can feedback human actions in real-time, we leverage the well-known space environment and task scripts astronauts rehearse in advance to devise models for human intent prediction and robot action choice. We describe a series of case studies for HIP and RAC using a minimal set of state attributes, including an abbreviated action-history. MDP policies are evaluated in terms of model

fitness and safety/efficiency performance tradeoffs. Simulation results indicate that incorporation of both observed and predicted human actions improves robot action choice. Future work could extend to more general human-robot interaction.

Chapter 1

Introduction

Effective human-robot interaction can make hazardous and potentially repetitive work traditionally done by humans safer and more productive by offloading work to the robot. In a space environment, risk to an astronaut is cumulative with exposure time to high-risk situations such as extravehicular activities (EVA). A highly-capable robotic system could separately accomplish the simpler, well-modeled tasks usually performed by human astronauts on EVA or IVA (intra-vehicular activity). Adding intelligence to these robotic systems will allow them to work alongside humans to reduce supervision overhead for the astronaut(s) thus increasing overall productivity. Human-robot interaction (HRI) scenarios often assume that the human and robotic agents' workspaces have little to no overlap while performing their tasks. However, under some local working conditions – for example, servicing a spacecraft or constructing a lunar habitat – the only way to shorten the work schedule to a reasonable timescale would be to allow humans and robots to *share* their physical workspace. Yet, allowing this overlap introduces safety issues that must be addressed.

Modern sensor systems can now enable robots to reliably sense nearby humans in real-time with sufficient accuracy to support safe close-proximity operations. Further, this information can be integrated into the robot's decision-making processes to allow the robot to be made "human-aware," customizing its reactions based on its human companion's observed and expected activities. Doing so allows us to relax the extremely conservative safety constraint of separating agent workspaces, when the input is within the bounded error that the decision-making scheme can support. Towards achieving safe yet efficient interaction, in this dissertation we develop and evaluate an autonomous framework for determining a robotic manipulator's optimal actions in real-time when interacting in close physical proximity to a human in a shared workspace environment.

This framework allows the robot to purposefully choose to avoid physical and mental conflicts with a human companion while each of these agents performs tasks to complete their own separately-assigned goals. We apply this framework to the space environment, and focus on interaction scenarios where the human does not or should not need to divert their attention to the robot. The robot is meant to unobtrusively work around the human rather than directly collaborate on task completion, minimizing requirements for the human to maintain situational awareness of the robot and its goals. We assume that this strategy will minimize mental-workload thus increase efficiency relative to models in which the human must supervise or actively avoid conflicts with the robot.

1.1 Motivation

Astronauts in a space environment are exposed to substantially more risk on a day-to-day basis than are their Earth-bound counterparts. Risk on EVA is particularly high, so mission directors seek to minimize the frequency and duration of astronaut EVA. For intra-vehicular activities (IVA), risk is linked to the confined, zero-gravity pressurized habitat module environment. In an emergency, or when a set of hard task deadlines exist, an astronaut needs to be able to travel from point A to point B with minimal delay. In this case, overall productivity is increased by keeping the environment clean and clear of the most common traversal paths as much as possible.

Some of the HRI challenges in space stem from the nature of the microgravity environment. Because objects float in space, there is no preferred body orientation, e.g., feet on the ground. Additionally, given even minor disturbances such as air currents, objects will not remain in a fixed location unless they are strapped down or actively held in place (see Figure 1-1). A human or robot also will freely float in space, a condition that can create a highly dynamic environment that can only be modeled and managed with constraints to simplify the large range of all possible interactions. Generally, astronauts will pick a preferred orientation based on characteristics of their environment then try to restrict their lower-body motion through anchoring devices when performing tasks in microgravity (see Figure 1-2) [2]. Fixtures such as restraining belts and foot/base restraints may also be available both for astronaut and robotic entities, creating a “fixed

base” condition that simplifies the environment and also improves force application capability.



Figure 1-1: Astronaut working in the International Space Station’s Kibo laboratory (from: http://www.nasa.gov/mission_pages/station/expeditions/expedition30/science_from_space.html , Photo credit: NASA). Astronaut’s feet are stabilized by elastic straps mounted to the wall.

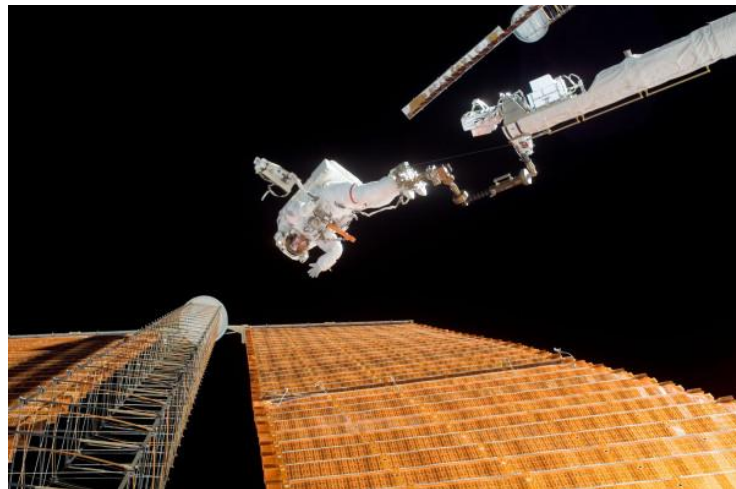


Figure 1-2: EVA spacewalk finishing repairs on a torn solar array (from: <http://twitpic.com/7782v4> , Photo credit: Douglas H. Wheelock / NASA). Astronaut motion is stabilized by an articulating portable foot restraint (APFR).

Although astronauts have demonstrated the ability to adapt to unexpected situations as they are encountered, numerous tasks are also executed “by the book” through scripts that may need to be repeated many times for a variety of maintenance or mission tasks. Figure 1-3 and Figure 1-4 give examples of these scripted procedures.

EVA 1 SUMMARY TIMELINE

PET HR : MIN	IV/SSRMS	EV1	EV2	
00:00	<ul style="list-style-type: none"> SSRMS at ST install position near ESP-2 	EGRESS/SETUP (01:00) <ul style="list-style-type: none"> Post Depress and Egress Install COLTs 	EGRESS/SETUP (01:00) <ul style="list-style-type: none"> Post Depress and Egress SSRMS Setup 	00:00
01:00	<ul style="list-style-type: none"> SSRMS Maneuver to PM FRAM release ESP-2 PM FRAM inhibits in place 	REMOVE PM FRAM FROM ESP-2 (00:30)	REMOVE PM FRAM FROM ESP-2 (00:30) <ul style="list-style-type: none"> Rotate PM 	01:00
02:00	<ul style="list-style-type: none"> SSRMS Maneuver to ESP-2 clearance (-00:30 to PLB) 	TRANSFER PM FRAM TO PLB (00:30)	TRANSFER PM FRAM TO PLB (00:30)	02:00
		INSTALL PM FRAM ON LMC (00:30)	INSTALL PM FRAM ON LMC (00:30)	
	<ul style="list-style-type: none"> LMC RRM FRAM inhibits in place 	<ul style="list-style-type: none"> Swap Crew on SSRMS REMOVE RRM FRAM FROM LMC (00:20)	<ul style="list-style-type: none"> Swap Crew on SSRMS REMOVE RRM FRAM FROM LMC (00:20)	
03:00	<ul style="list-style-type: none"> SSRMS Maneuver to EOTP (-00:30) 	TRANSFER RRM FRAM TO EOTP (00:30)	TRANSFER RRM FRAM TO EOTP (00:30)	03:00
	<ul style="list-style-type: none"> EOTP RRM FRAM inhibits in place 	INSTALL RRM FRAM ON EOTP (00:20)	INSTALL RRM FRAM ON EOTP (00:20)	
04:00	<ul style="list-style-type: none"> SSRMS Maneuver to ESP-2 egress position (-00:15) ELC-2 MISSE 8 inhibits in place 	SSRMS CLEANUP (00:45)	MISSE 8 ORMATE-III R/W INSTALL ON ELC-2 (01:20)	04:00
	<ul style="list-style-type: none"> SSRMS Maneuver to park position SSRMS RWS in Backup 	FGF PDGF TROUBLESHOOTING (00:35)		
05:00		PMA3 COVER INSTALL (00:45)	PMA3 COVER INSTALL (00:45)	05:00
06:00		CLEANUP/INGRESS (00:45) <ul style="list-style-type: none"> Cleanup (00:15) Ingress and Pre-Repress (00:30) 	CLEANUP/INGRESS (00:45) <ul style="list-style-type: none"> Cleanup (00:15) Ingress and Pre-Repress (00:30) 	06:00
06:30				06:30

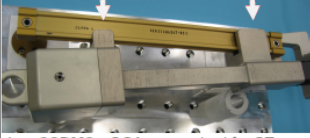
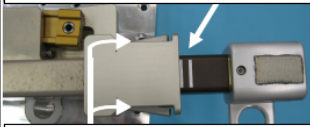
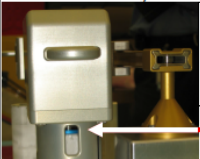
FS 7-3

EVA/135/FIN 1

EVA 1

Figure 1-3: Summary Timeline for Two Astronauts on EVA1, Flight STS-135 [3]

EVA 1 EGRESS/SETUP (01:00) (Cont)

IV/SSRMS	EV1	EV2
<div data-bbox="349 304 581 336" style="border: 1px solid black; padding: 2px; text-align: center;">Both "U" brackets engage HR</div>  <div data-bbox="297 493 604 525" style="font-size: small;">4. SSRMS: GCA as required for ST install</div> <div data-bbox="297 535 604 556" style="border: 1px solid black; padding: 2px; text-align: center;">Two lines visible after slider installation</div>  <div data-bbox="297 682 604 703" style="border: 1px solid black; padding: 2px; text-align: center;">Both slider paddles popped up</div> <div data-bbox="297 703 604 745" style="font-size: small;">5. SSRMS: Maneuver to APFR Install position</div> <div data-bbox="297 756 604 798" style="font-size: small;">6. SSRMS: Notify EV: Brakes on, ready for APFR install</div> <div data-bbox="297 798 604 955">  <div data-bbox="495 798 604 903" style="border: 1px solid black; padding: 2px; font-size: x-small;">Washer flush w/end of contingency pin housing</div> </div>	<div data-bbox="657 273 958 367" style="border: 2px solid black; padding: 5px; text-align: center; font-size: x-small;"> WARNING Stay 1 ft from UHF Antenna Avoid contact with sharp edge on ESP-2 HR 8012 </div> <div data-bbox="613 378 1006 399" style="font-size: x-small;">INSTALL COLTS (00:45)</div> <div data-bbox="613 399 1006 945" style="font-size: x-small;"> <ol style="list-style-type: none"> 1. Translate to ESP-2 aft/port (low road) 2. Stow Crewlock Bag on ESP-2 HR 8015 (aft/port/zenith) 3. Release gap spanner between Lab and ESP-2, stow with RET 4. Retrieve long COLT from Crewlock Bag 5. Install Long COLT onto FRAM (ISS fwd) <ul style="list-style-type: none"> <input type="checkbox"/> Both "U" brackets engage HR <input type="checkbox"/> Two lines visible after slider installation <input type="checkbox"/> Both slider paddles popped up <input type="checkbox"/> Washer flush w/end of contingency pin housing (gap < washer thickness OK) 6. Remove COLT cap, stow in trash bag 7. Retrieve RET from COLT cap, stow on MWS 8. Retrieve short COLT from C/L Bag; install onto FRAM (ISS aft) <ul style="list-style-type: none"> <input type="checkbox"/> Both "U" brackets engage HR <input type="checkbox"/> Two lines visible after slider installation <input type="checkbox"/> Both slider paddles popped up <input type="checkbox"/> Washer flush w/end of contingency pin housing (gap < washer thickness OK) 9. Remove COLT cap, stow in trash bag 10. Retrieve RET from COLT cap, stow in Crewlock Bag </div>	<div data-bbox="1055 273 1356 367" style="border: 2px solid black; padding: 5px; text-align: center; font-size: x-small;"> WARNING Stay 1 ft from UHF Antenna Avoid contact with sharp edge on ESP-2 HR 8012 </div> <div data-bbox="1016 378 1421 399" style="font-size: x-small;">SSRMS SETUP (00:45)</div> <div data-bbox="1016 399 1421 945" style="font-size: x-small;"> <ol style="list-style-type: none"> 1. Translate to ESP-2 fwd (high road) 2. GCA SSRMS as required for ST install <div data-bbox="1055 525 1380 567" style="text-align: center; font-size: xx-small;"> NOTE Monitor clearance to AGB on FHRC </div> <ol style="list-style-type: none"> 3. Safety Tether swap to SSRMS LEE HR <ul style="list-style-type: none"> <input type="checkbox"/> Ball closed, slider locked, double locked, reel unlocked 4. Attach green hook onto ESP-2 HR 8011 _____ 5. Give SSRMS GO for maneuver to APFR Install position 6. On SSRMS GO, install WIF Adapter into SSRMS PFR socket with tether point toward LEE (PIP Pin required; not hitch pin) </div>

FS 7-18

EVA135/FIN

Figure 1-4: Partial Timeline for “Install COLTS” and “SSRMS Setup” Tasks on EVA1, Flight STS-135 [3]

While human productivity can be increased to an extent by careful scheduling and improved workspace layouts, astronaut productivity might also be increased by reassigning the most ‘dull and dirty’ tasks to highly-capable robotic systems. For example, a robot could complete some of the setup and cleanup tasks for EVA, or station cleaning and upkeep tasks on IVA. Overall productivity could also be increased further by adding intelligence to these robotic systems such that neither teleoperation nor close supervision is necessary. A fundamental assumption of this work is that eliminating the need for explicit communication or oversight reduces the astronaut’s mental workload or situational awareness with respect to dealing with the robotic system. Under circumstances where the robot and astronaut need to accomplish tasks in a common space such as a particular habitation module, productivity could be increased even further if we can lift the usual safety restriction of needing to keep the human and robotic workspaces separate. In such cases, the robotic system’s presence must neither introduce

unacceptable levels of risk nor interfere with humans in a way that reduces productivity or that introduces unacceptable annoyance or workload.

Including this capability in current systems would be immediately useful. NASA has deployed two robotic platforms to the ISS, one manipulator (Robonaut 2) [4] and one free-flying observer (SPHERES) [5], that are beginning to support experimental investigations of HRI on EVA. Both of these platforms are currently controlled primarily by teleoperation or some form of oversight with limited autonomy, acting generally nearby but not within an astronaut's immediate reach. Robonaut 2 moves slowly, is well-padded, and includes an onboard sensor system that immediately safes the robot in place if an unexpected impact occurs. SPHERES is small and moves slowly. Neither system can yet sense humans on their own, nor do they include autonomy that can perform human-avoidance should this information be supplied. Currently, any needed deconfliction must be handled by their human operators. Adding these capabilities would greatly enhance their performance and allow their use without diverting attention of the astronauts in orbit from other tasks. Such capabilities could also enhance performance for interactive mission tasks with the astronauts.

Human-robot interaction in a common workspace requires three basic capabilities. First, a robotic system must be designed to perform the set of tasks that have been off-loaded from the astronaut. Such a robotic system must be equipped with sufficient sensing and force/torque application capabilities to effectively and autonomously execute each task. Second, the robot system must be able to operate independently for extended time periods, requiring a basic capability to identify, prioritize, sequence, and execute tasks without supervision. Third, a safety management system must mitigate risk to sufficient levels for robot and astronaut to occupy the same physical workspace. Significant research has been devoted to the first two of these required capabilities [6,7,8,9] [10,11,12,13]; the third is currently being explored as state-of-the-art research [14,15,16]. This thesis focuses on achieving the second and third challenges: task selection and execution in the presence of safety constraints. We use the danger index for safe trajectory execution from Ref. [14] as a metric for safety, and we use a Markov

Decision Process (MDP) with passively sensed human state data to find such a policy (rather than a Partially-Observable MDP with explicit communication as in Ref. [15]).

For this work, safety is defined as “the condition of being protected against physical... or other types or consequences of failure, damage, error, accidents, harm or any other event which could be considered non-desirable, [or otherwise] the control of recognized hazards to achieve an acceptable level of risk” [17]. We categorize three different types of safety in our robotics research: electromechanical system, software system, and environmental. Electromechanical system failures that might compromise safety include electrical component failure, loss of power, physical device failures and consequences of wear-and-tear on the joints, linkages, and so forth. Software failures include loss of communication between devices caused by conditions such as sensor dropout or unaccounted-for data signal lag, and bugs in the operational code. As a simplifying assumption, our research presumes that mechanical and software safety can be assured for the duration of the mission. Thus, we focus on environmental safety issues, primarily resulting from the possibility of physical conflict.

A robot may encounter the potential for physical and mental conflict during interactions with the physical environment, including other agents and itself. A physical conflict results when there is potential for collision with other agents or the environment. We define a ‘mental’ conflict to represent situations in which an object to be sensed is occluded from the sensor. For example, the robot can move its arm between the astronaut’s eyes and a target of interest to the astronaut.

Recent research in safety management has focused on the types of damage and injury that can occur with physical collision, as well as the creation of useful safety metrics to catalogue and classify risks [18,19]. Other research has focused on reactive strategies for robotic systems: how to design them to be collision-safe, such as mechanical designs that reduce the inertia of a hit when a hit occurs, or control system feedback that allows a manipulator arm to stop in place or reverse its trajectory upon sensing an imminent collision [20,7]. We focus on an additional mitigation measure that could be combined with these efforts for maximal effect: the value of attempting to strategically plan the

robot's actions to avoid potential conflicts as well as achieve task-level goals based on *a priori* predictions of companion near-term intent.

We build from previous work in defining metrics for real-time conflict avoidance [14] and extend this work to predict future conflicts, enabling the robot to optimize action choices over predictions of human companion intent as well as its own task-level goals. Human intent prediction (HIP) is itself a challenging endeavor. While it is unrealistic to precisely predict specific movements over time, predictions at a higher level, of a human's *goals*, are far more reasonable, especially when given substantial knowledge of the companion's possible goals and the assumption of a known or fully-observable environment. In space, both the internal and external environments (IVA and EVA) are carefully modeled in advance, and most tasks are accompanied by detailed checklists and procedures for the astronauts to follow. Astronauts are also extensively trained on these procedures prior to each mission. A human astronaut's actions and goals are therefore expected to be more predictable than might be possible in most Earth-based HRI scenarios. Further, the gross motion sequences associated with task completion will also be more predictable and known for space applications because astronauts train extensively (e.g., in neutral buoyancy) for each EVA task and subtask in facilities with layouts similar to habitation modules and are closely observed as they do so.

Despite increased predictability, even experienced astronauts do not always follow the checklists exactly, and reacting to any anomaly, large or small, will likely result in intentional deviation from predicted sequences. Further, some scenarios are unscripted, as when the astronaut is eating a meal or just relaxing. This is a feature, not a bug: we want astronauts to be free to adapt to evolving circumstances. However, if we want a robot to share the environment with a human, then to help maintain safety, we must account for the uncertainty associated with human choice. If the robot cannot accurately predict the human's intent, this may annoy the astronaut or reduce overall efficiency, but safety can still be maintained so long as we also include reactive strategies that can sense and accommodate the actual physical trajectories of the human's motion.

We hypothesize that by predicting a human's gross motions from intent, we can program a robot to intelligently use this information to act in a manner that is optimal with respect

to both task-level goals and safety metrics. Therefore, a human-robot team in a shared workspace with separate goals can maximize overall productivity without compromising safety when there is no direct communication or supervision of the robot. If the robot is indeed successful at avoiding conflicts without supervision, the astronaut will have offloaded tasks without new overhead so that s/he may accomplish remaining tasks as efficiently as if the robot were not occupying the shared workspace.

1.2 Problem Statement

Given a human and robotic manipulator arm with separate goals sharing a common workspace, this thesis studies the problem of enabling an autonomous human-aware robot to act optimally and safely so that the robot achieves its goals and has little to no impact on the motion or goals of the human.

We propose that a robot need only do four things to act safely and optimally with awareness of the human: sense the environment and the human within it; translate sensor data into a form useful for decision-making; use this data to predict the human's future intent; and then use this information to inform its action-choice based also on the robot's goals and safety constraints. First, the robot must sense its environment, including the position and pose of its human companion. This can be a nontrivial problem depending on the type of sensory data and accuracy needed from the available sensors as well as the possibility for sensor occlusion or environmental noise. It can also be difficult to quickly and automatically identify a human within a cluttered visual scene, let alone to accurately isolate and extract their physical position and pose for estimation. However, once available, this data can be used to ensure trajectory-level safety constraints are respected [21], giving the robot a local physical awareness of the human. While acquiring this sensor data presents challenges, human sensing is not the focus of this thesis so we presume such data is available to the robot. Once human and environment states can be reliably observed, the robot must translate this information into a semantic representation that can support the robot's decision-making processes.

Because humans can move and change directions of motion quickly, a robot will have significant uncertainty regarding future pose and position if armed only with observations of current pose and position of its human companion. If a robot could translate its

observations of past and current human state to predictions of future state, this uncertainty could be reduced. Such a “human-aware” robot might enjoy substantially improved safety and efficiency relative to a robot without such models, especially if such determinations could be taken into account during task planning and scheduling. Knowing and responding to the human’s current state at this level allows early reactions that reduce the chances for near-term conflict. Accurate prediction of the human’s future intent allows the robot to select actions expected to offer greater reward over a longer-term. To achieve such awareness, the robot needs to translate its sensor data, e.g. a history of human position and pose estimates, to a goal-seeking behavioral state, e.g. a history of the human’s goals and actions. It must then decompose that goal-seeking behavior into expected future motions in a manner that informs robot decision-making.

For the robot to exhibit goal-driven behavior, it must have the ability to plan and schedule its actions at an abstract level, translating goals, observations, and predicted intent into safe and optimal actions. To do so, the robot must be capable of representing the traversable environment, including human companion state, its own state, and the impact of its own goals and actions. It then must incorporate and use this information to devise a safe, optimal action plan or policy. Defining optimality at this higher level is a challenge, as is selecting useful metrics for safety and goal completion. In a deterministic environment, a plan can be specified as a linear sequence of actions to accomplish a mission, with pre-scripted trajectories optimized offline that allow the robot to accomplish each planned activity. In an uncertain but observable environment, at each step the robot must sense the state, and then act in an optimal manner conducive to both safety and goal accomplishment for each reachable state. As our environment is uncertain, we seek to combine these two approaches: the robot plans in advance, but updates state estimates based on real-time sensor feedback to re-direct the robot toward safety-preserving actions or alternate goal-seeking actions as needed. Providing a safe and efficient real-time response when predictions are wrong requires that the robot’s autonomy architecture support task and trajectory planning and repair capabilities.

Human intent prediction (HIP) and robot action-choice (RAC) decision-making, the third and fourth challenges described above, are the focus areas of our research. We assume

that raw sensor data is handled by another process and that optimal robot motion trajectory sequences can be calculated offline and stored in a database for online use. With respect to HIP and RAC, we seek to enable the robot to understand what the human wants (their goals) and is trying to do (their actions), and to use this knowledge to determine the robot's optimal action-choice in each state to accomplish goals while avoiding destructively interfering with a human companion.

As a precursor to our investigation of HIP and RAC, we studied the viability of allowing a human and robot to pursue independent goals without communication in a shared workspace through a series of baseline human subject experiments. The challenge in this initial work was to collect evidence supporting or disputing a hypothesis that such operations could be both safe and of minimal impact to the human under these circumstances. Because we restrict ourselves in this research to interaction cases where the human's and robot's goals are separated and no direct collaboration occurs, we believe that it is reasonable to assume that the human does not need to divert attention to internally model the robot's behavior, or track or acknowledge the robot's actions. This is a reasonable assumption when the astronaut can trust the highly-capable robotic system to work without his or her oversight (i.e., the robot works autonomously in such a way that there is no need for situational awareness of the robot's tasks). The robot's goals and actions can then be ignored by the astronaut. If this is true, explicit communication should not be necessary during operations. This also implies that an optimal choice for a human-aware robot will never negatively impact the human as the robot works around the human. However, this idea of interaction without a need for explicitly communicating is an unusual and significant assumption. Most research assumes that a shared mental model and situational awareness must exist. A need to show that this may not be necessary motivated our initial experiments, which was used to inform our subsequent work to address HIP and RAC.

We make a number of simplifications in this work to scope the effort appropriately. First, we assume robot tasks always have a lower priority than human tasks, and the human is a non-adversarial agent. The first simplification drives the robot to select actions that do not interfere with predicted human activities, even if alternate, potentially-

interfering actions would derive greater reward with respect to the robot's goals. The second simplification allows the robot to presume the human will ignore the robot so long as it doesn't interfere. This implies that the human's actions thus productivity will be approximately the same alone versus in a shared space so long as no conflict occurs. Further, this implies that overall productivity for the human-robot team will be at least as high as the productivity achieved should the human or robot act alone as long as the human and robot avoid conflict. If the robot completes its own separate goals in addition to the human's efforts, more goals will be completed overall. If the robot completes none of its goals, the human will still have the same level of productivity.

We make several assumptions about our problem space:

- We assume that the robot has full observability of the human and its environment. Proper sensor placement and data parsing makes this a reasonable assumption. This simplifying assumption allows us to avoid reasoning about hidden states.
- We assume that the space environment is itself deterministic. This is reasonable because the EVA and IVA environments that we assume our astronauts will work within are completely engineered. Procedures for nominal and off-nominal scenarios are developed well in-advance of any circumstance requiring such planned action. This allows us to ignore any possibility of needing to perform automated environmental learning in our work, and allowing specification of all domain knowledge offline.
- We assume that the interaction scenarios of interest can be modeled with a "closed" (complete) action set as well as the specification of factors that lead the human to his/her choice of actions in this closed action set. This is an expansion of the previous assumption.
- We assume that each action is of sufficiently short duration that it can complete without interruption unless risk of an unexpected conflict occurs.
- We assume that humans generally act as rational agents and that this rationality can be exploited. This allows us to treat any random, unpredictable behaviors due to the uncertainty inherent in the human-sensing problem as bounded noise in our human model.

Simplifications to further constrain the problem space to a reasonable size for the space robotics application include:

- The human's most-likely structured action sequences are known in advance for EVA operations, or can be informed by long-term observation of human behavior for IVA collaboration. This eliminates the need to learn and match poses to actions during real-time operations.
- Human motion is so cumbersome and restricted on EVA due to the spacesuit that unscripted actions are unlikely to be attempted. This reduces the set of possible mismatches or uncertainty in human action-recognition.
- The human model does not include a model of the robot state, nor does it need to contain such a model. This assumption is valid so long as the human is indeed not distracted or impeded by the robot.
- We have sufficient memory and computational resources for robot decision-making and the storage of offline-calculated information. The assumptions of full observability and complete knowledge (thus pre-computation of plans/policies) make this assumption reasonable.

1.3 Research Objectives

This research studies challenges in modeling and decision-making associated with a robot operating in a workspace shared by a human. Specifically, the robot must accomplish its objectives and avoid environmental conflict during physically-proximal HRI when conflict-avoidance is of significant importance for safe, efficient operations. In this work, we assume that the human and robot have distinct goals, do not communicate, and that the robot must not interfere with any activity in which the human is engaged.

The goal of this research is to build a robot decision-making scheme that can predict a human's current and future intent, based on a known history of actions and goal state, and then use this knowledge to schedule the robot's activities. We also wish to characterize the full system in a manner that supports baseline safety and system-level performance evaluation.

Under the assumptions and constraints above, we make the following hypotheses, which drive each chapter of this work:

- Productivity of a collaborating human-robot team operating in a shared workspace can be maximized when the human has no need to supervise the robot. Supervision is no longer necessary when a robot can autonomously operate safely and efficiently with acceptable or no impact on its companion's productivity.
- If a human's actions can be classified as rational to within a known and bounded uncertainty, we can find a model that will assure an acceptable level of risk introduced by the robot during human-robot team operations.
- A robot can predict companion intent by identifying actions based on sensor observations without relying on explicit communication, then recognizing those observed actions as part of a sequence.
- The use of predicted companion intent results in improved real-time robot action choices over those made without it, when the relative worth of the intent data is known and both are supplied to a procedure derived from a sufficient domain model.

1.4 Approach

To accomplish the above objectives, we investigate a series of research thrusts that collectively support the safe, autonomous HRI challenges posed above. First, we devise, conduct, and analyze a set of human-subject experiments to validate our concept of realizing an HRI scenario with independent goals, shared workspace, and no explicit communication. These experiments offer insights into the types and impact of conflicts as well as the attitude of test subjects toward the nearby robot. Next, we introduce a novel autonomy architecture designed to decompose the activities of human intent prediction (HIP) and robot action choice (RAC) in a manner that simplifies decision-making complexity by enabling a full observability assumption and minimizing state-space thus search-space size. The remainder of the thesis studies formulation of HIP and RAC as Markov Decision Processes (MDP) in the context of space robotics simulation case studies. The following paragraphs introduce further specifics of our HIP and RAC formulations.

To predict human intent, a robot must recognize actions of its human companion based on observed physical motions. Presuming full observability of the human’s physical state, but an uncertain model of how observed physical state translates to future state changes, we can construct a Markov chain to describe the evolution of the human’s state. Inclusion of limited state history over a finite horizon within each state can improve prediction of future states given that most goals can only be accomplished by executing a sequence of tasks. Then, rather than human intent being cast as hidden state features in a partially-observable (hidden) Markov model, we instead treat intent as the “actions to optimize” in a Human Intent Prediction (HIP) Markov Decision Process (MDP) designed to generate a human action (intent) policy based on decision-making criteria (models, rewards) that simulate those used by the human. Use of the MDP formulation specifically for HIP helps reduce the state space to a tractable size. Keeping this human intent prediction model separate from the robot action-choice (RAC) part of the decision-making process also allows us to project human intent forward in time through the MDP model, maximizing the Bellman equation over immediate and discounted future reward with the expectation that we will be able to use this HIP information to then optimize robot behavior through the RAC MDP.

1.5 Contributions

The contributions of this work are as follows:

- Initial human subject experiment results show that a safe robotic manipulator arm with limited human-aware planning can operate in a shared physical workspace with a human performing separate tasks without that human suffering a statistically-significant decrease in his or her task completion efficiency.
- The assumption of independent human and robot goals, in a scenario where the robot is directed to not interfere with the human, is exploited in our autonomy architecture to reduce complexity through separation of deliberations associated with HIP versus RAC. This novel problem decomposition reduces computational complexity and enables observability assumptions not possible in an integrated HIP/RAC framework.

- The HIP problem has been structured in such a way that a Markov Decision Process (MDP) can be used instead of a Partially-Observable MDP (POMDP) to determine predicted human intent without unduly increasing the model complexity.
- This research represents the first application of HIP-informed RAC to a space-based application. Indeed, space is a compelling first application because our assumptions are more likely to hold true: astronaut actions are more constrained and scripted than would be expected for humans operating in most Earth-based environments.
- The impact of a specific combination of safety and efficiency terms used in a robot action-choice (RAC) planner is evaluated; we integrate Kulic’s *danger index* [14] as a safety term, and the *incentive for goal completion* and *estimated energy use necessary for action-completion* as efficiency terms. Our work therefore extends Kulic’s work by including danger index in a multi-objective cost / reward function used by RAC that is in turn informed by HIP.

1.6 Innovations

The innovations of this work are as follows:

- Our exploitation of the independent human and robot goal assumption enables a novel decision-making architecture for HRI that is innovative in its decomposition of knowledge, data flow, and complexity management.
- Use of Kulic’s danger index [14] in the safety term within the integrated RAC MDP reward function is innovative in that it enables explicit tradeoffs between safety and efficiency and because it implicitly includes up-to-date information on the human’s intent in RAC while still supporting decoupling between the HIP and RAC MDPs.

1.7 Outline

In Chapter 2, we present background on robot kinematics and dynamics, autonomous planning and scheduling focusing on those developed for space applications, and information relevant to HRI in space applications. We then give a brief introduction to uncertain reasoning using Markov Decision Processes (MDPs). In Chapter 3, we

describe a set of human-robot collaboration experiments designed to enable evaluation of the impact of a robot's presence and motions on human performance, workload, and focus of attention using a series of objective and subjective metrics. A safe robotic manipulator arm is used, and the human-robot productivity is evaluated for a 'dumb' system that only avoids current-action near-term conflicts when given "ideal" intent data. The data obtained from these experiments reinforces our hypothesis that HRI with independent goals is possible and can be safe and efficient. Chapter 4 outlines our system architecture, describing the decomposition of decision-making into HIP and RAC MDPs and their connections to observer and other robot systems. Chapter 5 describes the human intent prediction (HIP) MDP and its application to a space robotics case study. Chapter 6 discusses the robot action-choice (RAC) MDP and presents simulation results from the combined HIP+RAC system, comparing results RAC performed without human state data. Chapter 7 presents conclusions and outlines future work related to further development and deployment of the HIP+RAC architecture.

Chapter 2

Background

2.1 Introduction

The study of robotics encompasses many disciplines. The architecture in this work relies on background in knowledge representation and decision-making under uncertainty, human-robot interaction, and robotic manipulation as a fixed-base manipulator is the platform used for case studies. Sensor technology and algorithms for navigation and feedback control are also critical for an autonomous robot, although not the focus of this work. Robotic manipulators can be mounted on either a fixed-base platform or to a free-base platform such as a roving vehicle or aerial system. In this work, we assume the manipulator is affixed to an in-space structure, similar to how the astronaut would anchor to a fixture for stability.

In our research, we focus on human-robot interaction (HRI) between a human astronaut and an autonomous fixed-base robotic manipulator arm in a space environment. The study of human-robot interaction requires the integration of concepts from computer science (artificial intelligence, multi-layer architectures, symbolic decision making, etc.), physics-based control systems (feedback control, navigation, and sensing), human factors or cognitive engineering, and psychology. Most HRI work presumes shared goals and tasks between all agents with explicit communication used to optimize task assignment and coverage at a high level; the safety of the agents is usually not called into question because the workspaces are not shared. Conversely, our two agents are restricted to implicit communication to minimize distraction for the astronaut and do not share tasks. Because of the safety issues inherent in occupying a shared workspace, we must therefore quantify safety in a manner that can be factored into robot decision-making.

Below, background relevant to this work is presented in the following areas: kinematics, dynamics, and control of a robotic manipulator, autonomy architectures and planning

methods for decision-making, and work relevant to defining and assuring safe physically-proximal HRI.

2.2 Robotic Manipulation

2.2.1 Kinematics and dynamics

To control a manipulator arm, we must know the joint angle and angular velocity of each joint in joint space, from which we can calculate the position and velocity of each joint and the tooltip in three-dimensional space for trajectory-following of an absolute path relative to objects in the environment. We discuss the mathematics of the former below. We do not model a specific tooltip in this work as the localized action of a tooltip does not typically impact actions of a human companion so long as tasks are distinct.

2.2.1.1 Robot Manipulator Kinematics

There are two main conventions for placing the location of reference frames along a manipulator arm and calculating the transformation matrices for kinematics called the Denavit-Hartenberg (D-H) parameter method. The first specification is discussed in Spong and Vidyasagar [22], the second in Craig [1]. We follow the formulation specified by Craig in our simulations and experiments. A kinematic transformation matrix is specified in terms of two frames A and B, where ${}^A P_{BORG}$ locates frame B's origin with respect to frame A and ${}^A R_B$ is a rotation matrix from frame B to frame A. Then

$${}^A P = {}^A R_B P + {}^A P_{BORG} \quad (2-1)$$

can be given in the form

$${}^A P = {}^A T_B P \quad (2-2)$$

where the 4x4 transformation matrix ${}^A T_B$ is given by:

$${}^A T_B = \left[\begin{array}{ccc|c} {}^A R_B & & & {}^A P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array} \right] \quad (2-3)$$

As specified in Craig [1], the transformation matrix between frames attached to the manipulator arm linkages with Denavit-Hartenberg (D-H) parameters $\theta_i, \alpha_{i-1}, d_i, a_{i-1}$ is given by:

$${}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2-4)$$

where

a_{i-1} = the distance from \hat{Z}_{i-1} to \hat{Z}_i measured along \hat{X}_{i-1} ;

α_{i-1} = the angle from \hat{Z}_{i-1} to \hat{Z}_i measured about \hat{X}_{i-1} ;

d_i = the distance from \hat{X}_{i-1} to \hat{X}_i measured along \hat{Z}_i ; and

θ_i = the (variable) angle from \hat{X}_{i-1} to \hat{X}_i measured about \hat{Z}_i .

Since the transformation matrix is given by

$$\begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2-5)$$

$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$ is the rotation matrix and $\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$ is the translation vector.

2.2.1.2 Inverse Kinematics

Consider a closed-form symbolic solution for fully-specified position and orientations (Craig, pp. 113-114, 117-121) [1]. From the forward kinematics transformation matrix shown above, we can find a closed form joint-space inverse kinematics solution by

manipulating transformation equalities via an algebraic or geometric method. Closed-form solutions are easier to implement and faster to compute, and many manipulators are designed to take advantage of this (see pp. 117-125 of Craig [1]). A derivation of the inverse kinematics equations for the Michigan Manipulator Arm (MM-Arm) used for this research is shown in Appendix A.

2.2.1.3 Robot Manipulator Dynamics

For robotic manipulators, a 6x6 Jacobian matrix is used to relate joint velocities to Cartesian velocities of the manipulator arm tooltip:

$${}^0v = {}^0J(\Theta)\dot{\Theta} \quad (2-6)$$

where

$${}^0v = \begin{bmatrix} {}^0v \\ {}^0\omega \end{bmatrix}, \text{ a } 6 \times 1 \text{ vector with}$$

0v the 3x1 linear velocity vector of the tooltip with respect to the base frame

${}^0\omega$ the 3x1 angular velocity vector of the tooltip with respect to the base frame

$$\Theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}, \text{ the } n \times 1 \text{ vector of joint angles of the manipulator (with } n \text{ joints)}$$

Often the Jacobian matrix is partitioned into a translational Jacobian (from joint velocities to linear velocities) and a rotational Jacobian (from joint velocities to angular velocities).

$$\begin{bmatrix} {}^i v \\ {}^i \omega \end{bmatrix} = \begin{bmatrix} {}^i J_{trans} \\ {}^i J_{rot} \end{bmatrix} \dot{\Theta} \quad (2-7)$$

${}^0J_{trans}$ can be calculated by direct differentiation:

Since

$${}^0\mathbf{v}_T = {}^0\dot{\mathbf{p}}_T \quad (2-8)$$

and

$${}^0\mathbf{v}_T = {}^0\mathbf{J}_{trans}\dot{\Theta} \quad (2-9)$$

by taking the partial derivative of

$${}^0\mathbf{p}_T = {}^0\mathbf{T}_5^5\mathbf{p}_T \quad (2-10)$$

we get the Jacobian.

${}^0\mathbf{J}_{rot}$ can be calculated from:

$${}^0\mathbf{J}_{rot} = \begin{bmatrix} {}^0\mathbf{R}(z) & {}^0\mathbf{R}(z) & \cdots & {}^0\mathbf{R}(z) \end{bmatrix} \quad (2-11)$$

where

$${}^A_B\mathbf{R}(z) = {}^A_B\mathbf{R} * \hat{z} = {}^A_B\mathbf{R} * \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (2-12)$$

the third column of the rotation matrix.

Note that one can easily change a Jacobian's frame of reference:

$${}^A\mathbf{J}(\Theta) = \begin{bmatrix} {}^A_B\mathbf{R} & \mathbf{0} \\ \mathbf{0} & {}^A_B\mathbf{R} \end{bmatrix} {}^B\mathbf{J}(\Theta) \quad (2-13)$$

Jacobians in the force domain are related to Jacobians in the velocity domain through:

$$\boldsymbol{\tau} = {}^0\mathbf{J}(\Theta)^T \boldsymbol{\zeta} \quad (2-14)$$

where

$${}^0\mathfrak{S} = \begin{bmatrix} {}^0F \\ {}^0N \end{bmatrix}, \text{ a } 6 \times 1 \text{ vector with}$$

0F the 3×1 (linear) force vector of the tooltip with respect to the base frame

0N the 3×1 (angular) moment vector of the tooltip with respect to the base frame

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix}, \text{ the } n \times 1 \text{ vector of joint torques of the manipulator (with } n \text{ joints)}$$

2.2.1.4 Singularities

The Jacobian describes a linear transformation (mapping) from joint velocity to Cartesian space. If the Jacobian is nonsingular, we may invert it to calculate joint velocities from tooltip Cartesian velocities.

$$\dot{\Theta} = J^{-1}(\Theta)v \quad (2-15)$$

Using this equation, if we had a pre-specified trajectory to follow, we could calculate the necessary joint rates given a certain desired velocity vector at each instant along the path. Those values of Θ for which the Jacobian is not invertible (singular) are called singularities. There are always singularities at the boundary of a manipulator's workspace, but sometimes there are also singularities inside the workspace. When a manipulator is in a singular configuration, it has lost one (or more) degrees of freedom of movement (in Cartesian space) – in other words, there is some direction or subspace (in Cartesian space) in which it is impossible to move the arm. These singularities need to be known and avoided, because loss of a degree of freedom implies loss of control in a certain direction. From a safety standpoint, if we need to replan the arm's trajectory/motion, we will not want to restrict our movements in such a manner. This problem is even more clear in the force domain:

$$\tau = {}^0J(\Theta)^T {}^0\mathfrak{S} \quad (2-16)$$

When the Jacobian loses full rank (which happens at a singularity), there are certain directions in which the tooltip cannot exert static forces. In some cases, this can be seen to be an advantage of the manipulator – the arm can exert large forces with small joint torques to create mechanical advantage. In terms of safety, however, this would be something one would want to avoid. From a practical standpoint, however, an inability of the arm to exert a force in a particular direction also means that if a force is exerted from that direction on the arm (and no joint torque is needed to balance it) then the structure of the arm itself is the only mechanism resisting it.

A Jacobian and singularity analysis for the robotic manipulator arm used in this work is given in Appendix A. For the experimental system utilized in Chapter 3, we issued joint angle reference commands and relied on joint-level servo control loops embedded in COTS (commercial off-the-shelf) servos to follow these trajectories.

2.2.2 Manipulator Trajectory Generation

Given a desired tooltip waypoint or sequence of such waypoints, a trajectory planner must produce sequences of joint motions to achieve each commanded tooltip position and orientation. With no obstacles or singularity issues, these commands can simply represent smooth motions from an initial joint angle to the final joint angle. Given obstacles, such as static objects or a human occupying a shared workspace, the robot must optimize its motion in a manner that is efficient but meets safety (collision-avoidance) constraints.

There are numerous methods for trajectory generation in three-dimensional (3D) space, both Cartesian and joint space, which can provide time-optimal or path-optimal or gross (non-optimal but fast) solutions [23,24,25]. There are grid-based search [26], optimal B-spline [27], and neural network methods [28], as well as gradient descent and artificial potential field methods [29]. Common metrics for manipulator trajectory optimization include time, fuel, energy, path length, distance from objects, velocity and acceleration of tooltip, and force of impact. Computational complexity can be an issue, especially when obstacle avoidance is required.

There are a few methods for implementing object avoidance in real-time; one way to do this in a mostly-static environment is to calculate offline a database of robust trajectories

online, and then choose the trajectory that meets constraints online. We are able to use this method because the set of possible tooltip waypoint goals can be known a priori.

In addition to the traditional cost metrics listed above, for safety purposes we adopt Kulic's danger criterion and danger index [21,14,30]. The first term can provide a safety-oriented cost metric for trajectory optimization or repair (replanning); the latter can be used to reduce the velocity of the trajectory in real-time, as well as a safety constraint used to eliminate cached paths from consideration.

The danger index DI is the product of three terms:

$$DI = f_D \cdot f_V \cdot f_I \quad (2-17)$$

$$\text{a distance factor } f_D(s) = \begin{cases} k_D \left(\frac{1}{s} - \frac{1}{D_{max}} \right)^2 & : s \leq D_{max}, \\ 0 & : s > D_{max} \end{cases}$$

where

$$k_D = \left(\frac{D_{min} D_{max}}{D_{min} - D_{max}} \right)^2 \quad (2-18)$$

$$\text{a velocity factor } f_V(v) = \begin{cases} k_V (v - V_{min})^2 & : v \geq V_{min}, \\ 0 & : v < V_{min} \end{cases}$$

where

$$k_V = \left(\frac{1}{V_{max} - V_{min}} \right)^2 \quad (2-19)$$

$$\text{and an inertia factor } f_I(I_{CP}) = \frac{I_{CP}}{I_{max}}.$$

The terms above are defined as:

D_{min} =minimum allowable distance between human and robot (sets factor to 1),

D_{max} =distance between human and robot at which safety is assured (factor becomes 0),

V_{min} =set to a negative value (factor is zero when robot is moving away from a person),

V_{max} =maximum relative velocity (sets factor to 1),

I_{cp} =effective inertia of the critical point,

I_{max} =maximum safe value of robot inertia,

v =approach velocity (positive when moving towards each other), and

s =distance from critical point to nearest point on person.

2.3 Task Planning & Scheduling

To plan tasks, mission goals are decomposed and sequenced into a series of activities that collectively accomplish goals. Deterministic planning methods can be used in cases where models and the environment are sufficiently static or certain for a pre-set sequence of activities to be applicable, at least over the horizon for which the plan executes. Probabilistic planning methods taking uncertainties in models and the environment into account must be used when we have some information about the likelihood of events and the success of actions but uncertainty in how each action will change world state. Task planners typically rely on search to optimize solutions thus are generally computationally-intensive. Because of their computational complexity, planning cycles are typically executed offline in advance of the system entering the world. Then in real-time, the plans or policies that have been generated are executed. Should events transpire that were not fully handled within the pre-computed plan/policy, techniques such as iterative plan repair [31,32,33] can then be applied to enable the system to adequately function. Repeated occurrence of anomalous events can also prompt machine learning [34] to better enable the system to account for these events in its future planning cycles.

2.3.1 Deterministic planning

Deterministic planning assumes a closed-world (no unexpected events) and is based on search over a set of actions from an initial state to a goal state. State transitions map actions to changes in world state features, and with deterministic models each state is uniquely transformed to another state given a particular action with absolute certainty. Optimal search methods typically select actions based on a function ($g(n)$) describing cost

of traversing from the initial to current node n in the search space, and a heuristic cost-to-go estimate ($h(n)$). Techniques such as A*, uniform cost, and greedy search guide exploration through the search-space.

A number of planning methods have been defined by Artificial Intelligence researchers. Early techniques such as STRIPS use forward or backward chaining to select action sequences that match symbolic goal feature-value pairs, with later extension to partial-order planning (POP) addressing issues such as the Sussman Anomaly that prevent simple chaining algorithms from always yielding optimal results [35]. Techniques such as hierarchical task network (HTN) planning focus on establishing multiple layers of abstraction to decompose planning into a tractable set of local planning instances. HTN can enable an agent or multiple agents to solve relatively complex planning problems efficiently, so long as the domain is specified in a manner conducive to the HTN structure [36].

Some of the above approaches may be extended to non-deterministic spaces by performing conditional planning and execution monitoring. Conditional planning must occur when the outcome of an agent's actions cannot be predicted with certainty, where at every node a set of different possibilities must be expanded and new plans generated depending on the various possible states of the environment [37]. In partially-observable environments, we must also account for a belief state that is computed based only on what we can observe. Execution monitoring is then used to determine when the plan is valid versus when replanning must occur (see also: Russell and Norvig, Chapter 12.4-12.5) [35].

Although nondeterminism can be exploited to model uncertainty, any available information on the probability of certain events or states being reached is lost. As a result, it is impossible to optimize plans based on a sense of expected outcomes, and it is also impossible to prune unlikely but possible states from a search space to manage complexity.

2.3.2 Probabilistic planning

Researchers have developed uncertain planning techniques that take into account both the likelihood of each world state being reached for each choice of action, and the relative reward of reaching each state. Such techniques have been built on Markov Chain models of discrete state evolution, which are extended to Hidden Markov Models (HMMs) when state is only partially observable.

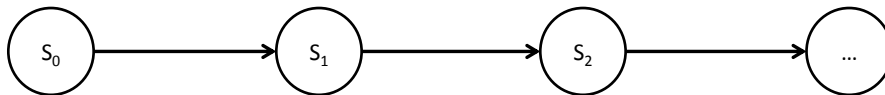


Figure 2-1: Markov Chain Model

Each state in a Markov chain is subject to the Markov assumption: that we only need to know the previous state to know the probabilities of reaching any state $S_j = \{S_0, S_1, S_2, \dots\}$ in the next cycle.

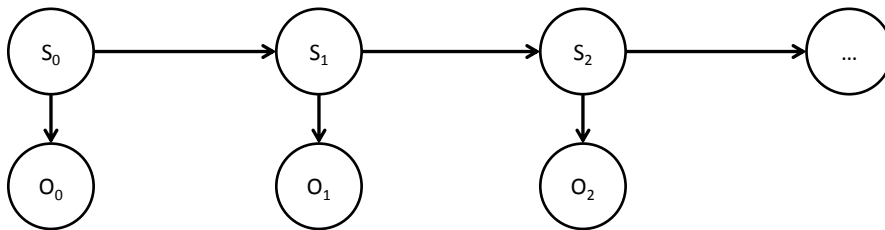


Figure 2-2: Hidden Markov Model

Hidden Markov Models are used when we only have partial observability of the system state. In this case, we use what we observe to determine a belief state – a set of possible states and the probabilities that we are in each state, given the observation O_i . They are an expansion of Markov Chains.

Markov Chains are good for characterizing how a state space evolves over time given some initial state probability vector. For planning, the Markov chain concept must be extended to include the notion of action choice. The Markov Decision Process (MDP) has been developed for this purpose, with the MDP applicable to state-space systems with

full observability while the Partially-Observable MDP (POMDP) applied to systems with hidden state features [38]. The goal of the MDP or POMDP is to specify an optimal policy mapping actions to states. This policy, when executed, will assure that the system executes the best possible action in each observed state.

The MDP, also known as discrete-time stochastic dynamic programming (SDP), can be described as: [39,35]

$$MDP = \{S, A, T(s^i, a_k, s^j), R(s^i, a_k)\} \rightarrow \pi(s^i) \quad (2-20)$$

where S denotes the set of possible states s^i ; A denotes the set of available actions a_k . $R(s^i, a_k)$ is the set of state-dependent rewards for performing action a_k at state s^i , and $T(s^i, a_k, s^j)$ is the set of all transition probabilities $p(s^j | s^i, a_k)$, the probability of a state s^j occurring as an outcome, given an action a_k executed from a state s^i . We assume that the optimal policy is time-invariant, i.e., consistent across all decision epochs.

The transition probability function tensor can be described as:

$$\begin{aligned} p(s^j | s^i, a_k) &= T(s^i, a_k, s^j), s^i \in S, s^j \in S, a_k \in A \\ \text{satisfying } \sum_{j \in \{1, \dots, n_s\}} T(s^i, a_k, s^j) &= 1, \forall i \in \{1, \dots, n_s\}, \forall k \in \{1, \dots, n_a\} \end{aligned} \quad (2-21)$$

representing the probability that the system will transition to a state s^j , when performing an action a_k in a particular state s^i . Optimal policies are typically computed using Gauss-Seidel value iteration or policy iteration over the infinite-horizon Bellman equation (Puterman [39], Chapter 6.2):

$$V(s^i) = \sup_{a_k \in A} \{R(s^i, a_k) + \sum_{s^j \in S} \lambda p(s^j | s^i, a_k) V(s^j)\} \quad (2-22)$$

Policies can also be optimized over a finite horizon or discounted infinite horizon. The MDP as formulated has a reward function but not a cost function, but costs can be represented as negative reward.

Once the optimal policy $\pi(s^i)$ is computed, this policy can be executed. The resulting Markov chain is then annotated by actions applied at each state, which generate associated rewards.

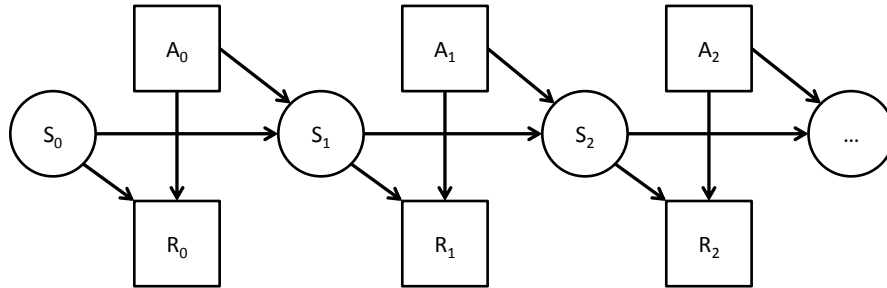


Figure 2-3: MDP Representation

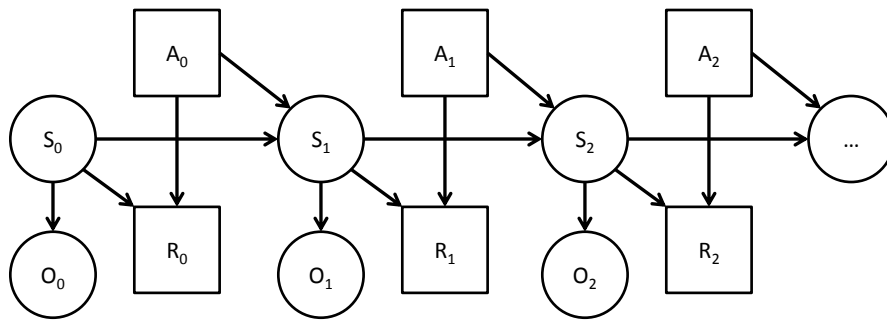


Figure 2-4: POMDP Representation

POMDPs are to MDPs as HMMs are to Markov Chains: they include an observation element as part of the extended representation. This represents added uncertainty – we are not sure of current state s^i , we only have *beliefs* of the likelihood of each s^i given current observation vector o^i . Thus, a state S^i in a POMDP is a vector of all possible states s^i and the beliefs associated with them. The goal of a POMDP is to find the mapping of probability distributions (over states) to actions. The probability distribution vector over all possible states is called the belief state, and the belief space is the probability tensor for the corresponding MDP if we had full observability [40]. Thus, one can approximate a POMDP using a MDP when the belief state is either known and unchanging (thus allowing the uncertainty to be incorporated directly into the transition

probability model) or the certainty in the state estimate remains sufficiently high that any future increase in certainty would have negligible effect.

MDP and POMDP methods are generally optimal, but are computationally intensive both in terms of memory and time requirements. The POMDP is in fact more computationally intensive to the extent that it is impractical for large-scale models. For this reason, the MDP (full observability) is preferred when possible, and policy development is best done offline and potentially on offboard computing resources, particularly when considering the limited capability of space-based computing platforms. For this research, we require uncertain reasoning because human behavior and the environment are uncertain, but developed models are formulated to be as tractable as possible through problem decomposition and formulation in a manner that enables a full observability assumption.

2.4 Space Robotics and Manipulation¹

Most deployed space robotic systems have to-date been human-centric, with space robotic human-support systems either requiring direct human supervision or having been designed to halt when close to impinging on a human's work envelope to prevent injury. There are many examples of these systems. AERCam [41,42] is a free-flying six degree-of-freedom (DOF) spherical camera platform flown on shuttle mission STS-87. AERCam was intended to improve situational awareness for shuttle and extravehicular activity (EVA) missions. The Personal Satellite Assistant (PSA) [43,44], a similar zero-g six DOF free-flyer, was developed and tested at NASA Ames. While AERCam was teleoperated, the PSA had sufficient autonomy to station-keep based on fiducial markings in its environment, but it did not sense or react to human presence (unless the human carried fiducial(s) to track). SPHERES is a set of three zero-g free-flying robots, used inside the ISS cabin; they are teleoperated but can perform stationkeeping maneuvers relative to each other [45,46,5]. The Mars Exploration Rover (MER) [47] pair, Spirit and Opportunity, are remote teleoperated systems designed for planetary surface exploration without humans in their physical environment. Over their highly-successful deployments, MER rovers have seen substantial upgrades to their autonomy software for

¹ This information in this section is from a paper accepted by the AIAA Journal of Aerospace Information Systems (JAIS) to be reproduced under the title "Human Productivity in a Workspace Shared with a Safe Robotic Manipulator" [106]

data compression, navigation, and planning/scheduling. These technologies are transferable to collaborative missions including augmentations in the robotic system's ability to perceive and react to other agents, human or robotic.

Several large-scale space manipulator systems have defined the state of the art for space-based manipulation: Ranger, Canadarm, Canadarm2, and Dextre [8,9,48]. The latter two are currently on the International Space Station (ISS), and have recently been joined by the smaller Robonaut 2 platform. The Ranger and Dextre systems have two highly dexterous 'arms' designed to complete scripted EVA activities while astronauts remain indoors. The Canadarm is a single multipurpose arm, as is Canadarm2; in addition to manipulation without an astronaut, both Canadarms have been used to interact with an astronaut on EVA by use of their end effector as a work platform upon which astronauts can stand and be maneuvered about, decreasing their physical movement effort. Ranger and the Canadarms are teleoperated systems, while Dextre is a supervised system capable of executing scripted automation sequences. Launched in 2008, Dextre completed its long testing cycle in December 2010, and successfully finished its first official repair job in February 2011 [49,50]; as of June 2012, it has completed two rounds of joint operations with NASA's Robotic Refueling Mission (RRM), demonstrating the feasibility of on-orbit robotic satellite servicing and repair [51,52].

There have also been advances in systems meant to physically collaborate with humans. Notably, NASA's Robonaut systems [53,4,54], highly-dexterous human-analogue designs, have been extensively tested on Earth for eventual automated space operations. Robonaut was originally meant to replace astronauts, but now is intended to complement humans on EVA. The first Robonaut (R1) was initially teleoperated, with incremental implementation of capabilities, allowing it to automatically execute task sequences (such as grabbing objects) while operating near humans with minimal workspace impingement [53]. The Robonaut 2 (R2) system was developed in collaboration with General Motors, and one of two prototypes now resides on the ISS; its mission dictates it undergo a year-long testing cycle inside an isolated chamber to study and validate its operational characteristics in a zero-g interior environment prior to use [4]. R2 is equipped with force-sensing capabilities that will stop the robot's motion if it contacts a human (or other

object) or shut it down completely if struck with sufficient force; the arm itself is well-padded in case such a hit occurs [55,56]. This capability is an important first step in enabling safe physical human-robot interaction [18,19]. This, combined with research into intent prediction and smart planning, may contribute to a further increase in the autonomy of such highly-capable systems, allowing for direct close-proximity human-robot collaboration. Our research is complementary to the R2 tests: we focus on human-robot interaction in a “safe lab”, but do not attempt to mimic the zero-gravity constrained ISS environment in the experiments we conduct.

2.5 Human-Robot Interaction (HRI)

Robots are good at performing preprogrammed dull, repetitive tasks for long periods of time with speed, efficiency, and accuracy. From an ethical viewpoint, robots are also more easily replaceable than humans, so we have tended to give them the dirty and dangerous jobs that humans do not want to perform. However, as robotic functionality has increased, we have begun to see the advantages of having robots work in human spaces, and this has led to a discussion of what robots should do, and what humans should do, and who is best at each task. For instance, humans are better at thinking and planning than robots because we are able to learn and intuit knowledge dynamically on the fly, and we are more flexible in our thinking. Thus, most robotics have been controlled by humans to some extent as we have slowly increased their capabilities; it is intuitive that we have had humans dictate to the robots or computer systems what must be done to fill in these gaps of functionality in the interim. More abstractly, robots are designed by humans to help humans, and those humans must decide at some level what tasks they want the robots to be able to perform for them.

There are multiple levels of autonomy for robotic systems being used in HRI: [57]

Teleoperation: A human operator sends low-level motion commands to the robot through devices such as joysticks or haptic devices; in some cases the operator may command scripted motion sequences, these would be low-level commands explicitly sequenced by the human operator.

Supervisory control: direct oversight: A human operator manually inputs traversal waypoints, grasp directives, and action timings and monitors the robot as it computes and follows low-level control sequences to accomplish operator-specified directives. The human operator can interrupt at any time to change or re-prioritize actions.

Supervisory Control: partial oversight: The robot makes some decisions on its own, including decomposing high-level goals into primitive motion sequences and autonomously executing each motion primitive, but will query a human when detecting any unexpected or anomalous situation, or will wait for new directives each time a designated goal has been completed. Human supervisors may interrupt decisions or offer additional input to the system, but such supervision is not a full-time activity.

Full automation: A robot is given all necessary goal, constraint, environmental state, and task information at the beginning of its work cycle, and all decision-making is expected to be done by the robot. If the robot cannot complete a task, it either adapts until it can handle the situation effectively, or it replans to enable continued operation, potentially in pursuit of a different goal.

The person teleoperating or issuing supervisory directives to the robot could be the one sharing their physical environment, or they could be using the robot as a medium through which to interact with its companion. Either or both is considered HRI. Robots are not necessarily restricted to one mode of operation. There is fixed-mode autonomy when it only operates at one level of autonomy during a particular mission; in sliding-mode autonomy, the robot may itself choose to switch between different levels of autonomy in real-time during its mission [57,58]. Transitions between these modes are initiated by the robot and are usually assumed to occur immediately. If a nontrivial transition sequence must occur, then corresponding procedures and analyses must ensure continued stable and safe operation.

As robots and computing power have grown cheaper and ubiquitous, the fields of human-computer interaction (HCI) and human-robot interaction (HRI) have grown. In human-computer interaction, no physical interaction occurs except perhaps minimally at keyboard, mouse, and [touch]screen interfaces. Human-robot interaction, by comparison,

often but not necessarily involves direct physical interaction; HRI also may include varying levels of verbal and nonverbal communication. Human-robot collaboration is similar to HRI, but explicitly deals with cases where the human and robot must interact together to achieve a common goal or goals which usually require physical interaction. Note that this thesis deals strictly with interaction, not collaboration. There has also been increasing demand for robots that can act around humans with little to no human input or oversight. Researchers are beginning to show that inclusion of a ‘smart’ manipulator in a common workspace can result in a decrease in overall human workload and increase in productivity [59,60].

However, we cannot simply insert a fully-autonomous robot and human into the same environment: there are safety and efficiency concerns that must be addressed. The highly-capable robots that might be most productive might also be the most “physically persuasive” because, whether they are faster or stronger or heavier, they have the capability to physically overwhelm humans [60,61]. This ‘natural physical persuasiveness’ becomes a serious problem when robots are working in close proximity to humans; this is the exact reason why industrial robots and humans have needed to have their physical workspaces segregated from each other: these robots move without regard for, or awareness of, human presence and will injure any human sufficiently unlucky to get in their way. Because of the previous research conducted in segregated HRI teams, we have begun to determine when it would be most convenient and efficient to be able to have physical interactions occurring over close-range distances. We no longer want to keep our physical workspace separate and segregated from robots when it is unnecessary.

To address this issue, researchers have developed systems to sense and model human state in real-time, then account for the human in a robot's decision-making process. Examples of recent research into tools supporting safe and efficient autonomy include safe real-time trajectories for physically-proximal operations [62], safety/injury metrics [18,19], industrial collaboration safety standards [63,64,65], the efficient/productive distribution of teams of agents and the breakdown and assignment of tasks (both homogeneous and heterogeneous) [12], scheduling problems (centralized and distributed)

[66], real-time plan repair [13], and implicit or explicit communication between agents. [67,68] As an example of state-of-the-art in HRI, in Ref. [62], experiments were conducted where a robot was directed to find an object in the environment, pick it up, move into a human's workspace, and hand the requested object to that human. The human's position was sensed using an automated laser-based positioning system and real-time trajectories were created by running an A* search to find a path to the handoff state through a potential field 'safety bubble' determined from a cost function.

In Ref. [18] and Ref. [19], the need for effective safety metrics is emphasized, as it is clearly demonstrated that the current widespread use of car-crash safety metrics for these purposes is flawed. An example of this is a lack of warning in a crushing or excessive force application case. In such a scenario, acceleration would be low or negative (deceleration), normally indicating a safer situation, while conversely the manipulator could be pinning a person between itself and an immovable object thus applying excessive force. Industry is incrementally revising their industrial robot safety standards to support limited human movement within the reachable workspace during active operation [63,64,65]. By using a manipulator that can sense its environment and that constrains force and torque application below acceptable limits in experimental testing and real-world use, safety will be promoted by first enabling the robot to avoid a nearby human then second to sense and limit force application should contact with the human actually occur.

HRI safety concerns in a space environment include:

Human safety on EVA: Astronaut suit integrity must be maintained, and the robot must not puncture or otherwise damage the hull of the station. Environmental concerns, such as suit climate and radiation exposure, the stress and fatigue levels of the humans, and maintenance of a direct, unimpeded route to egress for emergency situations are important considerations.

Human safety on IVA: The astronaut will not wear a suit during IVA, so in this case the robot must avoid damaging physical contact, blocking routes between modules for extended periods of time or in case of a need for emergency egress or module isolation;

the robot must also exhibit controlled predictable motions that humans can adapt to or move around.

Physical safety is not the only issue in HRI. The reason the robot is typically in a shared environment is because it can offload tasks that otherwise must be performed by humans. There is therefore a tradeoff between safety and efficiency in operations, in the amount of interference with or endangerment of the human versus the efficiency with which the robot is able to plan and execute its activities. The robot needs sufficient data to plan and react in a manner that is efficient but not intrusive. Without information about the human's goals or intent, robots can only treat human motions as random processes to be avoided, not anticipated.

Efficient human-robot operations have traditionally required communication between agents, enabling agreement upon goals and coordination of actions. Robotic system understanding of a human's goal state is critical in a shared environment due to the safety issues and a need to avoid harmful physical collision and visual occlusion, i.e., the robot blocks the human's ability to see a target of interest. Further, when the human's activities are prioritized over robot activities, the robot must be sufficiently aware of the human's goals and how they translate to actions so it can first ensure it does not interfere with the human, before attempting to achieve its own goals.

There is a nontrivial workload associated with communicating clear, concise, and relevant goal and action information and instructions to a robot, and the situational awareness issues associated with such oversight further decrease the human's productivity level [69,57]. Operator oversight of a human-robot team forces centralized decision-making which can increase efficiency but may lower overall productivity by placing high workload demands on the operator and requiring substantial communication. Further, humans tend to be unhappy when they are given no leeway to make decisions for themselves.

While explicit communication, via speech, gestures, or other methods, requires a mandatory minimum workload overhead for humans [69], implicit communication, accomplished when the robot observes and characterizes physical motion, can provide

cues on human intent without introducing communication overhead for the human companion. Decentralized decision-making for each robot and human and more freedom of choice for the human then becomes possible.

2.5.1 Human Modeling

Researchers have devised multiple methods to model a human. One can start from raw data characterizing physical motion, using time-sequenced video imagery to match a model of gross actions, or audio communication to determine a human’s goal state or intent. Human action recognition – the process of sensing and determining a human’s current task or action from observing their motion – is a difficult but solvable problem. Prior work has shown that this is possible using such methods as template matching and state-space models, the latter of which use a Hidden Markov Model (HMM) to identify gestures in real-time, a process also sometimes called “intention recognition” [70] [71] [72] [73] [74] [75] [76]. A survey of non-invasive visual human sensing methods, from which human intent can be derived, is provided in [77]. Newer surveys on this include real-time techniques for 3D decomposition and reconstruction of the explicit geometric poses and motion using similar methods based on regression [78], and methods action recognition through classification [79], for which many still utilize modified or expanded HMM methods. With speed and computational increases in computing technology, these HMM now may incorporate discretized 3D pose information directly as their input [79]. Other works also discuss prediction of intent from explicit and implicit non-verbal communication without action-mapping [80,81]; these show the feasibility of using “ideal” intent information without regard for source, as is done in our experiments. Such techniques supply not just the current state of the human, but also the known action state-space for the human. There has also been work done into physics-based 3D full-body human motion tracking using Bayesian filtering methods [82].

Such work deals with identifying an in-progress state, not predicting what a human will do next. At a low-level, it is very hard to predict exactly what motions a human may make. It is for this reason that humans in HRI have generally been treated as having random, unpredictable behaviors, especially given uncertainties in human sensing and translation of sensor data to actions/intent. However, because humans generally act as

rational agents, we believe that this knowledge can and should be leveraged and exploited. At a higher cognitive level, humans tend to be more consistent in their decision-making process than at lower levels where trajectory planning tends to be incremented on-the-fly, resulting in far more efficient pose trajectories. The action-state space at a cognitive level also is abstracted from minor trajectory discrepancies to task-level goals thus has significantly reduced variation in possible choices.

2.5.2 Robot Decision-Making with Integrated Human Models

Although there has been a great deal of research done in classifying human action-intent from motion, not much work has been done to use the output of these action-recognition processes to predict the *next* most likely human action. There are two recent works that use Partially-Observable Markov Decision Processes (POMDP's) to predict the ongoing or current action. [Karami, et.al, 2010] computes the current task from observations of the human's actions given no explicit communication, showing results for a case where the human has only two possible mission tasks to choose between [15]. [Karami, et.al, 2009] describes a robot and human with a shared mission without explicit communication and differing goals, and tests the accuracy of using extended MDP's to predict human intent in simulation with two different simulated human policies: random-choice and closest-first. Results were encouraging when the simulated human's policy was not completely random, and help motivate our choice of an MDP for our work [83]. However, neither paper addresses both using human subject experiment data to populate the models and dealing with multiple mission tasks. [Matignon, et.al, 2010] discusses the use of a POMDP by a closely-supervised robotic system in explicit communication with a human to determine the next task it should accomplish; this system uses the POMDP to determine the task that the robot should perform next with strongest-belief of the human's preferences, and identifies when its own model of human preferences does not seem to match the observed human actions such that it must explicitly request new information from the human [84]. [Schmidt-Rohr, et.al, 2008] pursues a similar thrust, with a *filter*POMDP approach: a HMM is used to filter multi-modal explicit communications, which is then fed into the nominal POMDP formulation; the focus is on continued human direction of a supervised robot [85]. Their setup uses human dialog and gestures as observations to help calculate a combined belief state for the hidden,

uncertain human activity state, with the output being the robot's next explicitly-directed action, rather than its acting as a pure human prediction module in the course of the robot making its own decisions.

MDP's can be used to learn from and help deconflict collaborative activities by establishing a common task assignment distribution between human and robot via a shared mental model (SMM); the robot chooses the best action according to the model it is learning as both agents adapt to each other. This SMM approach is applicable when both of the agents are capable of performing all actions in an overlapping workspace and would like to efficiently share the work [86]. There is also recent work on the use of a hierarchy of connected MDP models for a robotic bartending application, where the robot is trained for direct social interaction with multiple humans. This work uses a type of human action-recognition called a Social State Recognizer [87].

Hands-off socially-assistive robotics that interact with humans near but outside their workspace are also relevant to this dissertation. These systems do not perform "helpful work" that completes tasks their human partner would otherwise perform, and they do not pose a collision risk to their human companion. Instead, the physical embodiment (presence and motion) of these verbally- and gesturally-communicative systems is used to motivate their human partner as the human works, to increase the human's productivity. One recent study in this field details a robotic exercise coach that determines when and how to communicate helpful support using a type of human action-recognition called user activity recognition. This system performed robust real-time arm pose recognition and reported results of human subject testing to validate the concept [88].

2.5.3 Human Subject Experiments

Human subject experiments are essential to evaluate HRI concepts and protocols. Per Institutional Review Board (IRB) protocol, experiments must be designed carefully following Design of Experiments (DoE) practices established by the human factors and engineering communities [89,90], such that the data obtained can be used to support or disprove a given hypothesis.

Some examples of system- and agent-level performance measures are given in [61] and [91]. The second reference discusses one procedure useful for obtaining subjective workload assessments called the NASA Task Load Index (TLX). The Task Load Index was developed during a three-year, 40-experiment research effort by the Human Performance Group at NASA Ames Research Center [91]. The Task Load Index has been subjected to numerous validation studies, some of which include supervisory control in laboratory experiments.

The TLX has six load sources: mental, physical, temporal, performance, effort, and frustration. These are measured by a combination of weights and ratings. The weights and ratings are selected by the test subject after each test. Each rating r_i is selected as a tick mark upon a number line between 0 and 100, with gradations of 5 points. Weights W_i for each load source are found by comparing pairs of load sources in unrepeated combinations and tallying the number of times that load source is chosen. This is a maximum across all $n=6$ weights in pairs ($k=2$), of $\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{6!}{2!(4!)} = \frac{6*5}{2} = \frac{3*5}{1} = 15$ points to be distributed across all 6 W_i , with the maximum possible tally for any weight being 5. Weights account for “differences in workload definition between [subjects] within a task and differences in the sources of workload between tasks,” while ratings “reflect the magnitude of that factor in a given task” [91]. Derived data such as the adjusted ratings R_i and the weighted rating R_W defined below can be compared between subjects with less variability than other methods.

$$R_i = W_i * r_i \quad (2-23)$$

$$R_W = \frac{1}{15} \sum_{i=1}^6 R_i \quad (2-24)$$

Chapter 3

Experiments on Human-Robot Operation in a Shared Workspace¹

3.1 Introduction

Through a series of human-robot experiments, recounted in this chapter, we sought to support or disprove the following principle hypothesis, as well as to analyze the validity of the associated assumption.

Hypothesis: Productivity of a collaborating human-robot team operating in a shared workspace can be maximized when the human has no need to supervise the robot. Supervision is no longer necessary when a robot can autonomously operate safely and efficiently with acceptable or no impact on its companion's productivity.

Assumption: We presume human-robot team overall productivity will be higher than the productivity achieved should the human or robot act alone.

Definitions: We define *productivity* as performance when impacted by workload (adverse conditions). *Task performance* is a function of safety (or risk), efficiency (task completion over time), and user preference (task priority); while *efficiency* implies optimal or near-optimal decision-making and quick execution of such by all agents.

To test this, we describe a set of human-robot collaboration experiments designed to enable evaluation of the impact of a robot's presence and motions on human performance, workload, and focus of attention using a series of objective and subjective metrics. During test operations, a seated human test subject was asked to complete simple cognitive and motor tasks, some with the robot idle and others with it moving to accomplish independent goals. We describe the manipulator and computer systems deployed in our tests, and then specify the experiments used to assess our hypothesis.

¹ This information in this chapter has been accepted by the AIAA Journal of Aerospace Information Systems (JAIS) to be reproduced under the title "Human Productivity in a Workspace Shared with a Safe Robotic Manipulator" [106].

Next, the test methodology is explained. Note that to focus our attention on human-robot productivity rather than robot capabilities and limitations, we provide “ideal” intent data to the robot: a human test conductor acts as a “Wizard of Oz,” pressing keys indicating which next task the human test subject appears to be pursuing when that task is initiated. This intent data has a low a level of uncertainty, given the experimental setup and the circumstances of the test. This data enables the robot to predict near-term conflicts and react appropriately to complete its own motion-based task(s), either by avoiding physical contact with the human or line-of-sight occlusion of the human’s gaze [92]. We then discuss our results and conclusions drawn from processing the suite of subjective and objective datasets.

3.2 Test Environment

Our experiments placed a robotic manipulator and seated human in a common physical workspace populated with fixtures that allow the robot and human to accomplish simple but realistic task-level goals. Figure 3-1 and Figure 3-2 provide an overview of the test setup, detailing the three interfaces used in human-subject testing: the robotic manipulator arm, the test subject interface, and the human sensor system (test conductor) interface.

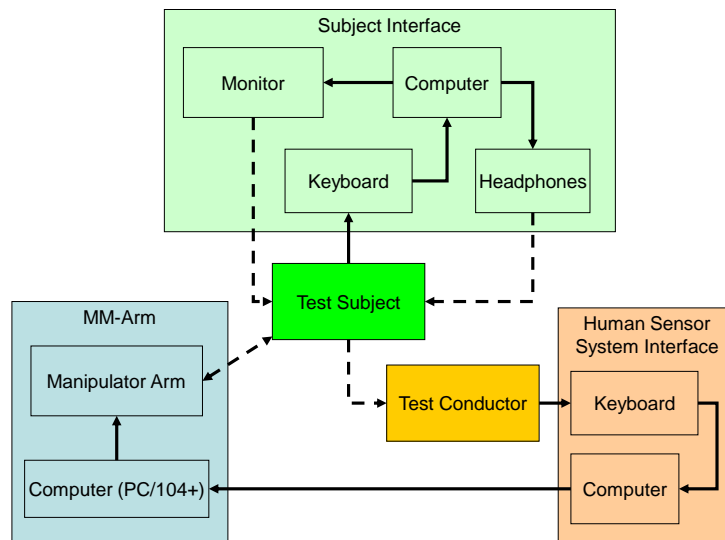


Figure 3-1: Hardware Subsystems for Human-Robot Experiments

Note that the dashed lines in Figure 3-1 show data that has an indirect impact on the receiving block, while solid lines indicate explicit communication between blocks via keystrokes on a keyboard or a direct data connection between components. Also note that while the task activations (“button activations” and “message display information”) in Figure 3-2 were created offline, they are not known to the test subject or robotic agents prior to their activation times.

The computers used by the test subject and test conductor were Dell single-core systems with adequate processing power running a Linux-based operating system. The robotic manipulator used in the experiments is called the MichiganMan(ipulator) arm, or MM-arm, a low-cost platform designed in-house and constructed primarily from commercial off-the-shelf (COTS) hardware [92,93]. The MM-arm is controlled by a PC/104+ computer stack which receives “ideal” sensor data from the “human sensor system” computer. This data is generated in real-time by a human test conductor who indicates changes in the human test subject’s task status via keystrokes, circumventing many challenges in sensing and inference associated with human task recognition that would otherwise complicate the testing. Meanwhile, a test subject interacts with a separate computer interface, nominally completing cognitive (math) tasks and sporadically completing physical tasks prompted by computer-generated messages (e.g., drink soda, eat a chip, press button). Test subjects listened to low-impact music on noise-cancelling headphones to tune out background noise, in particular the manipulator motor noise that introduces additional, possibly distracting, auditory information whenever the manipulator moves between target poses.

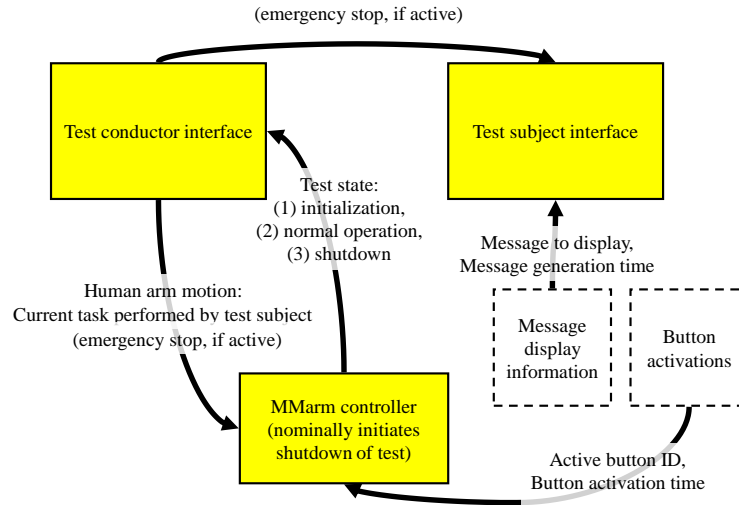


Figure 3-2: Software Infrastructure

Communication between computers occurs through a wireless router. During testing, the data shown in Figure 3-2 were communicated between modules in real-time via basic TCP/IP socket protocols. The data processed post-test included this information, as well as information logging keystrokes and internal processes. All three C++ software modules in Figure 3-2 (one per Figure 3-1 hardware module) time-stamped this data and stored it locally on each computer system in text files; the computers had synchronized system clocks, and data files were transferred to a main server after each test. The MM-arm control software included a communications interface, a task scheduler, and a motion controller. The test conductor interface included a communications interface and a keystroke logger, while the test subject interface included a more advanced visually interactive keylogger and a timed reader-displayer to display task activation messages. Both user interfaces were command-line text interfaces, capturing user keystrokes and reacting as described below. Shell scripts were set up on all three systems to help automate this process – reading a command from the console to determine the test set and test to run, synching the times, running the software modules using the corresponding test script, and copying the collected data to a centralized location once complete.

3.2.1 Test conductor interface

The purpose of the test conductor interface (simulated human sensor system) was to receive keystrokes from the test conductor corresponding to portions of the test subject’s

motion over the course of their task completion progress and send this information to the MM-arm control software for use. The keys used on the test conductor's keyboard were marked with colored paper with the possible test subject actions. Keys indicate human arm and hand motion segments enacted to complete the particular task in progress. For example, the "eat chip" task requires human arm movement outward to the chip bowl, grabbing a chip at the bowl, and return arm movement to bring the chip to the mouth for consumption. Keys for a subtask sequence were chosen in a close grouping together on the keyboard and given the same color-coding. Figure 3-3 shows the key layout used.



Figure 3-3: Test Conductor Interface Keyboard Bindings (number keys = button-pushing actions, yellow keys = drink soda, pink keys = eat chip)

Note that no differentiation was made between motions for a particular task by the MM-arm control software, and mappings from task to conflict were predefined. The test conductor interface software was a simple text-based program run at the console. During testing, the screen listed the key bindings and last selected motion. One key was also set up as an "emergency stop" switch for the MM-arm robot, as a last resort safety precaution. This switch stops all three programs gracefully, pausing the MM-arm before shutting it down and exiting the two interfaces.

3.2.2 Test subject interface

The purpose of the test subject interface was two-fold: (1) to show math problems and task activations to the screen, record keystrokes, and update the screen with partial math solutions or clear messages over the course of each test, and (2) to receive survey data from the test subject after each test. The keys used on the test subject's keyboard were marked with colored paper – the number line, backspace, delete, enter key, and spacebar.

Math problems were solved right to left in a tabular format using the number line (not number pad) and enter key to submit answers; this promoted use of only the right hand during testing. The ~ symbol was used to represent the user's current cursor position onscreen, and the backspace and delete keys removed the number value to the right of this cursor, to allow for the correction of mistypes. At the bottom of the screen below the math problems, predetermined messages instructing completion of an interruptive physical task were displayed on an interval specified by a test script. Test subjects were instructed to press the spacebar once when a task activation message is first registered and again after completing the corresponding physical task. Task activation messages were presented sequentially; in the event that a human-assigned task was not completed before a new task activation, the old task was dropped and the new task replaced the old. The test subject interface software was a simple text-based program run at the console, as shown in Figure 3-4. After each human-robot test series, this interface was replaced with an instance of an open source spreadsheet editor and a survey to be completed.



Figure 3-4: Sample Math Problem Display (with Blue Waypoint Target in Foreground)

3.3 MM-Arm Hardware and Control

Figure 3-5 gives an overview of the physical workspace in which human subject testing was conducted. The manipulator was located in the center of the space (shown at full extension), while the human subject sat in the chair (lower center) with task locations to the left, right, and forward of their station. A monitor for display was situated directly in front of the chair, and inclined lightly above head level. The manipulator and human were situated near each other so they shared the majority of their physical arm and end effector (hand) workspaces. This overlap in collaborative space did not extend into the space containing the human's head, torso, or legs when the human had a seated neutral

body-position, a design feature we conveyed to help test subjects focus on tasks completed in the overlapping workspace. The position of tasks and of objects associated with task completion were chosen such that manipulator arm and human arm movements would come into conflict, but also in such a way that manipulator would only physically conflict within a small envelope of trajectories the human was expected to follow when completing physical tasks.

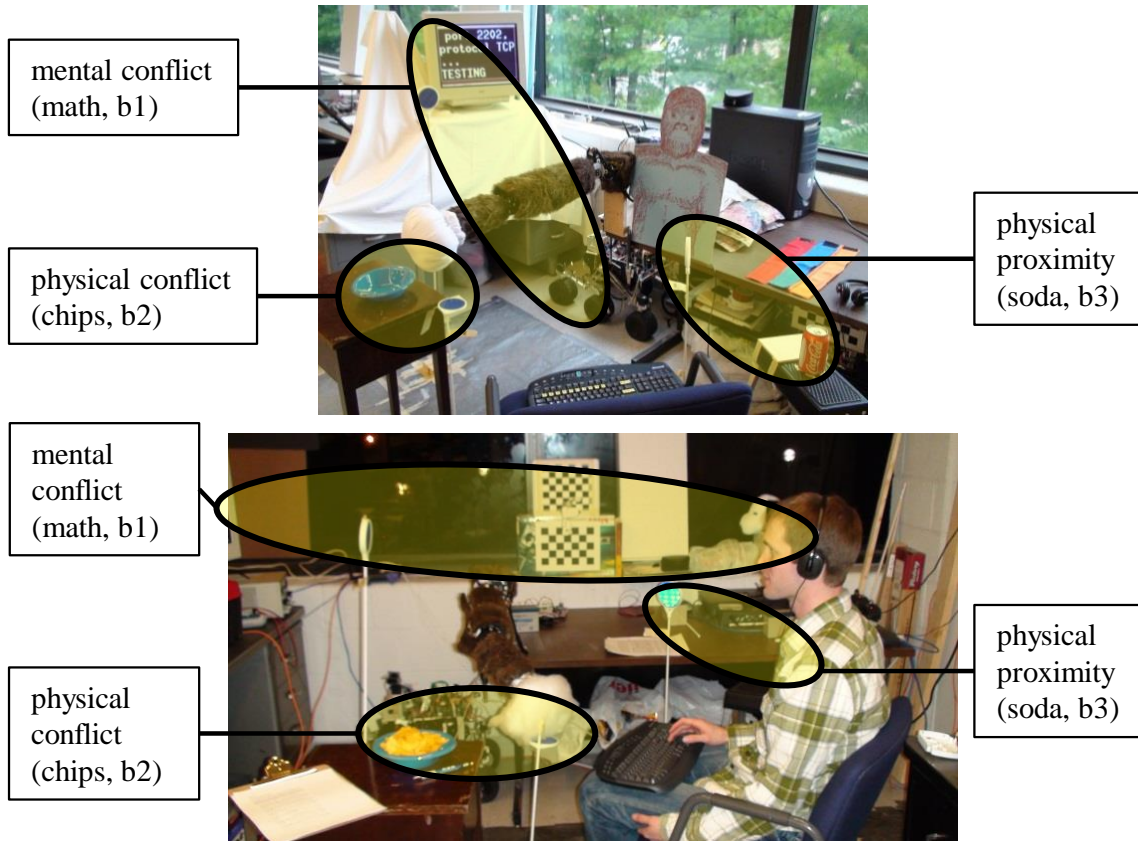


Figure 3-5: Workspace Setup with MM-arm; buttons b1, b2, and b3 are indicated to the Test Subject by Blue Reflectors

The MichiganMan(ipulator) arm (MM-arm) was designed to move in a workspace comparable to that reachable by the arm of a seated human. Emphasis was placed on ensuring MM-arm would be safe for the human subject testing described above. The MM-arm is a low-power, lightweight 4-DOF (degree-of-freedom) R-P-R-P (roll-pitch-roll-pitch) manipulator developed from low-cost components by University of Michigan

students. This fixed-base manipulator has size, speed, and range of motion similar to a human arm and its D-H parameters, which follow Craig’s convention [1], approximately correspond to both a human arm and NASA’s Robonaut 1 (R1) system [92,94]. These are given in Table 3-1. For our implementation, we specify an offset from the final joint axes to a fixed tooltip frame at $d_5 = -11.25$ inches, which extends from the wrist to the center of the soft padded “hand.” The forward kinematics of the manipulator can then be represented by a single transformation matrix. The MM-arm has an analytical solution for its inverse kinematics, due to the joint alignment.

Table 3-1: MM-arm D-H parameters

i	α_{i-1} , deg	a_{i-1} , in.	d_i , in.	θ_i , deg
1	0	0	3.814	θ_1
2	-90	0.345	0	θ_2
3	-90	-0.345	-16.5	θ_3
4	90	0	0	θ_4

MM-arm joints are plastic pan-tilt shoulder and elbow joint mechanisms, while the linkages between the joints are composed of multiple carbon fiber tubes. Cushioning materials of a neutral color are added over the quasi-rigid structures to mitigate any impact force a collision with the structure might impart, increasing the safety of the system. The manipulator wrist is covered with a ball of white cotton batting material to make it visually attentive and to soften the end. Speed and torque of the manipulator actuators are constrained to levels that would not be capable of injuring the seated test subject, particularly given the padding affixed to the manipulator. Commercial off-the-shelf (COTS) digital servos were used as the joint actuators, as they are robust, reliable, and relatively low-torque and low-power compared to typical industrial manipulator motors and motor-driver systems. The particular servos used have internal PD controllers and can be treated as black-box mechanisms with only joint angle inputs. Battery power is fed into a line driver board, preventing runaway and overvoltage power situations. This low-power setup mitigates the fact that there is currently no sensory closed-loop: the MM-arm cannot tell whether it has reached its destination or whether

contact has or may shortly occur. Ref. [95] gives an example of a different research platform with such sensing capabilities.

MM-arm’s closed world model includes all goal location and human conflict information for goal-based decision making. “Closed world” for the MM-arm domain is a valid assumption in our experimental setup due to the further assumptions that all targets of interest remain at known, fixed positions, that the MM-arm has a fixed base, and that the human is seated in a known location with features (e.g., arm length, gaze height) similar to those of a “generic” human model. We explicitly specify a conflict mapping prior to testing. For each human task, we define a set of robot poses that could or would potentially cause a physical or mental (gaze) conflict with the human. Because of the closed-world assumption, we can compute offline the set of trajectory poses or three-dimensional envelope through which the robot will command movements while completing each task. We can compute a similar envelope for the human tasks using a generic human kinematic model, and then compare these to determine the set of conflicting task-pose pairs. While this procedure was sufficient for tests described in this paper, real-world applications where either agent may not have a fixed-base location would likely require online identification of local conflict sets based on target, robot base, and human body movements near and through the shared workspace.

During each test, the MM-arm control software receives the human’s current task, checks the conflict mapping, and selects its current goal. A PC/104+ computer stack, two serial servo controller boards, and a line driver board are used to command and control the manipulator servos at the 9600 maximum baud rate imposed by the servo controller boards. For this work, the robot’s task scheduler is a simple FIFO (first-in first-out) queue with the task sequence for each test predefined in a text file (test script). To reduce risk and annoyance in a shared workspace, the robot was instructed to defer to human activities to (1) avoid physical contact and (2) minimize human gaze obstruction when the human is viewing the computer screen. This simple strategy effectively gives the human tasks priority over robot tasks. In the event of a conflict, the queue blocks as long as the conflict exists. The first task with no conflict is initiated; if no such task exists, the robot moves to a preset “neutral” (e.g., stowed) pose that never conflicts with the

human given our workspace configuration, or maintains its station if already at that pose. This simple task queue is sufficient for our experiments and would be replaced by a planner-scheduler in a deployed platform.

Table 3-2: MM-arm poses

Pose name	Joint angle pose, deg				End-effector location from MM-arm base, in.		
	θ_1	θ_2	θ_3	θ_4	x	y	z
Stowed	90	-90	0	0	0	-27.405	3.469
Unstowed 1	0	-90	0	90	-16.155	0	14.719
Unstowed 2	0	-32	0	90	0.8493	0	23.5856
<i>b1</i>	-55	-53	0	47	-8.154	11.6451	24.6568
<i>b2</i>	-35	10	21	6	4.5867	-3.7261	30.9510
<i>b3</i>	0	45	0	68	22.124	0	11.3295

The MM-arm was designed to emulate human arm motions in part because this has been a convention for human-interactive robots (e.g., for Robonaut) and also because humans are good at predicting how other humans move [96]. While explicit communication is disallowed for the testing, the human may still want to implicitly predict what the MM-arm is doing, even if this prediction is merely to establish a higher level of comfort or confidence in the robot. MM-arm movement is determined by a simple “direct path” joint-space algorithm that plans a smooth joint-space trajectory between forward-kinematic poses. Each robot-assigned task had only one unique “goal pose” associated with it. These poses were found by utilizing a test program to move the MM-arm such that the tooltip location corresponded to the necessary goal location (e.g., button locations), and the pose was selected according to the visual obstruction requirements for that goal pose. Once the final pose was determined, the joint angle set was recorded and used in subsequent testing.

Table 3-2 gives representative poses. Once a task with corresponding goal pose is selected, an arrival time T_f to reach this goal pose from the initial pose is determined (generally the initial time of goal selection T_s plus a preset time-of-motion, e.g., two seconds). To execute this maneuver joint angle commands are incremented as shown in Equations (3-1) and (3-2):

$$\Delta\theta_i = (\theta_{fi} - \theta_{si})/N \quad (3-1)$$

$$\theta_i(t + \Delta t) = \theta_i(t) + \Delta\theta_i, \theta_i(T_s) = \theta_{si}, \theta_i(T_f) = \theta_{fi} \quad (3-2)$$

N for our experiments was 30. A larger N helps reduce the speed of motion by commanding more interim waypoints to be met. This constrains manipulator joints to move at speeds significantly slower than the maximum rates despite the use of COTS servo modules. Such slow speeds promote safety and minimize distraction for the test subject. Within the range of acceptable safe speeds, our choice of N and time-of-motion were made carefully. A perception of too-slow motion would cause irritation and lower productivity, due to increased chance and duration of conflict, while motion perceived too quick would cause undue stress, apprehension, or fear. Either would lead to eventual distrust of the robot’s motions. Feedback from test subjects on manipulator speed is described below. Note that because the arm movements are done in joint space, the speed of tooltip motion is not consistent between poses, but for these tests safety and distraction factors dominate concerns over consistency in tooltip speed. Manipulator and servo constraints restrict the choice of valid angles for each joint. In software we further limit the elbow joint to emulate a “human-like” range of motion. With smooth motions between arm poses, the MM-arm never attempts to move outside the joint limits.

3.4 Human Subject Experiments

3.4.1 Test Methodology

This section describes our test setup and execution procedures. Human subject testing was pre-approved by the University of Michigan’s Institutional Review Board (IRB) in Behavioral Sciences.

3.4.1.1 Test Subject Selection

All test subjects were Aerospace Engineering student volunteers found by posting flyers and via email solicitation. Test incentives were free food before, during, and after test sessions. A nearly even mix of graduate and undergraduate students participated, with one female subject. No subject had appreciable background in robotics or human subject experiments.

3.4.1.2 Test Conductor Responsibilities

The testing process for each subject was supervised by a test conductor. Test conductors were required to take online certification tests created by the Behavioral Sciences Internal Review Board (IRB). Prior to testing, the test conductor educated test subjects on the testing process and fully informed of all procedures before being asked to sign a consent form. The test conductors or “wizards” were also tasked with maintaining a safe, comfortable, and unpressured test environment, as well as test subject confidentiality. They prepared and ran the test equipment and acted as the “ideal observer” for the human sensor system interface. They maintained safety in the test environment and acted as an ideal observer, supplying intent data with a low a level of uncertainty, by performing the following actions during each test: paying attention to the robot and human so that the ‘emergency stop’ could be struck to end the test if necessary, paying attention to the human’s motion and supplying new input to the robot only once the test subject’s next new motion had begun, tracking the action(s) being requested of the test subject on their screen (certainty in the test subject’s action), and correcting any input error to the robot immediately. This last action allowed the error due to mistaken keystrokes to remain negligible due to the robot’s low motion speed: once noticed a new keystroke could be performed and transmitted quickly, well before the manipulator arm could substantially follow through with a response.

3.4.1.3 Testing Procedure

Three test sets of nine tests were developed. Nine test subjects were run through all three test sets in the main round of testing; an additional three subjects completed only the first test set in the initial round. Test activities and timings were the same across subject and managed by the automated test scripts. To avoid cumulative fatigue, each subject completed all three test sets within a two week period, with a maximum of two test sessions per week and one test session per day within ~1.5-2 hours, with each test running 3-4 minutes. Prior to testing, subjects were given a quick ‘demo’ of the robotic arm, and had the opportunity to hold and physically move the manipulator when in a depowered state. This allowed test subjects to become more knowledgeable of the weight, inertia, and padding of the robotic arm. This introduction was designed to allow the subjects to make or better validate their own safety assessment when working near the

robot. Subjects were also shown a ‘demo’ of robotic motion at this time to familiarize them with the robot’s common poses and movements during task completion. Test subjects were told prior to each test whether the robot would be active; this was done to avoid unnecessary “surprise” or stress when the robot started its tasks. Without this information, the subjects might have felt increased levels of agitation or stress if/when the robot did not perform as the subject might naïvely expect, which would have required in-test discovery of this information, potentially resulting in a higher learning curve. Subjects were also told what types of tasks to expect, but nothing regarding task timing or frequency so they could not anticipate or plan task timings or sequences. Surveys were completed immediately post-test.

During testing, the test subject completed as many cognitive tasks as possible but was sporadically directed to complete the higher-priority physical tasks. Messages requesting the user complete a physical task were shown in large print along the bottom of the monitor where cognitive tasks were presented. Once requested, cognitive work could not continue until the physical task was completed, and task initiation (acknowledgement of message) and task completion was logged by the test subject. Test subjects were instructed to complete physical movements at their preferred pace to create a more realistic and relaxed environment. They were told the robot would defer to their movements, and taking longer to complete non-conflicting tasks would allow the arm more time to complete its own tasks, but instructed to focus on their own tasks rather than the manipulator’s motions or task progress. They were also told not to complete “button pressing” tasks for the robot.

3.4.1.4 Data Collection

We collected two main types of data to test our hypothesis: test subject data (in three subsets), and “sensor” data. Objective quantitative data derived from test subject keystrokes (*Type I*), which captured all GUI interface interactions, was post-processed to retrieve the time duration and success or failure of each completed activity as related to human performance for the computation of rates. Subjective qualitative data derived from NASA TLX surveys (*Type II*) captures the test subject’s opinions of how efficient they thought they were, the relative stress they felt from the given workload, and their

comfort level with the robot. An open-ended questionnaire (*Type III*) allowed more direct answers on the test subject's comfort level with the testing and robot interaction. Quantitative sensor data obtained from keystrokes by the test conductor was available to the robot in real-time and tracked the test subject's intended task goal (*Type IV*).

3.4.2 Assumptions and Constraints

To realistically scope experiments and focus results on the examination of efficiency and workload, we make the following assumptions:

- No faults or failures occur in either actuators or sensors (including test conductor data entry). Tests in which any such issues occurred were terminated and not used in our analysis.
- Seated human torso and manipulator base locations are known and constant, and all waypoint targets for human and robot tasks are stationary and predefined. This enables definition of robot trajectories and conflict sets prior to test script creation and testing.
- No appreciable learning curve, no task-switching subject overload, and no variability from choice of handedness existed. We investigate this learning curve assumption as part of our experimental results.
- Test subjects stay on task unless the robot distracts them. The learning curve and task focus assumptions allow us to assume user preference and task priorities are constant.

Tasks with the same level of difficulty require approximately the same execution time to simplify data processing, and each task has common best-case and worst-case execution times. This last assumption allows the use of test scripts developed offline for all test subjects.

3.4.3 Test Matrix

In this section, we define the task sets the human-manipulator team complete and identify appropriate combinations of conflict scenarios in order to test our hypothesis. The test subject is asked to complete three types of tasks: cognitive tasks, multi-step physical tasks, and movement-only physical tasks. The robot can only accomplish (physical) movement-only tasks in the shared workspace. While the necessary tools and objects for

task completion are shared in some cases (e.g., button pressing), for our experiments human and robot tasks are presumed independent, i.e., neither human nor robot can offload tasks from the other. To identify changes in human performance, we perform direct comparisons between complementary pairs of tests – one where the human performs all tasks, and one where tasks the robot can accomplish are offloaded from human to robot. (In these experiments, responsibility for completing movement-only physical tasks is given to only the human or only the robot for any particular test. The human and robot are not given the same type of task in the same test – real-time collaboration in task assignment is beyond the scope of this work.) We treat the cognitive task as “most important” and the physical tasks as desired but either unplanned or unexpected. To identify changes in test subject workload, we chose low-impact tasks with minimal learning curve so the metrics would be more sensitive to changes from the robot’s activity than differences in task sequences. To accommodate this, simple, everyday tasks were chosen that have analogues to tasks completed during on-orbit operations. These tasks were independent, easily completed, and not expected to create cumulative fatigue given the short test duration, making robot activity the sole major impact on productivity.

To identify changes in (human) focus of attention, avoidable collisions or close-proximity motion of the robot near the human are represented by classifying human-robot paired tasks as conflicting vs. nonconflicting. This yields four task classes: i) nonconflicting, robot stationary, ii) nonconflicting, robot moving, iii) visual (gaze) conflict (mental conflict), and iv) physical conflict. The first represents tests in which the robot was not moving to complete any tasks, while the latter three encompass active robot tests. The use of test scripts of tasks and activation times can then ensure the human-manipulator team will encounter the appropriate suite of conflicts.

We first defined specifics of each task, ensuring at least one task of each type was specified. We used mathematical addition as a cognitive task, consumption for the multi-step physical tasks, and “button pressing” for the movement-only physical tasks. These tasks were chosen because they have similar analogues to common tasks in a space environment, such as an EVA for spacecraft repair. The cognitive task requires moderate

concentration and a clear line-of-sight, as might be required when following familiar steps to inspect or diagnose a problem with an electronics module. The physical tasks involving consumables are multi-motion pick-and-place tasks, one for stowing or retrieving consumables such as repair or habitation module supplies. The button-pressing involved ‘quick’ movement to handle an overriding concern, such as grabbing a toolbox that might be able to float away.

For the math problems task, we chose randomized simple addition problems, in a three line $XXX + XX = ?$ format to avoid cognitive delay that would otherwise result from switching math operations. Input was acquired right to left (ones, tens, hundreds, thousands place) from the keyboard number line to nominally occupy both hands and to avoid an ‘expert’ number pad typist biasing statistics. The two consumption tasks were eating chips from a nearby chip bowl and drinking soda from a nearby soda can. “Button pushing” required touching reflectors at static locations $b1$, $b2$, and $b3$, and was simulated as an external interruptive event. Next, we chose test subject task locations and robot poses to satisfy the task classes: non-conflicting stationary as the robot’s default (robot at rest and stowed so it posed no conflict, nc), non-conflicting human-robot task combinations with nearby moving robot (e.g., drinking soda with robot goal to press $b3$, nc), and human-robot tasks with conflict either in the subject’s *visual gaze* given math problems with robot goal to press $b1$ (mental conflict, mc) or in physical workspace overlap eating chips with robot goal to press $b2$ (physical conflict, pc). To best compare the impact of the robot on human performance, we constructed six test scenarios as shown in Figure 3-6. The chosen physical placements allowed multiple robot-active no-conflict comparisons. Robot movement could occur both near and far from the human, or the robot could not move at all. The ‘eat chips during $b2$ activation’ and ‘solve math during $b1$ activation’ tasks were the only activities with conflicts studied in our comparisons. We expected changes in human productivity according to the types of robot task (proximity of the robot) and the speed of the robot, and attempted to choose task activations exploring these factors in testing.

<ul style="list-style-type: none"> Nominal robot action <ul style="list-style-type: none"> Robot action without consideration for human test subject actions (showcases all arm movements to the human, a demonstration case)
<ul style="list-style-type: none"> Ideal <ul style="list-style-type: none"> Nominal human cognition only (math problem-solving with the manipulator at a static workspace position)
<ul style="list-style-type: none"> Human-only <ul style="list-style-type: none"> Human problem-solving with interruption (math tasks with interspersed messages to eat chips, drink soda, press buttons, or some combination thereof)
<ul style="list-style-type: none"> Robot-active, no-conflict <ul style="list-style-type: none"> Collaborative operation baseline with manipulator sporadically avoiding the human (similar to human-only, but with button tasks offloaded to the robot – best case with robot)
<ul style="list-style-type: none"> Robot-active with conflict <ul style="list-style-type: none"> Collaborative operation inducing frequent conflicts (mental conflicts, physical conflicts, or a mix of both)
<ul style="list-style-type: none"> Robot-active with conflict, overtaking <ul style="list-style-type: none"> Collaborative operation “stress tests” (more tasks than can be accomplished, or close to it – worst-case with robot)

Figure 3-6: Test Scenarios

Next, we built a test matrix over a set of test scenarios and conflict combinations. Table 3-3 shows the mix of non/conflicting tasks for the first test session. The robot-only scenario (Z) is introduced before the first test session to familiarize the test subject with the robot’s baseline motions. All test sessions begin with a math-only test (ideal scenario, A) to provide a best-case performance benchmark for each subject and to help check for the existence of a learning curve. Tests then become progressively more complex, introducing consumption and button-pressing tasks into the human’s schedule and moving the robot through a series of progressively more conflicting task sequences. Task timings allow tasks to be completed before reactivation, except in overtaking scenarios. We do not randomize the test order, with different orders of scenarios for different subjects, so that test results could be compared explicitly between subjects. Maintaining a fixed test order allows direct discussion of potential learning curve, task familiarity, and fatigue effects for the different tests.

Table 3-3: Test session 1: distribution of task category combinations

Scenario type	Human tasks			Robot	Conflicts		Overtasking
	Math	Food/drink	Buttons	Buttons	<i>b1</i>	<i>b2</i>	
Z				X			
A	X						
B	X	X					
C	X		X				
D	X	X	X				
E	X			X			
F	X	X		X			
G	X	X		X		X	
H	X	X		X	X		
I	X	X		X	X	X	
J	X	X		X	X	X	X

The second test session consists of paired complementary tests to the first session given in the same time sequence as the first session. This avoids the possibility of unaccountable differences due to fatigue that could result from a differing order of completion. Complementary test scripts are created from a test script by modifying it so that button tasks assigned to the human are reassigned to the robot (leaving the human more time for math problem completion) when the original script is human-only. When the original script is robot-active, button tasks assigned to the robot are reassigned to the human, but only when the human does not already have a task scheduled. Physical tasks that would overlap in time (within an assumed minimum time of completion) are discarded because the human subject is not asked to remember a task queue. We discard overlapping button tasks when creating human-solo complementary tests because this should lighten the workload in the human-solo tests; we want detrimental effects from the robot's presence to be clear. The third test session also included tests across the Table 3-3 scenarios. For this session, the MM-arm moves more quickly when aborting motion toward *b1* in an attempt to minimize screen occlusion (conflict) when the test subject resumes solving cognitive tasks earlier than expected. Comparisons with previous tests allow determination of manipulator speed impact on human performance.

Appendix B contains details on the number of interactions and conflicts that occurred in each test in every test set.

3.5 Test Metrics

In our human subject experiments, we track changes in human performance and workload to support or disprove our hypothesis following the standard Design of Experiments (DoE) practices established by the human factors and engineering communities [89,90]. We computed our objective performance measures from test subject keystroke data. First, we captured *task completion time*, the time elapsed between subject acknowledgement and completion of a task. We also evaluated cognitive task set *completeness*, *correctness*, and *incorrectness rates*, computed from the number of cognitive tasks (completed, correctly completed, or incorrectly completed) divided by the amount of time not spent on the physical tasks. These statistics provide measures of a test subject's ability to focus on the cognitive (math) tasks. The use of rates allows comparisons of tests across different test subjects regardless of test duration, number of task activations, or the comparative difficulty of the physical tasks.

We also investigated subjective performance measures. To establish metrics for subjective survey data acquired during our tests, we adopted the NASA Task Load Index (TLX), a procedure for obtaining subjective workload assessments developed during a three-year, 40-experiment research effort by the Human Performance Group at NASA Ames Research Center [91]. The Task Load Index has been subjected to validation studies, some of which included supervisory control and laboratory tasks. The adjusted ratings R_i and the weighted rating R_W metrics are defined as discussed in Chapter 2.5.3.

During data processing, we utilize the adjusted ratings for performance, effort, and frustration, as well as overall workload R_W . Performance provides an evaluation of a test subject's view of their own performance. Effort and frustration capture workload changes directly related to the robot's involvement. Given its definition in the TLX instructions, effort should be the most sensitive load source to any learning curve. Overall workload is divorced from test and test subject specifics and is used for cross-subject comparisons, similarly for the objective rates.

To put the TLX load scale ratings in perspective, adjusted ratings R_i have a range of 0 through 500 and the weighted rating R_w has a range of 0 through 100. We define “noise” as a change in the weighted rating of a load source by up to ~10-20, since a change in rating or weight by one gradation has a proportional effect on the adjusted weight. We describe how specific weights and rankings were determined in our results section below. Higher ratings indicate the test subject felt higher workload or pressure from that particular source, except for performance rating where lower ratings indicate a feeling of poor performance during that particular test. Higher weights indicate the test subject felt those load sources were more important contributors for that test than those with lower weights.

To identify changes in human performance, we perform direct comparisons between complementary pairs of tests and the robot’s impact, positive or negative, is inferred from differences in the task completeness and correctness rate and the human’s perception of their own performance between paired tests. To identify changes in test subject workload, we infer trends through: (1) task-specific workload, comparing task completion times and task incorrectness rates between task types, and (2) test-specific workload, from the TLX survey data [91]. The TLX load sources demonstrating workload come from cognitive task completion (performance rating) and robot presence (frustration rating), as well as overall workload R_w . To identify changes in human focus of attention, we infer the robot’s impact on level of distraction from differences in performance and workload between the human-solo and robot-active tests.

The ideal outcome of our experiments would be increased performance, decreased workload, and increased focus of attention on human tasks when the robot is active. This or neutral results in all cases would support our hypothesis and make a strong case for improvement of the robot’s collaborative processes. Major sources of distraction could result from the visible motion of the robot or hidden stresses from the lack of explicit human-robot communication such as fear of collision. Fear of collision would suggest lack of trust in the robot, where the human would be more comfortable or efficient with supervisory control or a non-overlapping workspace. Unavoidably, some distraction is expected due to hard-wired biological reflexes when confronted with motion in the field

of view, but a lack of consistency in task completion times may indicate that the human was diverting more attention to the robot than attributable to uncontrollable impulse. Distraction could decrease with further increased familiarity with the robot. However, since an attempt at this type of mitigation was included as the first test in session 1 (scenario Z, Table 3-3), a further decrease would not be expected.

3.6 Results

This section describes objective and subjective results compiled for our nine test subjects over the metrics and test scenarios outlined above. As described above, human productivity was determined from performance and workload comparisons between paired complementary tests and between human-only versus robot-active tests. Objective rate data was used for all twelve test subjects to assess any learning curve effects and evaluate human performance. However, data from the three test subjects who only completed test set 1 was removed from the learning curve and paired complementary curve comparisons. Subjective TLX data was collected in digital form for nine test subjects to provide further evidence for existence or absence of a learning curve as well as workload. We utilized eight of the nine test subjects' data. We excluded Subject 9's TLX data, as his ratings were anomalous. For Subject 9, the data saturated the low end of the recordable TLX rating scale and showed minimal, if any, variation in the ratings. Overall TLX workload scores for Subject 9 were outliers at the low end compared to all other subject TLX scores. Because we examined differences in ratings, this data was generally unhelpful for subject-specific analysis. Note that exclusion of Subject 9 data did not give the data a hypothesis-friendly bias; inclusion of the Subject 9 data would in fact have helped bias support towards our primary hypothesis.

We also discarded incomplete TLX survey data on a test-by-test basis. Comparisons between the two types of data identified possible correlations between workload and objective performance. For subjective metrics we define 'no significant change' as data occurring within one standard deviation. We plot this as simple lines for single-test comparisons and box plots for aggregated tests, with the box centered on the average and spanning standard deviation. For objective metrics we define 'no significant change' as confidence within noise of the median value(s). We plot this using bowtie plots, with the

lower quartile, median, and upper quartile values as horizontal lines; vertical whiskers extend to the extrema while +’s at the outer edges symbolize outliers, and the ‘notches’ – sloping inward lines of the bowtie – show the span of negligible noise. No overlap between ‘notches’ implies that the data have different medians with 95% confidence (a.k.a. difference at the 5% significance level, similar to the T-test for means). We discuss statistical outliers in the context of our hypothesis. Below, we first examine learning curve effects. Next, we perform paired complementary test and task comparisons. Finally, we study task completion time and subject-reaction comparison results.

3.6.1 Learning Curve

The TLX data gave mixed results in subject perception regarding learning curve. For the first test set, the order in which tests were run was: test Z, solo tests A-D in that order, and the rest included robot assistance (see Table 3-3 for “scenario type” identifiers). Test A was the math-only baseline, B was consumption-only, C was button-pressing only, and D included all human task types. Expected results according to the mix of task type and relative difficulty level are given in Table 3-4; test letter indicates what metric value listed on that row should be plugged in for that test. Figure 3-7 gives an example of a case exhibiting no learning curve effect.

Table 3-4: Expected relationship of test result data in test set 1

Learning curve				
Metric being compared	No	Yes (distinct)	Only math	Only consumables
TLX rating (subjective)	$\min(B,C) < D$ or $D < \max(B,C)$	$D < \min(B,C)$		
Correctness rate (objective)	$(\min(B,C) < D$ or $D < \max(B,C))$ and $\max(B,C) < A$	$A < \min(B,C)$ and $\max(B,C) < D$	$A < \min(B,C)$ and $(\min(B,C) < D$ or $D < \max(B,C))$	$\max(B,C) < \min(D,A)$

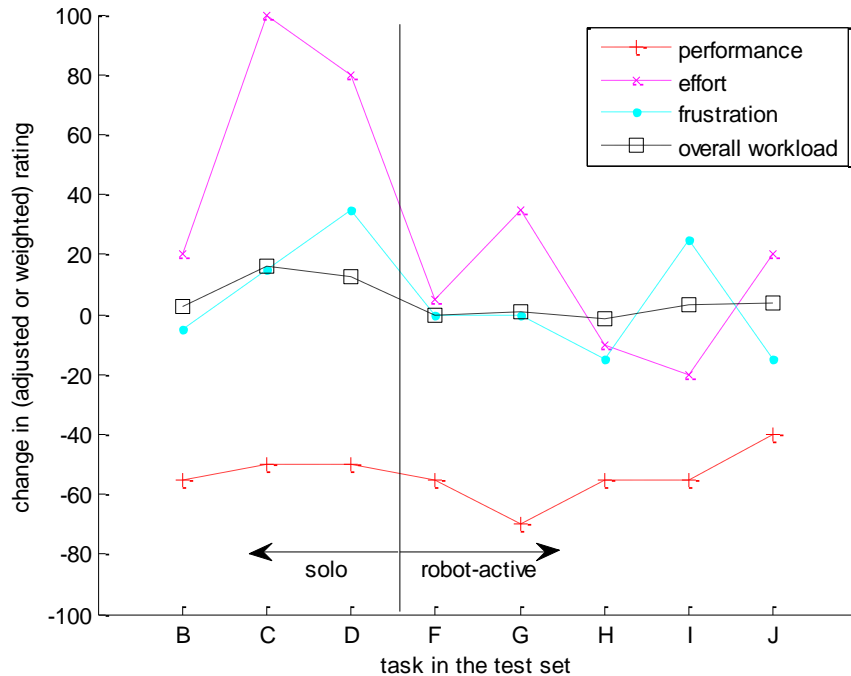


Figure 3-7: Selected TLX Load Source Ratings Relative to Baseline: Subject 5, Test Set 1

From the test set 1 data, aggregating across all subjects, the standard deviation is $\pm 20 R_i$ or R_w , and the noise level is $\sim 0.04-0.05$ problems/sec (about 2-3 problems per minute). The specifics of single-subject comparisons to these values for no significant change are listed in Table 3-5; on a row, numbers indicate subjects for which that type or lack of learning curve held true, - indicate subjects for which it was unindicative of a learning curve, and * represent subjects excluded from comparison due to anomaly or lack of data. We include a symbol ^ denoting a not-statistically-significant learning curve (a trend seen with noise level of 0) for completeness. The TLX data shows no perceived (subjective) learning curve for overall workload for any subject and a first-day learning curve for two test subjects in effort rating. The correctness rate data shows that no subject exhibited a statistically-significant learning curve. Three subjects exhibit no learning curve trend at all, and three subjects exhibited a distinct but not-statistically-significant learning curve on the first day. So, 1/4 of the subjects showed a clear trend, while three-quarters of the subjects showed a trend of some sort. There is a clear discrepancy between the TLX data and the objective data.

Table 3-5: Exhibited statistically-significant learning curves: test set 1 comparison

Learning curve	Subject number											
None – overall workload	1	2	3	4	5	6	7	8	*	*	*	*
None – correctness rate	1	2	3	4	5	6	7	8	9	10	11	12
Distinct – effort rating	1	2	–	–	–	–	–	–	*	*	*	*
Distinct – correctness rate	–	–	^	^	–	–	^	–	–	–	–	–
Any – correctness rate	^	–	^	^	^	–	^	^	^	^	–	^
Only consumables – correctness rate	^	–	–	–	^	–	–	^	^	^	–	^
Only math – correctness rate	–	–	–	–	–	–	–	–	–	–	–	–

a * denotes subjects that were excluded from the comparison, while a dash denotes a negative conclusion.

a ^ denotes a not-statistically-significant learning curve (noise level of 0); a number denotes significance

We also check for learning effects across test sets; this is indicated by a marked decline in the baseline cases across all three days that the test sets were performed. However, the correctness rate data shows no significant decline between the baselines, and neither do the TLX results shown in Figure 3-8. Both the objective and TLX data thus indicate no learning curve across different days.

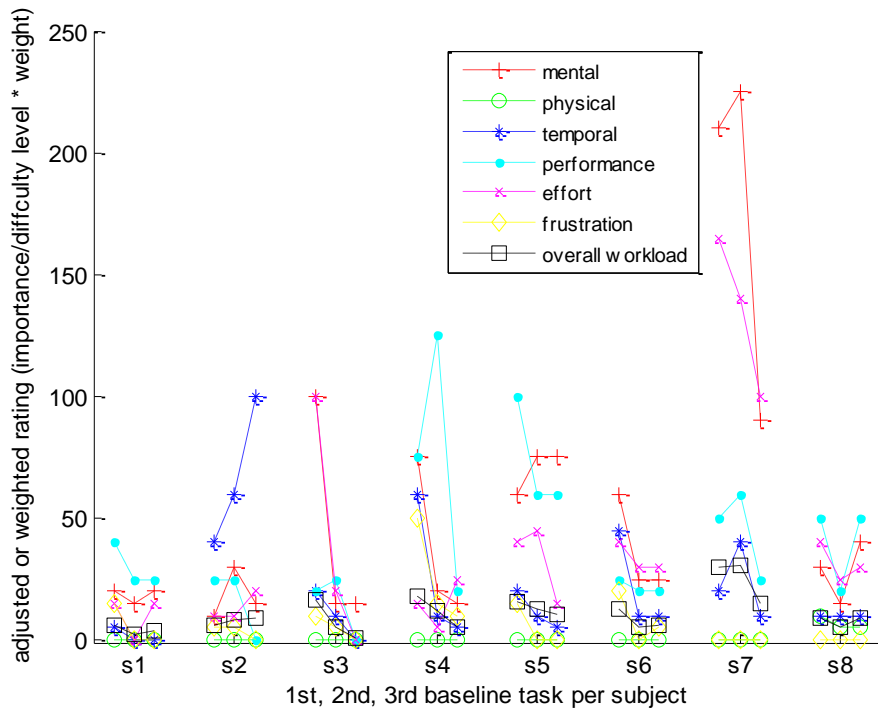


Figure 3-8: TLX Load Source Ratings for Baseline Cases Over All Test Subjects

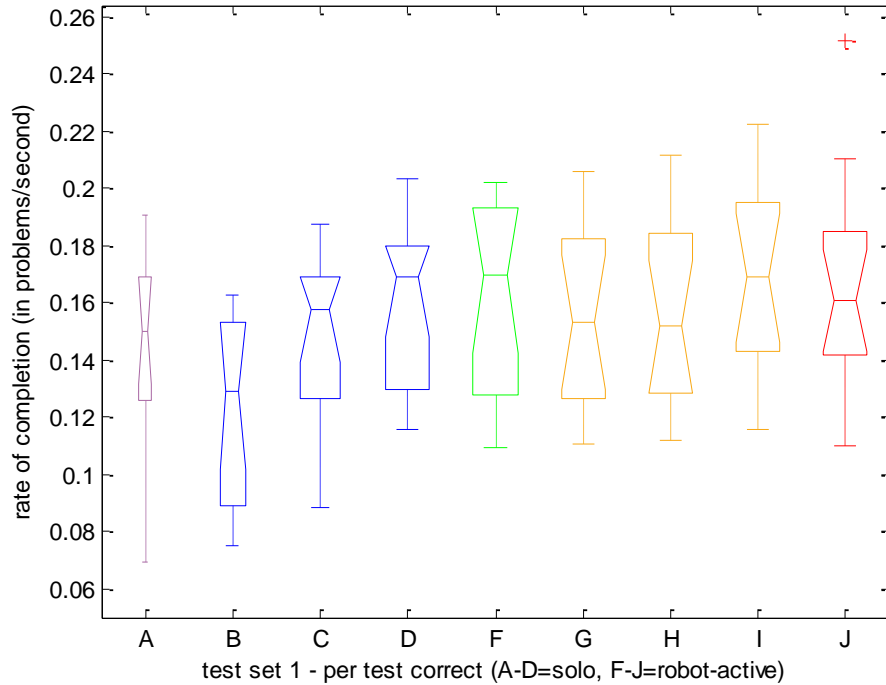


Figure 3-9: Correctness Rate for Math Problems: Across All Subjects, Test Set 1

Figure 3-9 shows the aggregated correctness rate data for each test type on the first-day across all subjects. Differences are within noise for tests B-D, indicating no general first-day learning curve for test subjects. We also verified there is no significant change across the multiple test days (not shown). Thus, no general learning curve from both the objective and subjective datasets is seen across multiple days, nor does the objective rate data show a trend to indicate the presence of learning curve on the first day. As the TLX data straightforwardly showed no learning curve for individuals when comparing overall workload (the main metric), there is also no general learning curve from the TLX results. There are individual subjects who appeared to believe they experienced a first-day learning curve based on TLX effort rating (which could bias the results of the robot-active cases to seem more efficient); a slight learning curve effect might also be discerned in the rate data. However, rate differences were not appreciable, inconsistent between individuals, and did not impact our conclusions, so we do not remove initial tests from consideration.

3.6.2 Paired Complementary Test Comparisons – Robot-as-Subordinate

To determine objective performance, we compare the correctness rates of paired complementary tests both singly and as aggregates across all subjects. This provides a comparison of human performance when the human must accomplish all tasks versus when the robot helps by completing button-pushing tasks for the human. Note that an increase in correctness rate represents performance *improvement*. A significant change is noted when correctness rates for single-subject solo vs. robot-active cases are outside noise. An increase in correctness rate indicates the change in rate was outside error and higher in the robot case than the solo case; a decrease in correctness rate indicates the change in rate was outside the error range and lower in the robot case than the solo case. Based on aggregate rates, some of which are shown in Figure 3-9, the solo vs. robot-active tests are within statistical noise in every case, with an average noise range of about ± 0.02 math problems/sec. This implies no significant difference in the subjects' objective workload between when there is and is not visible robot motion in the human's field-of-view.

Table 3-6: Trends for correctness rates: objective data

Scenario type (robot case)	Number of significant increase	Number within noise	Number of significant decrease
E	4	5	0
F	3	6	0
G	0	5	4
H(1)	2	7	0
H(2)	1	7	1
I	0	6	3
J	4	5	0

E-F = robot-active no-conflict, G-I = robot-active with conflict, J = overtasking.

Table 3-6 shows a distribution of the number of test subjects who showed significant change in objective performance between their solo and robot-active paired tests. In this direct comparison, the no conflict (*nc*) cases (E,F) and one mental conflict (*mc*) case (H1) show clear improvement in correctness rate when the robot takes over tasks, while clear performance degradation only occurred for physical conflict (*pc*) cases (G,I). There are

mixed results for the second *mc* case (H2). These results suggest mental conflicts disrupt the subjects less than physical conflicts, while no conflict cases do not disrupt. However, this could be an artifact of the particular task since the mental math problems did not require the subjects to watch the screen at all times, but rather encouraged many quick bursts of attention over the course of each problem. The overtasking case (J) also displays clear improvement relative to conflict cases without overtasking. This illustrates the existence of a tradeoff between overtasking and inclusion of the robot since the human was overtasked but not distracted by the robot in scenario *J*. This objective data provides evidence that a human does not have increased workload when there is no conflict or only mental conflict with an active robot, but that a human may have increased workload when there are physical conflicts with the robot. The worst-case degradation is from a *pc* test case, and resulted in a decrease of 0.07 correct/second. With average correctness rate of 0.15-0.2 correct/second, this is a drop of up to half the average rate, or 12-20 problems over a 3-4 minute test. This result suggests we want to avoid close physical operations when possible, and identify and mitigate all conflicts as soon as possible when close physical proximity of robot and human is required.

Table 3-7 shows a distribution of the number of test subjects who showed significant change in *subjective* overall workload between solo and robot-active paired tests. Note that an increase in workload implies the robot had a *negative* impact on performance. The TLX workload data has standard deviation of ~10 for perceived overall workload, which is within the 10-20 of the TLX ratings scale noise discussed previously. Overall workload remains the same or decreases with the robot active, as does the effort subscale rating; none are higher than standard deviation. There are no clear indications from the TLX data (increases in workload) that correspond to the degradation (decreases in rate data) seen in the objective rates data. This dichotomy suggests human subjects are more forgiving and “feel” less stressed by the robot than the objective measure of their work output heralds.

Table 3-7: Trends for overall workload: TLX data

Scenario type (robot case)	Number of significant increase	Number within standard deviation	Number of significant decrease
E	0	6	3
F	0	8	1
G	0	8	1
H(1)	0	8	1
H(2)	0	7	2
I	0	7	2
J	0	3	6

E-F = robot-active no-conflict, G-I = robot-active with conflict, J = overtaking.

3.6.3 Task Category Comparisons

3.6.3.1 Solo versus Robot-Active Cases

In this section we compare two test groupings – those where the human works alone (human-solo), and those where the robot is moving to press buttons (robot-active). A significant increase in test subject workload in the robot-active cases provides evidence that the presence of the robot was distracting the human, while no increase or a decrease would suggest the robot does not distract. Figure 3-10 shows adjusted ratings per-person across all tests within a test grouping, excluding overtaking cases. For each subject, for each load source, and for all cases, the change in TLX ratings was either within standard deviation or showed significant decrease. This indicates that the inclusion of the robot's involvement did not change subjective performance or workload significantly. Those cases in which a clear decrease occurred were unsurprising, given that in the robot-active cases less physical tasks were assigned to the human and the mental task of solving math problems is of lower impact than other tasks.

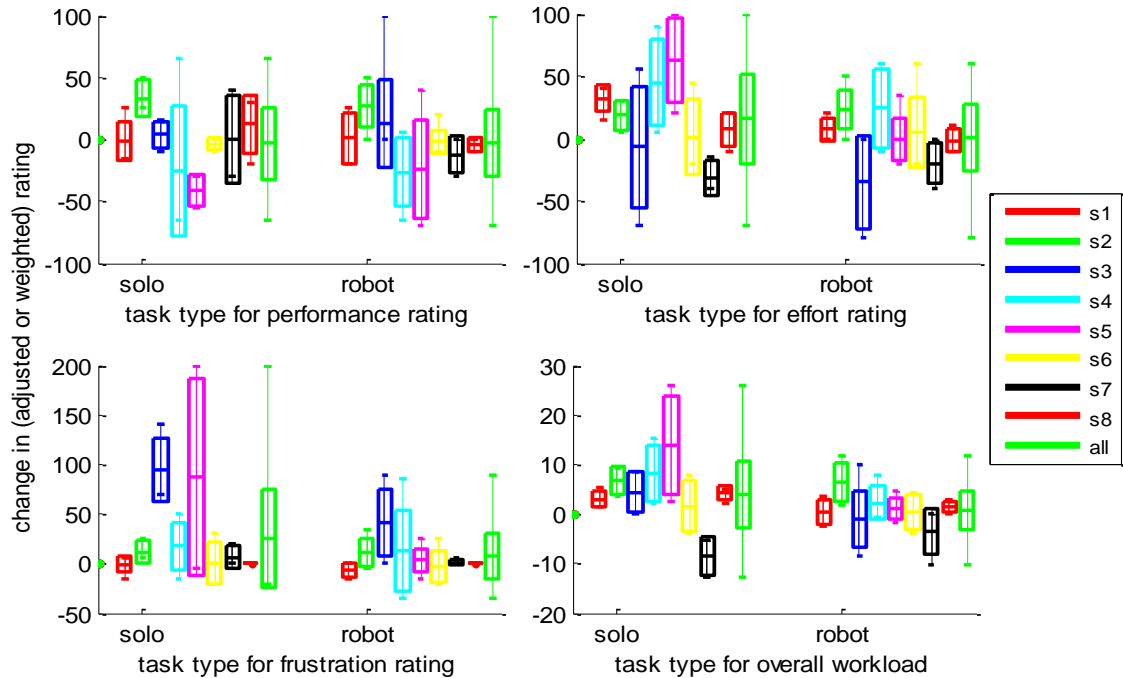


Figure 3-10: Selected TLX Load Source Ratings Relative to Baseline by Task Type, Subject

It should be noted that in almost all cases shown in Figure 3-10, the maximum change in adjusted rating is not higher than 150, and the change in overall workload is not higher than 25. This implies that worst-case there is a 30% change in adjusted workload and 25% in overall workload between non-overtasking cases. We also compute the objective incorrectness rate, providing an indication of math mistakes expected to be more frequent when a test subject is distracted by the robot. This data (not shown) is within statistical noise when comparing solo and robot-active cases per test set, per subject, except for subject 7 where the incorrectness rate was significantly lower than noise for the solo tests on test day 1. This is the only time this happened so it was likely not a true improvement and this subject exhibited learning curve effects on test day 1. Our data therefore indicates that significant increase in test subject’s objective workload when the robot was active.

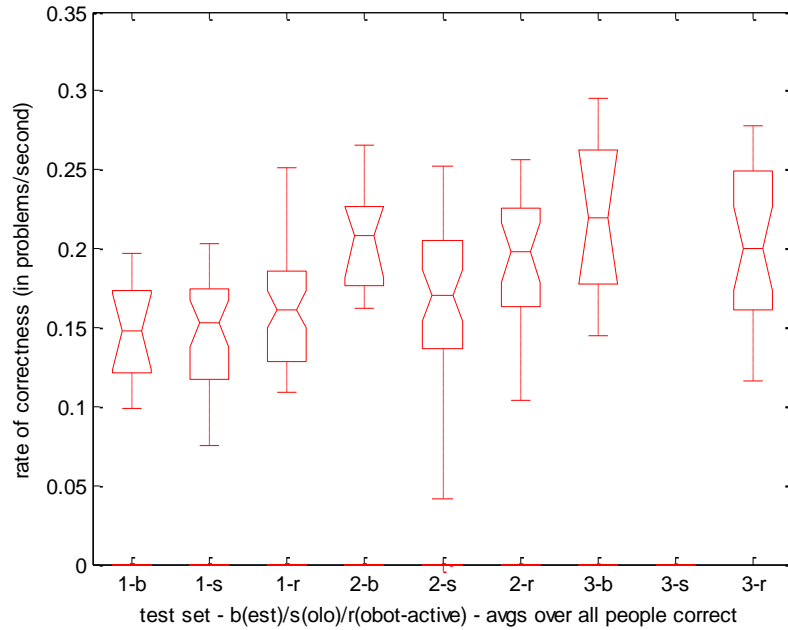


Figure 3-11: Comparing Correctness Rates between Test Groupings Across All Subjects

Next we look for more subtle changes in correctness rates. The aggregate correctness rates in Figure 3-11 show that within each test set, the solo vs. robot-active tests were within statistical noise, with an average error range of about ± 0.02 math problems/sec. Thus, the rates show no appreciable difference from each other. Across all tests, per person, the correctness rates are all consistent in average. Differences are within noise when comparing the solo and robot-active cases, except for subjects 3 and 7 who exhibit improved (higher than noise) correctness rates for the robot-active tests than the solo tests on only the first test day – also probably due to learning curve, for reasons similar to above, and thus negligible. This data also indicates that there are no major differences in subjective or objective workload with robot activity.

3.6.3.2 Robot-Active Conflict Cases

While there is negligible distraction effect seen between the solo and robot-active cases, there is some variation within the datasets themselves. As discussed above and shown in Table 3-3, there are several types of robot-active cases. For completeness, we also looked for trends in workload for the different conflict subcases— no conflict (*nc*),

physical conflict only (*pc*), mental conflict only (*mc*), and both physical and mental conflicts (*pc&mc*). Note that the intersection of any of these subcases is empty.

We next compared individual TLX ratings over the robot-active conflict cases. Frustration rating was high for all subjects in the *mc* cases. Verbal and written survey results indicated the subjects wanted the robot to move away faster in these cases. The subjective performance rating for *pc&mc* was much higher than for other subcases: dealing with more conflicts while still completing all tasks may have given a greater sense of accomplishment than less challenging tests. Objectively, we see that, per person, the range of correctness rates across tests is 0.02 correct/s within a test set (and 0.05 correct/s across all tests) with the error about half that. This is a noticeable but not dramatic change since this only translates to a maximum difference of 4-6 problems over a 3-4 minute test. Also, trends indicated no particular conflict type was better or worse, with overall performance consistent across conflict types in a per-person comparison. Type of conflict therefore does not appear to play a significant role in task completion or correctness, although subjects can be more frustrated by some conflicts than others.

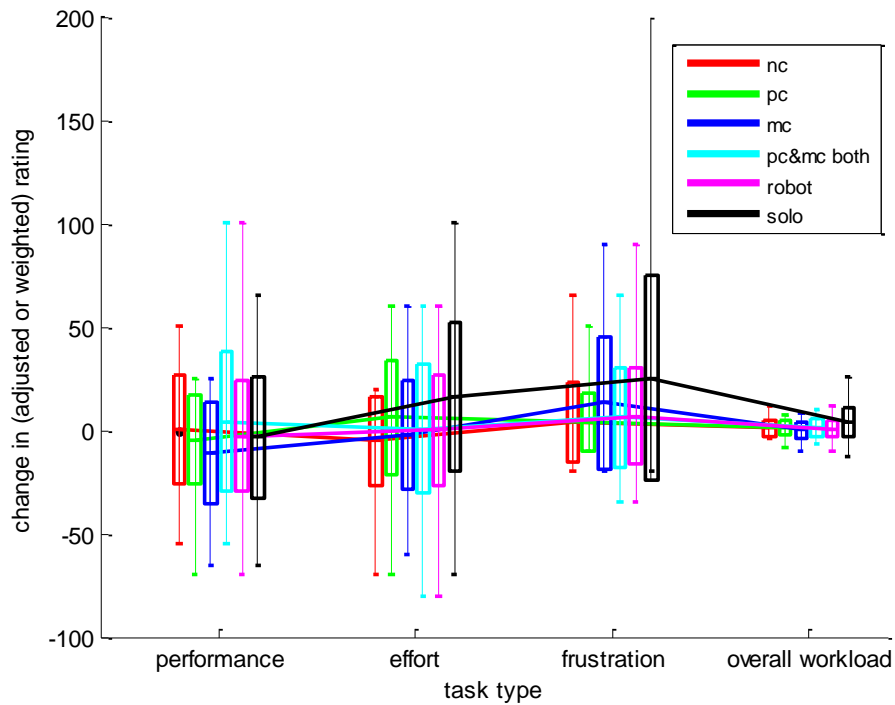


Figure 3-12: Selected TLX Load Source Ratings Relative to Baseline Across All Subjects and Tests by Task Type (no overtasking cases)

In Figure 3-12, we show aggregate results across all test subjects and test types. It is apparent that regardless of the robot's involvement, differences within-rating appear to fall within standard deviation. An interesting caveat is that the solo test-type cases have a larger standard deviation for effort and frustration and double the standard deviation in overall workload than do the robot-active cases. This echoes the *Type III* data: many subjects responded that they preferred the robot to complete the "annoying" button-pushing tasks. Such comments may be linked to the physical distance to the task location – the test subject had to lean forward to reach button *b1* – and nearby tasks reportedly caused less to no stress, according to *Type III* data. Future work could explore this issue in more detail by varying button locations between tests, allowing mixed-button-pushing (collaborative task assignments) between human and robot, and utilizing directed surveys querying subjects about this specific preference. An alternative explanation is that the test subject realized that the robot could not complete two of the four nearby tasks (eating/drinking) thus considered all tasks within this near physical space to be candidates for more efficient completion by the human test subject. In summary, both the subjective and objective datasets indicate that test subject workload increases with differing types of robot involvement, which supports our initial hypothesis.

3.6.4 Task Completion Times

If the offset of standard deviation for average completion time of a type of task does not overlap sufficiently between different tasks, then the workloads of the tasks can be considered different. Figure 3-13 shows aggregate task completion times.

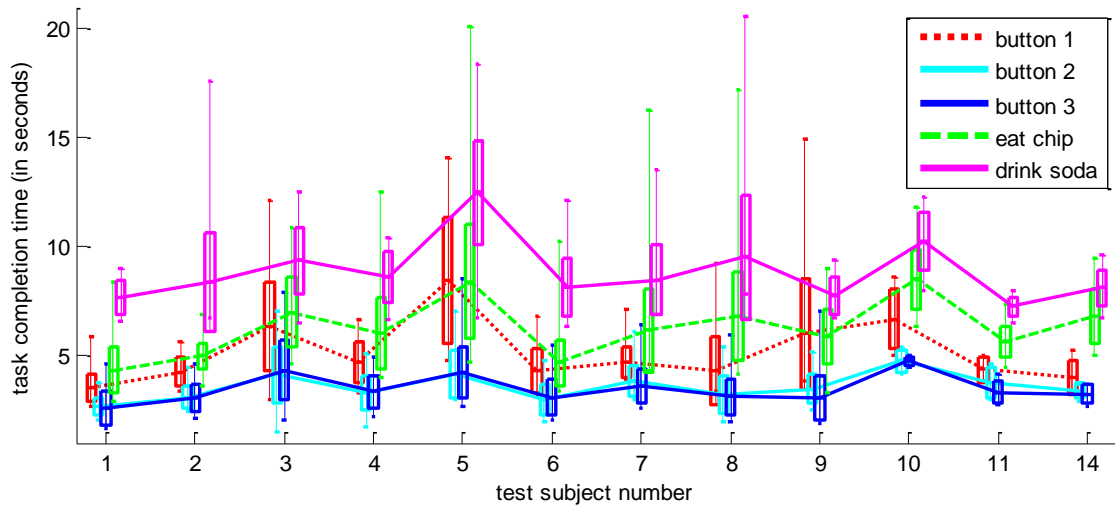


Figure 3-13: Task Completion Times, Across-All-Tests per Subject

From the figure, button *b2* and *b3* tasks have similar workload to each other, but less than *b1* which was most physically distant from the test subjects. Chip eating tasks by different subjects had a consistent standard deviation of ~1-2 seconds, but the average time of completion differs substantially between subjects. Soda drinking tasks have less variation per-subject and took 3-5 seconds more time to complete than the chip tasks; however, the standard deviations are so large for food/drink consumption tasks that this is not a statistically significant result. Individually, some subjects were internally consistent (low standard deviation) for the consumption tasks; however, those who were inconsistent followed no pattern to link with a distraction. From observation, the chip and soda tasks have more timing variation in portions of the movement done at the task locations (grasping the chip/can and the actual eating/drinking) rather than in physical extension and retraction arm movements. Although this might initially appear to be motivation to change the eating/drinking tasks in future work, the larger variation in delays observed at worksites would be expected in real-world tasks where the human or robot would typically grasp, complete a physical or inspection activity, or release an object at each site.

3.6.5 Subject Reaction

Test subjects provided notable feedback regarding robot speed and conflicts. Many test subjects indicated the speed of the manipulator was too slow, particularly in conflict cases. However, many of these same test subjects said that the manipulator was moving too fast when the motions were sped up slightly in test set 3, though the faster motions occurred infrequently. The speedup was from a 3 second traversal to a 1.5 second traversal, and could only occur up to two times per relevant test for movement away from button 1 if the task was interrupted. This data suggests the existence of a crossover point at which the manipulator's speed is perceived differently by the test subjects. Qualitatively, this crossover point likely exists between these two speeds for most subjects. Unfortunately, we cannot draw a strong conclusion at this time, as we did not include an explicit survey question to formally collect this information from every subject in this round of testing; this could be future work. However, despite this verbal reaction regarding manipulator speed, it is notable that when asked directly, subjects said that they felt safe working near the manipulator at all times. A dichotomy of behavior was observed during the *mc* cases with *bl*: some subjects would attempt to look over the manipulator to continue their work, while others either waited patiently for the arm to move away or slowed their movements during other tasks to give the manipulator time to finish. This behavior suggests subjects were unclear on the proper protocol for this conflict, suggesting additional guidance in how to react to conflict (wait vs. attempt to circumvent problem) in future testing. In this case, the uncontrolled factor was the tradeoff between the relative importance of completing the designated cognitive tasks in the timeliest manner versus conserving energy by patiently waiting.

In one interesting case not included in our statistics, a physical conflict occurred but the sensor signal was not sent by the test operator. The test subject quickly noticed the robot continued the conflicting task, paused to wait for the robot to finish, then completed their own task while casually commenting verbally upon the lack of response. This single data point suggests the human test subject could recognize and handle differences in the robot's behavior between cases in which it reacts to the human and cases when it does not (but remains a safe companion). This is important once the robot is responsible for sensing the human and predicting intent, as the goal would be for the robot to correctly

predict human response in most cases; however, it will be difficult to prove the robot's predictions will be accurate in all cases.

3.7 Preliminary Conclusions

We have proposed a hypothesis that a collaborating human-robot team operating in a shared workspace can enjoy maximum productivity when the human need not supervise the robot. To support or refute this hypothesis through human subject tests, we constructed and utilized a human-robot experimental apparatus that placed a seated human and fixed-base robotic manipulator in a shared physical workspace. We characterized test subject performance given a series of experiments in which a human completes cognitive and physical tasks with and without a manipulator executing its own conflicting or non-conflicting tasks. The goal of these experiments was to determine the extent to which a robot manipulator impacts human task performance and workload when operating in a shared physical workspace. Our results, both objective and subjective, support our hypothesis: productivity of the human in the shared workspace remains comparable to productivity of the human working alone so long as the robot does not interfere directly with the human's physical motions or perceptual focus of attention.

Key results from the above data processing section are summarized below:

- No first day learning curve is evident from overall workload. There are mixed results regarding first day learning curve in objective performance (correctness rate); no statistically-significant learning curve was evident from the object data, but there were some noticeable trends. No multi-day learning curve is evident from either objective or subjective datasets.
- The paired complementary test objective rate data indicates that test subjects are unaffected by robot presence except when physical conflicts occur. The subjective TLX data, however, shows no clear increase in workload under the tested conditions.
- In solo versus robot-active test aggregate comparisons, data indicates robot involvement did not significantly change subjective or objective performance or workload.

- Differing types of robot involvement show no general trend that test subject workload increases in either subjective or objective datasets.
- All tasks had task completion times with standard deviation on the order of their task completion times.

From our hypothesis, experiments, and data processing efforts, we draw the following conclusions:

- Our hypothesis and assumptions were supported by our results, with the exception of physical conflicts where the robot was not able to perform efficiently.
- Fast real-time response by the robot is essential to avoid human productivity decrease in the case of physical conflict.
- Cases were observed where the robot needs to begin moving away from a possible conflict site before the human physically moves toward completing this conflicting action due to upper (safety-constrained) bounds on the robot's movement speed.
- Test subjects consistently felt safe with their robotic companion, with some subjects requesting higher manipulator speeds to minimize conflicts/delays.
- Subjects could easily interpret whether the robot was continuing to pursue its goals versus executing a conflict avoidance action. The robot was not yet equipped to predict human intent beyond keyboard inputs provided by the test director.

The physical conflict scenarios suggest that intent prediction, potentially achieved with look-ahead or model-predictive planning to avoid the need for explicit communication of intent, could increase efficiency and minimize frustration when physical conflicts would otherwise arise. Intent prediction might redirect the robot to other objectives, or might simply require the robot to expedite or delay its current task sequence. Scheduling algorithms with probabilistic prediction will be key to meeting safety constraints while also maximizing collaborative performance. Test subject confidence in such a fully-automated system will be important to evaluate in future tests.

Future work could also include longer-duration testing to determine the impact of human fatigue on ongoing human performance and overall system performance. If the robotic system is able to offload dull and repetitive tasks from the human without significantly impacting human productivity levels, we anticipate the human-robot team will be more productive than the human alone. Longer-duration tests can enable a study of the tradeoffs between varying human-robot task allocation and task intensity versus work session duration on cumulative fatigue. There is also a question of when or whether a tradeoff occurs in level of distraction due to robot proximity for the different types of tasks. This was tested somewhat, but not exhaustively, in this work; testing for these purposes would involve changing the physical placement of the task in the workspace (eating near versus eating far) as well as the proximity of robot motion. Direct comparisons of near-conflicts versus far-conflicts for physical and mental tasks were also not tested in this work (only near-physical and far-mental tasks were performed); testing with fewer or no mixtures of task types would be interesting to pursue. It will also be interesting to conduct a longer series of tests over several weeks or months, seeking individuals with a background in robotics, or offering less-knowledgeable participants a more thorough understanding of the robotic test platform prior to testing. Tests could then examine the effects of extended prior knowledge and/or newly learned knowledge versus multi-test experience on human productivity. Productivity comparisons broken down by gender and age might also be interesting, if a study with enough individuals for a statistically-significant comparison can be conducted.

Chapter 4

System Architecture with Feedback for Human-Robot Interaction

4.1 Introduction

In this chapter, we present a three-tier autonomy architecture [97] that enables a robot to incorporate human observations and models in a manner that best supports safe, efficient, and autonomous (unsupervised) robot decision-making in a shared environment under uncertainty. We aim to increase the robot's efficiency while still safely avoiding conflict with a human companion by using the human's current sensed status to also predict future human intent, and then supply this information to the robot's planner so it can better optimize its choices when working in close proximity to that sensed human.

Algorithms and models were chosen to accommodate shared workspace human-robot interaction (HRI) when there are distinct task sets for each agent. We assume the human's current state is fully-observable. Autonomy architectures have typically been cast within a three-tier (3T) framework for robotics [97,98,99] with separate layers for planning/scheduling, plan execution, and low-level device or feedback control. In this work, we focus on the planning and execution layers, although we illustrate interfaces to a lower-level observer and controller typically found in the lowest architectural layer. With respect to the 3T architecture concept, we incorporate symbolic task planning as the highest layer of decision-making. This layer allows the robot to develop plans or policies that incorporate observed human actions into robot decisions, with resulting plans then capable of reacting accordingly. Robot (manipulator) trajectory tracking and state estimation through an observer populate the lowest layer of 3-T and are presumed in this work to always function correctly. This strict separation of task-level planning and plan execution from physics-based control and sensor data processing (state estimation) allows us to explicitly focus our attention on higher-level decision making and to carefully define the dataflow between modules in a manner consistent with state-of-the-

practice. Exploration of interfaces and shared information impact on system performance is a key to successful autonomous behavior in an HRI context.

While most planning layers in a 3T structure are modeled as integrated planning systems, abstraction and decomposition are common mechanisms to accurately and efficiently model the domain of interest. As discussed in Chapter 2, most researchers use methods such as Hidden Markov Models (HMMs) to characterize human intent. If HMMs were incorporated into a robot decision-making framework, a Partially-Observable Markov Decision Process (POMDP) would be required to then solve for the robot's state changes and optimal actions based on the human's state. However, POMDPs are widely recognized as being impractical for decision problems of even modest complexity. Fortunately, as discussed at the end of Chapter 2.3.2, so long as the accuracy of the input (characterized human intent) is higher than a user-specified threshold, indicating the MDP can closely approximate the POMDP with uncertain belief state, the human action can be considered observed thus "recognized". The choice to use the MDP rather than POMDP significantly reduces computational complexity, which is important given the suite of HRI activities that might be required and given the potential need for space-based computing resources to be used to build optimal policies. One of our key architectural contributions, as discussed below, is to then further simplify the problem space by separating the decision-making related to human intent prediction (HIP) from the decision-making process associated with robot action choice (RAC). This novel architectural choice decomposes the problem and will be shown to enable full observability of each MDP state-space. The use of (predicted) human intent as feedback for robot decision-making, both for operation in a shared workspace and for space applications, is novel. These HIP and RAC processes, which are the main focus areas of Chapters 5 and 6, are discussed with respect to our autonomy architectural framework below.

4.2 Motivating Example

Throughout the rest of this work, we assume that this prediction of a human's motion in a physically shared workspace takes place in a space environment. We therefore present a case study for an astronaut on IVA inside a pressurized spacecraft to help ground

discussion of our architecture as well as subsequent more detailed discussion of decision processes. In our IVA example, the astronaut is tasked with computer work while keeping an eye on a nearby experiment that may require some upkeep. He/she is sitting in front of a computer console with food and drink nearby, snacking intermittently while working. This environment lends itself well to human intent prediction (HIP), as the motion prediction element complexity is greatly reduced due to the physical constraints imposed by the seating device that secures the human in place in the zero gravity environment. Meanwhile, the robot is conducting maintenance requiring traversal between multiple worksites in the shared workspace. For the robot to work without disturbing the human, it must predict situations in which the human will need to access different parts of the workspace as well as view without occlusion parts of the habitat (e.g., a computer display showing data associated with the ongoing experiment).

The demands imposed by a spacesuit on on-orbit EVA are far more physically restrictive than the shirt-sleeve garments worn on IVA. However, we can still assume that tasks are somewhat scripted, and that mobility will be purposely restricted to enable the astronaut on EVA to move and apply forces and torques efficiently. Due to the suit, IVA tasks generally require less time and energy expenditure to complete than would a similar task on EVA. Further, risk is increased in EVA because the suit and anchoring mechanisms provide less protection and security than does the space habitation module. This suggests that IVA HRI with an autonomous robot is more likely to be explored near-term than EVA HRI, although both could be beneficial to a mission. The laboratory experimental setting in Chapter 3 was intentionally chosen to be an analogue of the IVA setting described here, to allow us to familiarize ourselves with the scenario and capture knowledge of similar motion-primitives to those expected in our IVA scenario.

4.3 System Architecture

Our architecture depicted as a 3T structure is given in Figure 4-1. The mission operator supplies the mission goals and tasks to both the astronaut and to the robot's planning process, including environmental information, task sequences to accomplish each goal, and goal priorities and assignments. This is done prior to the start of any given HRI episode. The mission operator can update the planner with more scenarios so that the

robot can create additional plans for later use, or send commands to the executor directly (not shown) if he/she wishes to circumvent the planner and take direct control of the robot's actions. In our research, the role of the mission operator is not explored as we devote our attention to decision-making and execution processes found within the planning and executor layers.

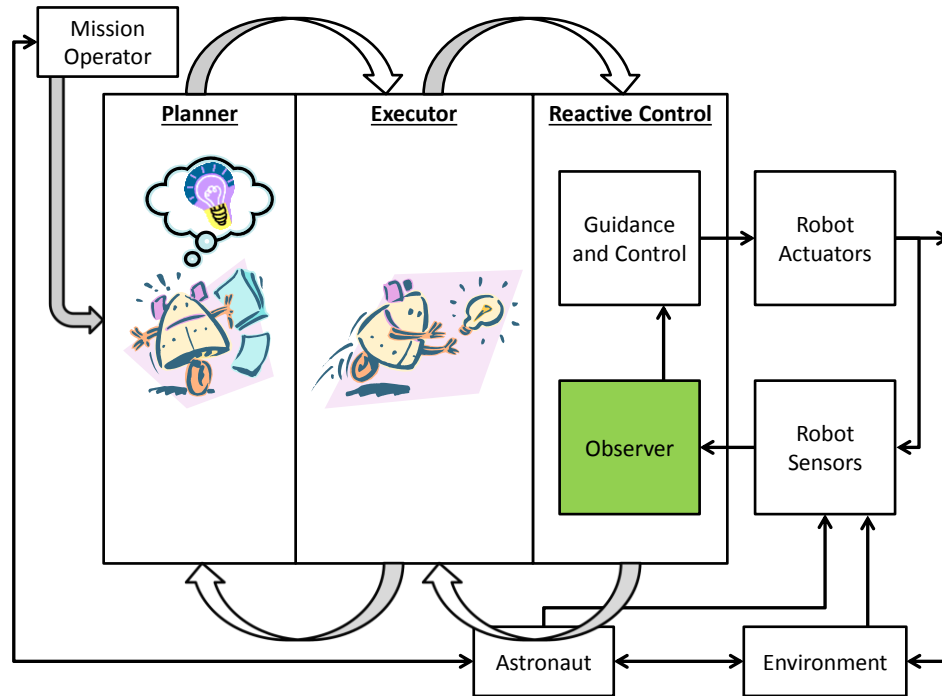


Figure 4-1: General 3T Architecture for Space HRI with Feedback, System-Level

The inclusion of sensor updates as feedback into the system lets higher decision-making and executor layers reconfigure – thus react – to observations in real-time. The observer module in the reactive control layer supplies the majority of the feedback used by the executor and planner, including the results of the robot's own actions as well as the current state of the human and the environment. The guidance and control module outputs actuator commands to follow the trajectories supplied by the executor, but the module can react immediately to unexpected actions sensed by the observer and modify its commands for safe physical distancing using Kulic's danger index, as will be discussed in later chapters [14]. The guidance and control module supplements the observer with trajectory error information, both for the robot and for the human, to help

the higher automation layers react appropriately. The executor chooses the optimal action for the robot in real-time according to the activated plan it has previously received from the planning layer, driven by the feedback from the observer includes knowledge of itself, the human, and the environment.

The planning layer supplies the plans for the executor to follow. However, because symbolic planning tends to be computationally complex, replanning at this higher level should occur as infrequently as possible. Ideally, all plans or policies would be created offline then executed online, although in anomalous situations such a model may not be sufficient to support fully-autonomous contingency operation. The planning layer is told which scenario is taking place by the mission operator before the start of each interaction and, in our work, supplies a single optimal policy to the executor that includes appropriate robot actions for all possible outcomes (states) expected in our closed-world system for that particular scenario. In future work, a feedback loop could be added between the observer and planner, then scenario-recognition logic could direct the planner to either replan or activate a cached plan when an unexpected event renders the executing policy as no longer valid. Such an extension would support autonomous policy switching and replanning under most circumstances.

Because we assume separate tasks and goals for the human and robot in our problem space, a natural decomposition is to separate this problem into two parts: one dealing with inferring human tasks and goals thus intent, and one for selecting robot tasks and goals. Robot action choice requires knowledge of its own goals, the environment, and of the human's current and future intent to enable avoidance of physical conflict. Because we also assume that the human will not react to the robot if the robot acts in a manner that does not interfere, the human's goal driven intent is assumed not to change due to the robot's actions; thus, while the robot must account for the human in its decision-making, we assume the human decision-making process need not account for the robot. This simplifies the problem in that, while feedforward from human intent prediction (HIP) to robot action choice (RAC) is essential, feedback from RAC to HIP is not needed.

Thus, we formulate our planning system as two separate MDPs: a human intent prediction (HIP) MDP that models the human and predicts their next action(s), and a

robot action-choice (RAC) MDP that determines the robot’s optimal policy action-choice. The resulting policies are separate but linked by the output of current and future predicted intent of the human that is output by the HIP policy and used by the RAC policy, as shown by the blue arrow in Figure 4-2. The green boxes indicate the areas of focus in this thesis.

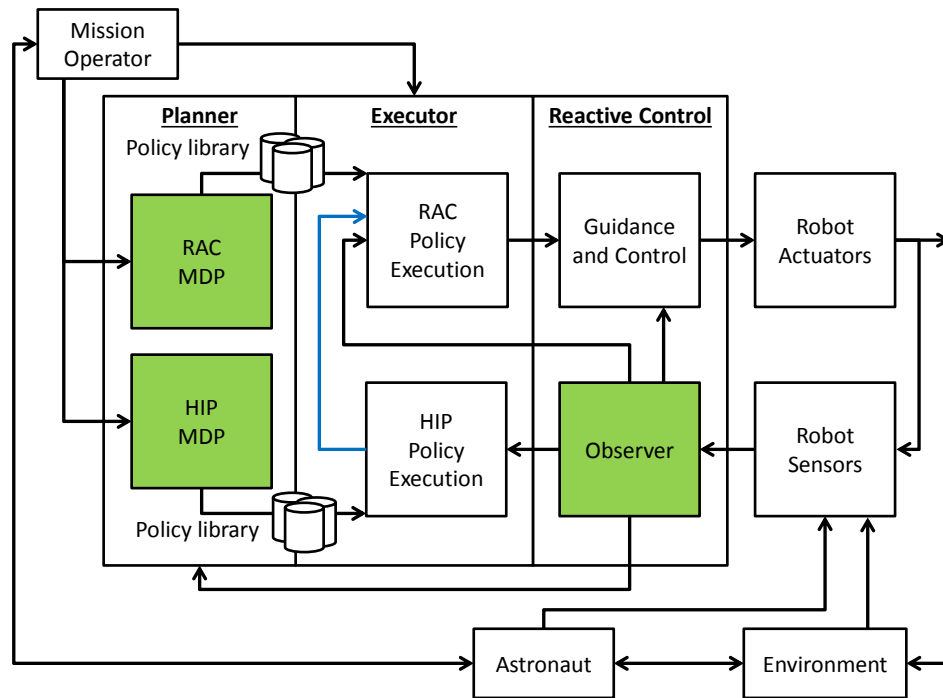


Figure 4-2: 3T Architecture with Decomposed Human Intent Prediction (HIP) and Robot Action Choice (RAC)

Specifics of our autonomy architecture are shown above in Figure 4-2. Sensor data is processed and evaluated by an observer, which generates both continuous and symbolic parameter estimates. This information in turn drives the policies and the lower-level executor processes. The MDP computes policies offline and updates policy use online as needed (in future work). The mission operator selects which policies in the library database should be used and triggers the update process when necessary. The locally-optimal robot action-choice is given to the trajectory generator as the reference input, which in turn outputs a signal used by a low-level controller which creates the explicit

actuator commands for the robot to follow. The robot then acts on the environment and the feedback loop closes.

The robot senses its human companion, and the observer module processes the information to extract the human's current state using known state-of-the-art methods. This includes the human's goals and action-history. This human state data is input into the HIP MDP policy, which then outputs the action the human is most likely to take, which we call a human's intent. Human intent is predicted based on a priori knowledge and real-time observations, and we define it as the best-matched or most-likely in-progress and future action-choice(s) that the human is or will be pursuing to complete their mission goals. This predicted human intent is then taken as part of the input state vector to the RAC MDP executor, along with the robot's own internally-tracked task-level goals, action-history, and zone information which details the safe distance between the human and robot, to select the optimal next action or action sequence for the robot to enact. Selected actions are then executed by the reactive controller and the cycle repeats.

Consider the IVA problem in which we sense the position and pose of the human within the space station environment. This data is added to an evolving time-history of pose and position information and used by the observer to determine the velocity and acceleration of the human's torso and limbs. Through a simple mapping process, also presumed to exist distinct from our MDP and policy executor modules, we match a currently-evolving trajectory to a physical zone that may correspond to a particular action or goal with some accuracy, such as an in-progress but as-yet incomplete movement of a hand towards a drink. These physical zones are calculated offline in the context of the scenario, according to the environment, the actors (human and robot in the environment), and the goals to be accomplished within the environment. We use sensors and a known model of the environment to determine when particular tasks have been completed by the human by sensing the impact of action-completion – for instance, a hand coming into contact with a drink container and then grasping it – and matching it explicitly to start and end segments along the timeline of the trajectory of human motion. We assume that these actions are sensed with 100% certainty once resolved by the recognition process. These known actions are added to an evolving action-history, which the observer tracks as well.

The potential for incorrect resolution by a well-trained system is presumed sufficiently low to be negligible in this work, which allows us to reduce our transition probability tensor by declaring the action-history to only evolve forward in time. If actions are ever incorrectly identified, the action-history will include this action but it will ultimately cycle off the stored history. In this case, transition probabilities may be biased by the incorrect action history element, but safety will still be preserved through the reactive layer. To execute the MDPs in Matlab on a traditional computer, the size of the state space of each MDP model generally needs to remain below $n_s = \sqrt[2]{\frac{3*10^9}{64*2*n_a}}$, where n_a is the number of unique possible choices of policy action. Otherwise memory and computational time constraints may arise, given a standard MDP implementation in which the full probability tensor and policy for each state are stored.

Note that, under our assumptions, if there is no possibility of human conflict or if no human is present, then there is no need for uncertain reasoning. The only uncertainty we explore here stems from the inclusion of the human in the robot’s space. Also note that when future human prediction is unnecessary, the HIP policy block reduces to a pass-through for the fully-observable human goal and action information; the RAC then becomes a reactive planner at the discrete-action level, similar to the one-step reactive implementation discussed in Chapter 3.

Note that while the human need not model the robot during interaction, we must formally restrict our models to not include human tasks or goals that are directly dependent upon the robot’s completion of its own goals. This allows us to simplify our models, but this restriction is unnecessary if we can guarantee the robot’s on-time completion of such cooperative tasks. Relaxation of this assumption is necessary to include the general impact of robot actions on human goals and intent for cooperative work, but this is outside the scope of this thesis which presumes independent agent task sets.

The observer supplies the HIP and RAC policy executors with current MDP state. Specifically, HIP policy matches observed human and environmental state to the predicted human intent action ($^H a_{HIP}$); this in turn is fed into the RAC policy which outputs the optimal robot action given observed robot and environment state, currently-

observed human action ($^H a_{obs}$), and the next predicted action from HIP ($^H a_{HIP}$). We assume that the robot does not fail in any action that it may take, but that it is interruptible mid-action, e.g., to activate a new action when a change in RAC MDP state indicates this new action is optimal prior to ongoing action completion. We presume full observability of each MDP's state, reasonable for a well-equipped IVA environment.

In

Figure 4-3, we show the timing of the state update process for human intent input to RAC, provided through prediction (HIP) and through direct observation. The observer necessarily lags HIP, in that HIP outputs the next predicted action which will be ultimately observed. There are two possible scenarios for observer output: a recognized action is provided (including no-op), or the observer outputs a flag indicating the ongoing action is “unknown” (not yet identified). Note that HIP state updates are considered instantaneous for simplicity, meaning that the HIP will never predict the “currently observed action”, instead it will always predict the next action, until the observer recognizes that next action as in-progress.

Consider Figure 4-3. In case (A), the current action has not yet been predicted, and the observer will output $^H a_{obs}$ as “unknown.” This prompts the HIP policy to calculate the most-likely in-progress action for $^H a_{HIP}$, and the RAC policy to use the HIP output as the in-progress action; no future intent can be determined with high accuracy for RAC use. In case (B), the observer has now (correctly) identified the in-progress action $^H a_{obs}$, which is used by the HIP policy to generate the future intent. The RAC policy uses $^H a_{obs}$ as the in-progress action and $^H a_{HIP}$ as the predicted future intent. The cycle then repeats.

If the HIP model is not perfectly accurate, there is the possibility that the observed action, once finally resolved, may not match what the HIP policy first predicted the in-progress action to be. Thus, in the RAC MDP transition probabilities, we allow for cases where the HIP model might be wrong by allowing an intermediate transition of $^H a_{obs}$ to “unknown” for the wider spread of possibilities prior to that resolution process.

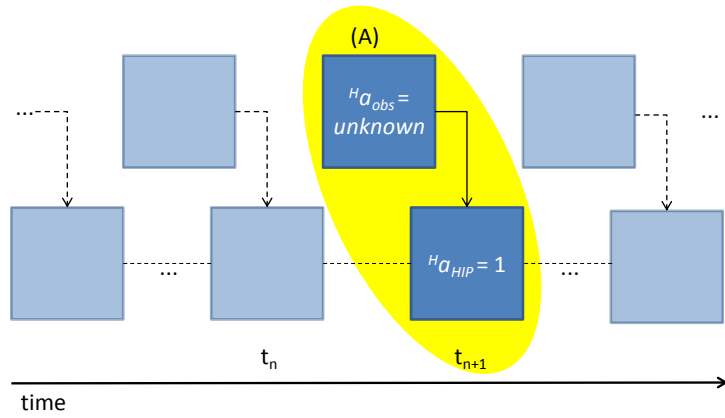
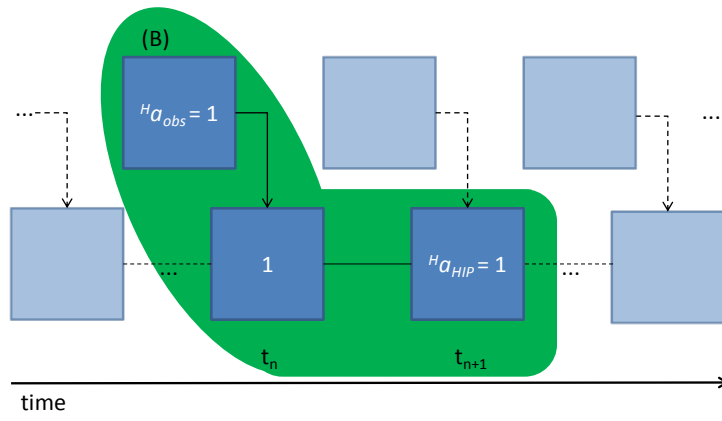
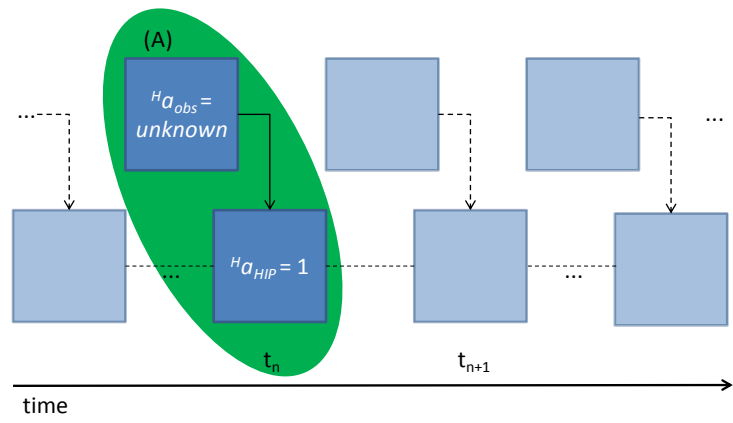


Figure 4-3: Timing of Intent Updates as Used by Robot Action Choice (RAC)

While we assume that the mission operator is a human being at mission control, we expect that the role of the mission operator could be filled by an intelligent sensing system that can identify contexts where mode- and policy-switching is necessary. The inclusion of this module is used to allow a decomposition of the policy state space over multiple task sets, rather than requiring a policy that encompasses the whole of the physical space available to the human and robot over an entire mission. An example of this would be to create the policies necessary over a 24-hour period given the general task timeline for the space station that day. The state spaces for each policy-solution could be decomposed by temporal (time of day) or spatial (per room or working area) separation. Theoretically, to automate the mission operator position, it could be replaced with a MDP that could match likely scenarios – and their associated policies – to the current human-robot system sensed state, take into account the human’s sensed actions and priorities within context, and would output the policy and operational mode most likely to be needed by the robot for the next bout of human-robot interaction.

This brings into focus the delineation of responsibility and oversight between the mission operator and the policy-space of the robot’s planning layer. The planner in our architecture, while in the operating mode that is the focus of our research – human-robot co-location – does not attempt to determine correlations between the shift among larger task sets at different physical locations. Determining what must be included in the state space representation that the robot uses for its determination is the realm of the mission operator. For example, if there are three task sets in a room that a human might perform, the created HIP and RAC policies might include state spaces that include each task set separately, and then every possible combination of the task set spaces. The mission operator would select which of these policies should be used at any point in time, and would attempt to include uncertain terms inside the MDPs that model when a transition between these policies might, or might need to, occur. The planning layer would only concern itself with aspects of, and transitions within, the state space it is given. The mission operator would also concern itself with controlling mode shifts to alternate operational modes outside the scope of this research, such as direct human-robot interaction or other types of collaboration that require explicit verbal or gestural communication.

Chapter 5

Human Intent Prediction¹

5.1 Introduction

For human-robot interaction (HRI) to occur in a shared workspace without collision or conflict, we must have some way for the robot to sense the human and act appropriately based on that knowledge. Chapter 4 described the autonomy architecture defined for this purpose. Chapter 5 now describes and evaluates the Markov Decision Process (MDP) strategy used to predict *human intent*, which we define as *the goal-driven action that a human is or will be attempting to complete*. This policy action is a one-step lookahead to the action the human is expected to execute, beyond what the observer is currently able to predict. No explicit communication takes place between agents, yet we assume that the human's state is fully observable for the reasons outlined in Chapter 4.3. Therefore, this human intent prediction (HIP) MDP model takes as input knowledge-engineered information that allows a best-matching policy to be output for use by the robot. This knowledge is manually generated in this research; any process for experimentally learning such information is relegated to future work. The resulting HIP MDP policy takes as input only a human's current high-level goals and abbreviated action history. This includes their current in-progress action, which we assume can be found using one of any number of available action-recognition techniques, as discussed in Chapter 2.5.1. The HIP policy outputs the most likely future action(s) of the human, requiring modeling and detection of all actions that have an impact within and upon the shared environment. We hypothesize that solving for and making available this additional predictive lookahead to the robot's action-choice procedures will enable the robot to make better-informed decisions under the majority of circumstances.

¹ Some of the information in this chapter is reproduced or modified in part or in full from References [104], [105], and [107].

Most MDP problem formulations give policies that match observed state to an optimal “action” when executed. Our HIP model requires a non-standard formulation because the idea of “optimality” under these circumstances is non-traditional. Here, we do not try to find the “best” parameters that would result in the most rational human behavior or action and the most efficient state transitions towards goal satisfaction. Instead, we measure HIP MDP performance (reward) by how well the policy is expected to match a human’s actual semi-rational responses in real-time. We explain below how this performance metric translates to parameter selection in the HIP MDP formulation, particularly the reward function. We define two types of human models: a ‘*simulated human*’ model, which is generalized to statistical norms of human reaction obtained from human subject testing; and a ‘*human matching*’ model, which attempts to produce the same output as a particular human subject and requires online updates for improved accuracy. These two modeling perspectives can be represented using the single HIP MDP formulation presented in this chapter.

Below, we first describe our HIP MDP formulation. Next, we present two case studies and their domain representations, discuss the expected impact of changes in the model parameters, and then evaluate simulation results. The two scenarios we discuss are cast in EVA and IVA environments. The EVA case requires the astronaut to remove a panel on a spacecraft, where the astronaut’s actions include the retrieval of a screwdriver, removal of screws and removal of the panel. The IVA environment is similar to the test scenario used in the human subject experiments described in Chapter 3; the astronaut’s actions include eating chips, drinking soda, solving math problems, and pressing a button while seated at a workstation. We present metrics for evaluating the performance of the generated HIP policies, and apply them to analyze simulation results.

5.2 Markov Decision Process (MDP) Formulation for Human Intent Prediction

The MDP is defined as: [39,35]

$$MDP = \{S, A, T(s^i, a_k, s^j), R(s^i)\} \rightarrow \pi(s^i) \quad (5-1)$$

It is comprised of a set of states S , available actions A , state-dependent rewards $R(s^i)$, and transition probabilities $T(s^i, a_k, s^j)$. We define an “action” a_k as a primitive task that may

require multiple motions but will complete without interruption. An optimal policy for the MDP can be found using Bellman's equation (see Chapter 2.3.2, Equation (2-22)).

The human's physical state is received from the observer module in the architecture (see Figure 4-2 in Chapter 4.3) and is assumed to be fully-observable. State includes the human's goals and a k -observation history of human actions we denote as the action-history with $k = n_h$. State may include the current action being undertaken by the human $a_{n_h+1}^i$ (in-progress but not yet completed), if available for the application domain. The actions represent the task-level primitives the human would execute alone or in sequence to achieve a goal. The reward of executing a task-level action in any state is a function of expected goal satisfaction. The transition probabilities are calculated from the current state, which includes the action-history. In a well-formulated HIP MDP, the optimal action chosen for any given state most closely matches the choice that the human would actually take given current goals and environment state.

The MDP requires a finite, discretized state-space, and computational tractability requires minimization of state-space size thus model complexity. In the HIP MDP, the state is comprised of a set of features, most of which have binary values except for the action-history. Each attribute in the action-history is integer-valued and can take the value of any corresponding action in the set of actions A , which has cardinality n_a each denoted by an element in $[1 n_a]$. The specific model for the HIP MDP is given below.

5.2.1 States and Actions

The set of n_s modeled human states $S = \{s^1, s^2, \dots, s^{n_s}\}$ each denoted s^i includes two elements: the human's goal-state $\{G^i, F^i\}$, and the abbreviated action-history $\{A^i, a_{n_h+1}^i\}$ of observed actions.

Thus, each HIP MDP state is given by:

$$s^i = \{G^i, F^i, A^i, a_{n_h+1}^i\} \quad (5-2)$$

The human attempts to satisfy goals in $\{G^i, F^i\}$ via action-choice a_k at any given time. We differentiate between two types of human goals: a set of n_g mission goals $G^i = \{g_1^i, g_2^i, \dots, g_{n_g}^i\}$, and a set of n_f high-priority interruptive goals $F^i = \{f_1^i, f_2^i, \dots, f_{n_f}^i\}$. We assume that these objectives are conditionally-independent from each other and that they cannot be further simplified or combined. A human's need to satisfy a high-priority goal F^i could conditionally impact the probability of achieving mission goals in G^i , the reward they associate with goal completion, or a mixture of both. These dependencies, however, would be difficult to accurately capture, so our models instead simply quantify the relative reward of each goal to guide HIP MDP policy optimization.

Generally, an action could also impact more than one goal, but again for simplicity we map actions to not more than two distinct goals so that interaction and dependence can be minimized. A sequence of actions may be needed to accomplish some goals, while in other instances, a single action may be sufficient. Action sequences are central to our model as astronaut and robot task completion often requires a multi-step script. For instance, all high-priority goals are assumed to complete following one uninterruptible action associated with that goal f_z^i .

High-priority goal achievement may be required either from flags set at the onset of the mission or by sensed events that trigger the binary-valued goal achievement flags. Examples of high-priority goals include handling a warning or alarm or "catching" an unexpectedly dislodged "floating" object. Generally, each mission goal and high-priority goal is binary-valued, $g_z^i \in \{0,1\}, f_z^i \in \{0,1\}$. A mission goal is only fulfilled (1, complete) or unfulfilled (0, incomplete), and a high-priority goal is active (0, flagged as requiring attention) or inactive (1, complete or not required). We could use a finite-valued set for each goal in cases where the additional knowledge of the explicit action-history sequence does not change the outcome of transition probability or reward. Instead, we estimate progress toward goal completion through inclusion of an abbreviated action-history, as described below.

Action-history is part of the state s^i , supporting a finite-memory structure that allows recent past action choices to impact future rewards but still supports the Markov assumption required for the MDP formulation. The abbreviated action-history $A^i = \{a_1^i, a_2^i, \dots, a_{n_h}^i\}$ of limited length n_h supplies sufficient information about the human's past for HIP MDP decision-making ('human intent prediction'). The actions stored in the action-history can indicate partial goal completion in the transition probability function to determine the likelihood of future goal fulfillment. The action-history can also be used to reward certain sequences of actions over others. The parameters in the action-history, a_k^i , $k = \{1, \dots, n_h\}$, are from the set of human actions $A = \{1, 2, \dots, n_a\}$, thus $a_k^i \in A$. We assume that the set of human actions A modeled in the MDP collectively support completion of all specified mission objectives. An MDP *action* refers to a subtask that may require multiple primitive movements through or manipulations within the workspace to complete. We further assume an external action-recognition capability underlying policy execution that can accurately translate thus "observe" each task-level action from observed motions and manipulations. For now, we also assume that every action included in the action-history did complete successfully.

The last term, $a_{n_h+1}^i$, is the *current or in-progress action* as identified by the observer through real-time action-recognition. If the in-progress action is identified with certainty, the optimal policy for the HIP MDP is the next most-likely action that the human will undertake after $a_{n_h+1}^i$ completes. Since we assume all movement to be goal-oriented, if the process of action-recognition cannot surpass a threshold of certainty for its result, the observer labels the action $a_{n_h+1}^i$ as either "no-op" (human appears to be idle) or "unknown" (when identified as moving but the goal is not yet identified). The policy output of the HIP MDP in this case is a model-predictive estimate of $a_{n_h+1}^i$. Per the Chapter 4 architecture, $a_{n_h+1}^i$ is then passed to RAC.

In summary, the HIP MDP predicts the human's next intended action, thus is always one step ahead of the observer. An accurate HIP model will find the optimal choice a_k that matches the next action $a_{n_h+1}^i$ actually observed.

5.2.1.1 Length of action-history

We assume conditional independence of the goal objectives, as previously discussed. For now, we assume that the value of n_h is consistent for each model and can be chosen or otherwise optimized offline.

A goal can be completed by a single action or sequence of actions. An action-sequence may be interruptible or non-interruptible. If the latter, the sequence must be restarted after any interruption. If each sequence is non-interruptible, the length of the action-history can be set to the length of the longest action-sequence. An example of a non-interruptible sequence would be running and observing an experiment, where skipping a step or failing to record data over a period of time would require the experiment to be restarted. Interruptible action sequences are perhaps more common. For example, hand-tightening a bolt can be performed by multiple “turn-bolt” actions with no negative consequence due to interruptions between turns of the bolt.

In some cases, the order of actions may be partially specified. For instance, in unfastening a panel from a wall, each screw must be removed, but the screws could be removed in any order. Interruptible action sequences can require an increase to action history length, while partial orders do not lengthen the history but do require recognition of more permutations in the sequence when computing transition probabilities. We discuss action-history length in the context of specific case studies below. The MDP assumption requires this.

We have chosen action history over inclusion of intermediate goal state values because this allows an intuitive mapping of observed action-sequences into the state, whereas partial goal completion might be more difficult to observe. Including the action history also allows the reward function to assign preferences to specific action orderings based on historic preference over the a_k past action horizon.

5.2.2 Transition Probability Function

The transition probabilities for the HIP MDP are dependent upon the action-history and the current goal state. We use the action-history and in-progress action $(A^i, a_{n_h+1}^i)$ to determine how likely it is that a goal will or will not transition from

incomplete/unfulfilled (0) to complete/fulfilled (1). Each goal is fulfilled by an n -tuple action-sequence, where n_z^i is the minimum number of actions to complete a particular goal g_z^i for a sequence of actions in A^i for state s^i . Action-sequences may be totally or partially ordered in A^i , as well as interruptible or non-interruptible. Mapping a sequence of actions to a goal requires knowledge of the sequence of events, hence the action-history. This is necessary to correctly represent scripted action sequences within our formulation. We include all combinations of possible orderings in the set of sequences that may accomplish a goal g_z^i . All high-priority goals f_z^i are assumed to be accomplished with a single action for simplicity and because they require simpler short-term actions. Action history length n_h is consistent for each model and presumed pre-specified. In our case studies, we choose values of n_h that reflect memory or reference to an action script followed by a human astronaut.

Goal sequences from Equation (5-3) simplify transition probability computation by eliminating consideration of impossible state transitions ($p=0$). Action-sequences can be ranked by efficiency and expected preference.

The MDP transition probability tensor is:

$$p(s^j | s^i, a_k) = T(s^i, a_k, s^j), s^i \in S, s^j \in S, a_k \in A$$

$$\text{satisfying } \sum_{j \in \{1, \dots, n_s\}} T(s^i, a_k, s^j) = 1, \forall i \in \{1, \dots, n_s\}, \forall k \in \{1, \dots, n_a\} \quad (5-3)$$

Equation (5-3) represents the probability that the system will transition to a state s^j when the human performs action a_k in state s^i .

The transition probability map, specified as a tensor or set of action-specific matrices, capitalizes on the fact that for the astronaut-robot domain, the astronaut will likely follow step-by-step procedures for typical activities conducted on the space station. For cases where there are known procedures, HIP transition probabilities are set to near one for the expected state-action outcomes. Alternative paths, while less likely, can still be modeled.

The transition probability equation we use for human intent prediction is:

$$T(s^i, a_k, s^j) = \prod_{z=1}^{n_f} p(f_z^j | s^i, a_k) * \prod_{z=1}^{n_g} p(g_z^j | s^i, a_k) \quad (5-4)$$

This product formulation for the MDP probability tensor presumes conditional independence between mission goal and high-priority goal flags, and reduces to Equation (5-5) for conditional independence between all goal flags:

$$T(s^i, a_k, s^j) = \prod_{z=1}^{n_f} p(f_z^j | f_z^i, A^i, a_{n_h+1}^i, a_k) * \prod_{z=1}^{n_g} p(g_z^j | g_z^i, A^i, a_{n_h+1}^i, a_k) \quad (5-5)$$

In Equation (5-5), we presume the action history in s^j will contain action a_k as the “previous action” with 100% probability at the next iteration. We also nominally assume that either a mission goal (or goals) or a high-priority goal will transition from one state to the next, but not both simultaneously. This reflects the expectation that either a mission goal or goals will transition, a high-priority goals flag will change, or no goal flags will transition between s^i and s^j . The action history is always expected to update between states. These cases are illustrated in Figure 5-1.

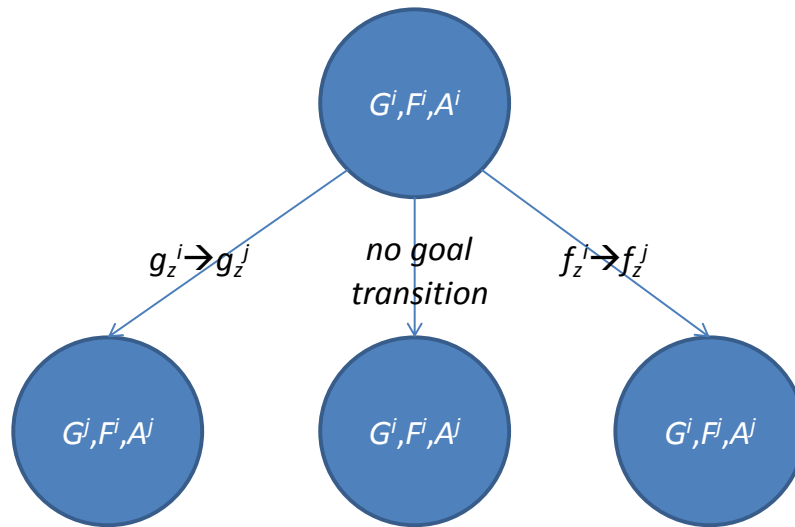


Figure 5-1: General transition cases for HIP, with no in-progress action supplied

To reduce the complexity of our probability tensor, only next states s^j corresponding to action-history updates that follow an expected action sequence or script will be reachable

from s^i , where reachability implies a nonzero transition probability from s^i to s^j . The action-history update process from $A^i \rightarrow A^j$ as shown in Figure 5-1 is assumed to proceed as follows: the oldest history action is forgotten, all history actions are shifted by one timestep (slot in the history), and the newest action is set to the chosen action a_k .

This handling of the action-history described above requires an assumption that all actions will be observed thus properly inserted in the action-history. We also allow an action to be unknown to represent delay in observations.

As shown in Table 5-1, we refer to a transition for a mission-goal g_z^i due to a completing action a_k as having probability p_{kz} in the no-change case ($p(g_z^j = 0 | g_z^i = 0, a_k) = p_{kz}$), and $(1-p_{kz})$ for a transition from 0 to 1 ($p(g_z^j = 1 | g_z^i = 0, a_k) = 1 - p_{kz}$). In the general case, these could have superscript i (p_{kz}^i) to distinguish between separate probabilities for each state s^i .

Table 5-1: Mission-goal transition probabilities for HIP

Action	Goal state transition		Probability
	g_z^i	g_z^j	
a_k	0	0	p_{kz}
	0	1	$1-p_{kz}$
a_k	1	0	0
	1	1	1

The first term of Equation (5-5) is the effect of the high-priority goal flag on the transition probability. For our work, we assume that any action that influences a high-priority goal has a 100% probability of that high-priority goal becoming inactive once the action is completed.

Mission goal transition probabilities – the second term in Equation (5-5) – are a function of goals already accomplished, the action history, and the current selected action. However, there are several simpler cases that capitalize on conditional independence when possible. Mission goal probability may become:

- $p(g_z^j = 1 | g_z^i = 1) = 1$ when the goal flag is set and cannot reset ($g_z^j = g_z^i$)

- $p(g_z^j | a_k)$ when only the current choice of action matters in determining goal satisfaction, (e.g., for a coin flip)
- $p(g_z^j | g_z^i, a_k)$ when a goal is accomplished by a single action
- $p(g_z^j | g_z^i, A^i)$ when there is a delayed reaction time (for instance, after taking a medication) but no new action is required.

A simple example formulation for the probability of f_z^i is:

$$p(f_z^j | s^i, a_k) = \begin{cases} 0 & \text{if } f_z^i = 0 \text{ and } a_{n_h+1} = \text{unknown and } a_k \text{ does not satisfy } f_z^i \\ 0 & \text{if } f_z^i = 0 \text{ and } a_{n_h+1} \neq \text{unknown and } a_{n_h+1} \text{ does not satisfy } f_z^i \\ 1 & \text{otherwise} \end{cases} \quad (5-6)$$

In Equation (5-6), f_z^i remains set (fulfilled) once accomplished. If f_z^i is clear (unfulfilled), we assume an action (a_k or $a_{n_h+1}^i$) that can fulfill f_z^i will fulfill f_z^i with probability 1.

An example formulation for the second term, $p(g_z^j | s^i, a_k)$, may have two parts: the probability of mission goal objective g_z^j being or becoming 1 (completed) due to the action history A^i of state s^i , e.g. the impact of the action history on the transition probability of the goal completion, independent of the action a_k ; and the impact of the selected a_k alone.

A more general form of Equation (5-5) is necessary to allow for conditional dependence between some of the goal states. We do this to enforce constraints on possible transitions between states. This is discussed further in Chapter 5.4.1 and Chapter 5.4.2.

$$T(s^i, a_k, s^j) = f(s^j | s^i, a_k) \quad (5-7)$$

For models with a mixture of conditionally-dependent and conditionally-independent goals, each can be calculated individually and the product of transition probabilities can then be taken, as will be described for specific case studies.

To simplify our model and reduce computational complexity, we reduce the number of non-zero probabilities in our tensor to n_{bk}^i per action a_k . The uppermost bound on n_{bk}^i is

the number of states s^j in S with a possible action-history including an updated A^j with parts of A^i , $a_{n_h+1}^i$, and a_k . We define n_{Tk}^i as the number of state transitions with non-zero probability $T(s^i, a_k, s^j)$. n_{bk}^i is the number of possibilities of transition minus one, $n_{bk}^i = n_{Tk}^i - 1$, representing the final probability computed to make all sum to 1. n_b^i , for a state s^i , is then defined as the total number of possibilities of transition minus the number of actions:

$$n_b^i = \sum_{k=1}^{n_a} (n_{Tk}^i - 1) = \left(\sum_{k=1}^{n_a} n_{Tk}^i \right) - n_a \quad (5-8)$$

Above, we are simplifying the MDP by eliminating unlikely transitions from the (s^i, a_k) pair. If we need to relax this assumption but do not have additional statistical knowledge, we could subtract a small ϵ_{kz} from each non-zero probability p_{kz} and divide the ϵ_{kz} evenly among all the other n_ϵ^i states, where $n_\epsilon^i = n_s - \sum_{k=1}^{n_a} n_{Tk}^i$, for a transition probability of $\epsilon_{kz}/n_\epsilon^i$ for each of these low-likelihood states from s^i . Doing so allows the model to account for cases where there is low but nonzero possibility of transition to a state with low or negative reward, in particular to ensure such state sequences are handled properly in a real policy.

5.2.3 Rewards

The reward function defined for human intent prediction is given by:

$$R(s^i) = \sum_{z=1}^{n_g} \alpha_z r_1(g_z^i) + \sum_{z=1}^{n_f} \beta_z r_2(f_z^i) \quad (5-9)$$

This reward function $R(s^i)$ for each state is based on fulfillment of single-event and recurrent objectives of the human, providing a straightforward representation of human preference for the MDP. The reward functions are based on the normal or expected behavior of the human – a baseline of average behavior. We assume that discounted rewards for future states are the sole means to account for action preferences. Similarly, we also assume that, prior to learning, a human feels rewarded after seeing the results of the goals they completed, but not necessarily from performing each action itself. Once

the human understands and has learned which precursor actions will lead to eventual reward (e.g., from a script or experience), the full sequence will be seen as rewarding, including intermediate steps taken towards that goal. This initial pre-learning state is comparable to Equation (5-9), where reward is given once an objective is met; there is no reward for partial success. Use of the Gauss-Seidel value-iteration procedure over the Bellman equation allows value to be, in essence, back propagated through the model during the optimization process, similar to the way that reward is learned by a human.

Function r_1 calculates the impact of the current mission goal states on total reward to the human astronaut, while function r_2 calculates the impact of the high-priority goal states. Because we assume g_z^i and f_z^i are known sensed states, r_1 and r_2 are independent of A^i in our formulation. When the model transitions to a new state s^j , the goals and the action history in s^i are also updated inside the transition equation. We restrict function r_1 to non-negative values, while r_2 can be positive or negative.

Reward weights α_z , β_z are chosen in the range [0 1] and scaled based on maximum possible values of r_1 and r_2 . We assume that the weighting variables are constant for a given MDP policy. If a high-priority goal flag is active but not complete, the value of r_2 is negative and incurs a large cost so that, when accounting for weighting factors, the model will favor accomplishing high-priority goals. While our simple reward function does not contain costs (e.g., fuel, energy) for action completion of a_k , such costs could be included. Any no-op action with zero “cost” would then be selected as a default when no action can otherwise accomplish a rewarded goal.

Because the goals are binary-valued {0,1} attributes, we can use the goal flags directly when calculating reward r_1 . Equation (5-10) shows an example reward function; we assume that $|k_z| > 1$ to encourage high-priority goal achievement.

$$\begin{aligned} r_1(g_z^i) &= g_z^i \\ r_2(f_z^i) &= \begin{cases} 1 & \text{if } f_z^i = 1 \\ -k_z & \text{if } f_z^i = 0 \end{cases} \end{aligned} \quad (5-10)$$

In our HIP models, reward is only given once an objective is met; there is no reward for partial success. If a goal requires multiple actions to accrue in the action-history, those interim states receive no reward. Specific action history content could then factor into reward computation. In the above example, however, multi-action sequences will be identified through iteration with the Bellman equation and a relatively high discount factor given transition probabilities that appropriately reflect reward only after accomplishing specific action sequences.

To encode human preference with respect to goal-driven behaviors, we must address whether each behavior is driven by goals without context, or whether context drives human adaptation to the special circumstances of the environment [100]. We define structured context as prior knowledge specific to the environment – the human and robotic agent’s placement and movements within it, according to the goals which must be achieved. Ref. [86] gives an example of learning for a related problem involving collaborative human-robot team training. Because we reward only goal completion, rather than also rewarding intermediate milestones toward each goal, differentiation between multiple solution paths is only by path length, given a discount factor $\gamma \ll 1$.

5.3 Metrics for Performance Evaluation

We evaluate HIP MDP formulations by comparing policy outputs. We assess the impact of reward function weights by varying reward weightings of two goals at a time, and assess the impact of action-history length by varying the values of n_h . We assess the level of transience of these impacts by comparing the policy outcomes of the same reward and action-history length variations against different choices of probabilities p_{kz} .

When evaluating the policies resulting from different parameter choices, we discuss:

- changes in optimal action-choice as determined by the policy for the various G^i and F^i goal state transitions over all possible states
- changes in optimal action-choice as determined by the policy over each of a set of four “groupings” per the two tested goal-states $\{g_q^i, g_r^i\}$ in s^i – when both are unfulfilled ($\{0,0\}$), when one is unfulfilled and one is fulfilled ($\{0,1\}$ and $\{1,0\}$), and when both are already fulfilled ($\{1,1\}$)

We can evaluate the relative impact of changes in the model by varying the values of two parameters in the HIP model at a time, solving for the optimal policies for each set of parameters, and looking for the tradeoffs.

Note that we are not using common performance metrics such as time, energy, or efficiency to evaluate these models. Instead, what is most important here is how well the models can track human behavior – how well they can match a human’s statistics and explicit observed partial action-policy. What is also important is whether the model is reasonable and human-intelligible; we want the policies created by using a particular set of equations and parameters to make sense (intuitive setup and outcome), thus changes in the model parameters need to have explainable and understandable effects on the policy-output (consistency).

Below, we examine the ability to model both deterministic and stochastic problems using the HIP MDP formulation. We start by discussing the ability of our representation to model the simpler deterministic case, and then move to a discussion into how changes in the parameter values impact the behavior (policy output) when we move to a stochastic model. We also discuss stochastic model options, and examine the tradeoff between different types of goal interdependencies, e.g., independent goals (e.g., eating and math), partially-dependent goals (e.g., eating and drinking), and low-priority mission goals versus high-priority goals (e.g., eating and button-pushing). Currently, values for the transition probabilities are manually chosen, though they could also be determined from experimental results, and tuned to individual human actors when possible.

5.4 Case Studies

We investigate the impact of changes in MDP parameter choices including relative goal completion rewards and action history lengths, then we examine the effects of transition probabilities on policies. A series of simple examples are used to illustrate HIP MDP modeling choices.

Below we first show how HIP can occur with a deterministic model, then discuss uncertainty modeling. Next, we discuss domain representations for a deterministic HIP model applied to an EVA scenario with an unknown in-progress action. We then present

a series of progressively-complex stochastic HIP models for an IVA scenario, also with an unknown in-progress action. Finally, we discuss how to include the in-progress action in our models, and its impact on modeling and policy output. Evaluation of the HIP method in relation to other HIP algorithms is left to future work. A sound comparison would require human subject data for learning and comparatively evaluating modeling and inference methods.

For the presented case studies, we evaluate the changes in policy seen when varying the reward weightings, transition probability parameters, and action history length. For all examples, MDP policies are generated using the standard Gauss-Seidel value iteration algorithm over an infinite horizon, with discount factor 0.95 and an acceptable error bound of $1 \cdot 10^{-5}$. The choices of infinite horizon and a relatively high discount factor allow the MDP solver to account for reward obtained after multiple steps in a sequence.

5.4.1 Encoding Pre-existing Script(s) within a Markov Decision Process

While deterministic planners are effective at finding optimal solutions in deterministic spaces, a MDP can also find solutions in deterministic spaces when all probability tensor values are in the set $T(s^i, a_k, s^j) = \{0,1\}$. This section proposes a series of deterministic HIP examples. Such deterministic models may be realistic for space EVA or IVA scenarios in which an astronaut follows a rehearsed script. A script provides a known progression of state changes and actions which produce them, an explicit set of actions will always occur in a certain order given a starting state. A series of deterministic HIP cases is presented below.

5.4.1.1 Case #1 – explicit transition path (goals), reward all goal tasks equally

In this case, we explicitly define the possible transition path according to the goal/policy-action progression in the script, while rewarding all goal tasks equally (i.e., all goal rewards set to 1).

Table 5-2 shows an example motivated by the Chapter 3 study. In this domain, there are two goals (work motivation, blood sugar level) and two actions (computer work, eat chip). An example script using the domain from Chapter 3 would be: someone will do

computer work (to satisfy work motivation), then eat a chip (raise blood sugar level). No action-history is used.

Table 5-2: First example representation, computer work then eat chip											
State #	$g_1^i =$ raise blood sugar level	$g_2^i =$ work motivation	$R(s_i) =$ $g_1^i + g_2^i$	Transitions to next state # by action $a_x (p=?)$							
				a_1 eat chip				a_2 computer work			
				1	2	3	4	1	2	3	4
1	0	0	$R(s^i) = 0$	1	0	0	0	0	1	0	0
2	0	1	$R(s^i) = 1$	0	0	0	1	0	1	0	0
3 (unreachable)	1	0	$R(s^i) = 1$	0	0	1	0	0	0	1	0
4	1	1	$R(s^i) = 2$	0	0	0	1	0	0	0	1

If we relax the order of actions in the script, allowing a partially-ordered plan instead, the state space will still include two goals and no action-history; however, the transition probabilities change slightly, as shown in Table 5-3 (changes are shown in italics+bold).

Table 5-3: First example representation, computer work and eat chip (in any order)											
State #	$g_1^i =$ raise blood sugar level	$g_2^i =$ work motivation	$R(s_i) =$ $g_1^i + g_2^i$	Transitions to next state # by action $a_x (p=?)$							
				a_1 eat chip				a_2 computer work			
				1	2	3	4	1	2	3	4
1	0	0	$R(s^i) = 0$	0	0	I	0	0	1	0	0
2	0	1	$R(s^i) = 1$	0	0	0	1	0	1	0	0
3	1	0	$R(s^i) = 1$	0	0	1	0	0	0	0	I
4	1	1	$R(s^i) = 2$	0	0	0	1	0	0	0	1

5.4.1.2 Case #2 – explicit future-reward path, equal transition probabilities

In this case, we explicitly define one sequence for goal completion as the most rewarding, according to the goal/policy-action progression in the script, while distributing equal probability among all transitions to current (same) and future (forward-progressing) states. This sequence is encoded within the reward function weights, with decreasing weight assigned to goals that can be completed last or near last.

Table 5-4 presents a case with the same goals and actions as before, but a different reward and probability tensor structure. This different structure represents the progression of goal completion as given in the script. In this example, the state space includes two goals and no action-history, and follows the same ordered script as case #1 above. The reward function weight for work motivation goal is 2, reward function weight for raising the blood sugar level is 1, and all other reward function weights are set to 0. (Note, however, that state 3 could have $R(s^i)$ set to 0 for greater efficiency.) Technically, this MDP model is not deterministic, but its policy output has been verified to match the deterministic script with an appropriate choice of discount factor (close to 1).

Table 5-4: Second example representation, computer work then eat chip											
State #	$g_1^i =$ raise blood sugar level	$g_2^i =$ work motivation	$R(s_i) = g_1^i + 2g_2^i$	Transitions to next state # by action $a_x (p=?)$							
				a_1 eat chip				a_2 computer work			
				1	2	3	4	1	2	3	4
1	0	0	$R(s^i) = 0$	1/2	0	1/2	0	1/2	1/2	0	0
2	0	1	$R(s^i) = 2$	0	1/2	0	1/2	0	1	0	0
3	1	0	$R(s^i) = 1$	0	0	1	0	0	0	1/2	1/2
4	1	1	$R(s^i) = 3$	0	0	0	1	0	0	0	1

For HIP, transition probabilities must capture the likelihood that the human will succeed vs. fail in performing the action they are attempting. The optimal policy gives the maximum reward, including expected future reward. In Table 5-4, this highest-weighted term is completed in-sequence, but this method will only work when the ‘end’ goals (goal

tasks, not compound tasks) cannot deactivate and reactivate again. This is because, if interim states are expected to reactivate according to the transition probability model, the policy may give undue weight to this condition and cycle between lower-reward states, never attempting to complete all goals. If cost is incurred for each action and the final reward is not sufficiently high, the necessary action-sequence may never be selected.

5.4.1.3 Case #3 – explicit transition path (actions), reward all goal tasks equally

In this case, we explicitly define deterministic multi-step action sequences according to the action-history/policy-action progression in the script, while rewarding all goal tasks equally. The history length n_h is set equal to the number of goal-impacting actions in the script (e.g., not no-op); the transition probability is set to 1 only when the exact action-sequence of events up to that point in the sequence matches the script perfectly.

The script is the same as the ordered script for case 1 above. In this example, the state space includes one goal (which encapsulates both goals having been completed) and an action-history of length $n_h=2$. We combine the goals here for simplicity of explanation.

Table 5-5: Third example representation, computer work then eat chip												
State #	$g_1^i =$ raise blood sugar level and sate work motivation	$\{a_1^i, a_2^i\} =$ action-history	$R(s^i) = g_1^i$	Transitions to next state # by action $a_x (p=?)$								
				a_1 eat chip			a_2 computer work			$a_0 =$ no-op		
				1	2	3	1	2	3	1	2	3
1	0	{0,0}	$R(s^i) = 0$	1	0	0	0	1	0	1	0	0
2	0	{0,2}	$R(s^i) = 0$	0	0	1	0	1	0	0	1	0
3	1	{2,1}	$R(s^i) = 1$	0	0	1	0	0	1	0	0	1
X (all others)	--	{--,--}	$R(s^i) = 0$	$p=1$ to remain X			$p=1$ to remain X			$p=1$ to remain X		

Note that this third method is effective because we are using an infinite horizon solver with a large discount factor – the belated reward will be back-propagated to the beginning state only through those states with possible transitions.

5.4.1.4 Multiple domain-modeling options

While each of the deterministic case studies works in isolation, it might be more efficient to combine modeling strategies. Some of these examples result in larger state spaces with long convergence times. The distribution of equal transition probability across all forward-progressing states in case 2 is the only exception. An example of another option would be: explicitly define the deterministic multi-step action sequences *and also* define one sequence for each goal completion flag as the most rewarding, according to the goal/policy-action progression in the script (rather than rewarding all goal tasks equally). These options should, however, be chosen for their accuracy in modeling the domain, as some domains may contain goals that can be satisfied flexibly with little overhead for an “alternate” sequence.

Using a policy resulting from a MDP for a deterministic scenario would have the same effect as giving a copy of the human’s explicit action-script to the RAC module. The robot is aware of the script the human is following, so it would use a perfect model of the human’s action assuming that the human does not deviate from the script.

5.4.2 Case Study #1 – EVA space repair example, deterministic system

In this section, an example HIP MDP is developed for a simple EVA scenario, a “spacecraft panel removal” activity. For illustrative purposes, we again assume that the observer always treats the in-progress action as ‘unknown’, implying that the action-recognition observer will provide the observed action to the MDP policy executor only after that action has completed. We also assume deterministic execution in this initial EVA case.

In the nominal case, this panel removal activity requires that a toolbox be retrieved for a screwdriver, four screws removed from the panel, and then the panel itself removed. Note that this is a partially-ordered plan, given that the screws could be removed in any order.

In this human task model, we define the baseline problem to include only the goal of panel-removal, with no trade-offs with other goals – we wish to demonstrate the model’s capability of encoding a structured script in a MDP. The actions associated with panel-

removal have ordering constraints. For instance, in the ideal case a human cannot grasp and remove a panel while already holding a screwdriver in one hand, due to a lack of dexterity in the space suit gloves on EVA.

For the MDP, we assume that action-sequences do not need to be specified for each leg of the trajectory, only at the task level. For instance, unscrewing a screw would involve the following sequence: move hand to object (the screw), adjust screwdriver in hand to correct position, unscrew object, move object to destination (tether to side of panel), deposit object (place screw at correct secondary location). We assume for now that each of these task-level actions are uninterruptable, and that an action has not been included in the action-history unless it has been completed successfully.

5.4.2.1 States and Actions

For our case study, we model the above simple extra-vehicle activity (EVA) scenario including the actions of toolbox retrieval (1), picking up a screwdriver (2), removing a screw from a panel from a given position X ($X=\{1,2,3,4\}$, completed by actions a_{31} , a_{32} , a_{33} , and a_{34} , respectively), setting down a screwdriver (7), and removing a panel (8). These eight tasks are required for astronauts in EVA as well as for humans on Earth. We do not include a no-op action because in this section we assume these EVA scripts are executed deterministically and thus should involve ‘pauses’ in expected work.

Table 5-6 and Table 5-7 describe the human’s actions and goals, and the meanings of the variable status used for our domain. The actions are integers from 1 to 8, corresponding to the eight labels given below.

Table 5-6: Domain Representation of actions a_k^i

Discrete Value	Corresponding Action
1	toolbox_retrieval (a_1)
2	retrieve_screwdriver (a_2)
3,4,5,6	remove_screw_from_panel_position_X (a_{3X} , $X=\{1,2,3,4\}$)
7	set_down_screwdriver (a_4)
8	remove_panel (a_5)

The mission goals are binary-valued, with 0 corresponding to incomplete and 1 corresponding to complete; we use subsets of these in our example representations (see Table 5-7).

Table 5-7: Domain Representation of goal-objectives

Goal Obj.	Values	Corresponding Goal
g_1^i	{0,1}	panel_removed (g_1)
g_2^i	{0,1}	close_positioning_of_toolbox (g_2)
g_3^i	{0,1}	all_screws_removed (g_3)
g_{3X}^i	{0,1}	screw_in_position_X_removed (g_{3X} , $X=\{1,2,3,4\}$)
g_4^i	{0,1}	holding_screwdriver (g_4)

In this case study, this script includes several constraints: the toolbox must be positioned close to the human before the screwdriver can be retrieved, the screwdriver must be held for a screw to be removed, all screws must be removed and secured, and the screwdriver must not be held before the panel can be removed. A relaxed order in screw removal might require the ability to encode a partially-ordered plan into the model.

For the case study above, there are three basic ways to define the state space, with pros and cons for each:

$$\begin{aligned} \text{Case 4a:} \quad & s^i = \{g_1^i, a_1^i, \dots, a_8^i\} \\ & g_z^i \in \{0,1\}, a_k^i \in \{1, \dots, n_a\}, n_a = 8, a_k \in \{1, \dots, n_a\} \end{aligned} \quad (5-11)$$

$$\begin{aligned} \text{Case 4b:} \quad & s^i = \{g_1^i, g_2^i, g_3^i, g_4^i, a_1^i, \dots, a_4^i\} \\ & g_z^i \in \{0,1\}, a_k^i \in \{1, \dots, n_a\}, n_a = 8, a_k \in \{1, \dots, n_a\} \end{aligned} \quad (5-12)$$

$$\begin{aligned} \text{Case 4c:} \quad & s^i = \{g_1^i, g_2^i, g_{31}^i, g_{32}^i, g_{33}^i, g_{34}^i, g_4^i\} \\ & g_z^i \in \{0,1\}, n_a = 8, a_k \in \{1, \dots, n_a\} \end{aligned} \quad (5-13)$$

The first state space representation (case 4a), given by Equation (5-11), includes one goal and an action-history one larger than the minimum length necessary for goal-completion

to be recognized. It is the least efficient representation, with $n_s=2^1*8^8=33,554,432$ states, and follows the Case #3 example of deterministic domain representation discussed above in Chapter 5.4.1.3. However, it is the most straightforward and human-readable representation. We show how this is encoded, but we do not run an example to completion, in part due to memory constraints and largely due to the timescales involved in calculating even the policy outcomes. Table 5-8 shows the projected memory requirements. From the simulation results below (Chapter 5.4.2.4), we estimate a calculation time for the policy's value iteration stage alone of at least 56 minutes *per iteration*.

To allow the calculation and use of the reduced transition probability tensor, three internal lookup tables are necessary. There is a direct tradeoff between calculation time and memory space for the reduced tensor implementation here: these tables only need be computed once per state space representation for varying probability and reward parameter sets, but take a nontrivial amount of memory and time to compute. Memory requirements could theoretically be reduced further by functionally calculating these mappings during runtime instead; however, this would likely exponentially increase the time necessary to calculate the reward and transition probabilities and for the value iteration procedure to complete. It is also noteworthy that the reduced transition probability tensor requires only half a gigabyte of memory for 1 byte per data element (n_s*n_a*2). Recasting the problem using a full-sized transition probability tensor would have required $n_s*n_s*n_a=33,554,432^2*8=9.0072*10^{15}$, or upwards of 9 million gigabytes of memory for 1 byte per data element, instead, illustrating savings in the reduced transition probability functional model.

Because of the memory constraints and runtime issues, it is prudent to reduce the size of the state space whenever possible, decomposing our mission into smaller MDPs. Recall that our architecture in Chapter 4 has already taken this into account – in a space application, an astronaut would be expected to touch base with a mission operator at mission control whenever he/she moves on to a new set of activities. The mission operator they are talking to will note these changes and can dictate to the robot when to switch between policies. Thus, many small MDPs could be made to cover portions of the

astronaut's workday, instead of attempting to use one large MDP to cover all circumstances.

Table 5-8: Memory requirements, case 4a (1 goal, $n_h=8$)		
Item	Size (number of elements)	Approx. memory allocation needed
Reduced transition probability tensor (binary-valued)	$n_s * n_a * 2$ = 33,554,432 * 8 * 2 = 536,870,912	4.3 GB, default (double, 64 bits) minimum: 0.54 GB (uint8, 8 bits)
Mapping of i to state attributes (for given s^i) (integer-valued)	$n_s * (n_g + n_f + n_h)$ = 33,554,432 * (1 + 0 + 8) = 301,989,888	2.4 GB, default (double, 64 bits) minimum: 302 MB (uint8, 8 bits)
Mapping of state attributes to i of s^i (for given vector of state attributes) (integer-valued)	(range of g_z) ^{n_g} * (range of f_z) ^{n_f} * (range of a_k) ^{n_h} = $(2)^1 * (2)^0 * (8)^8$ = 33,554,432	272 MB, default (double, 64 bits) minimum: 34 MB (uint8, 8 bits)
Mapping of s^i to s^j corresponding to values in reduced probability tensor (integer-valued)	$n_s * 2 = 33,554,432 * 2$ = 67,108,864	536 MB, default (double, 64 bits) minimum: 67 MB (uint8, 8 bits)
Reward vector (binary-valued)	$n_s * 1 = 33,554,432$	270 MB, default (double, 64 bits) minimum: 34 MB, (uint8, 8 bits)
Value vector (floating point)	$n_s * 1 = 33,554,432$	270 MB, default (double, 64 bit) minimum: 135 MB (single, 32 bit)
Policy vector (integer-valued)	$n_s * 1 = 33,554,432$	270 MB, default (double, 64 bits) minimum: 34 MB, (uint8, 8 bits)
Matlab instance (64-bit 2012a under 64-bit Windows 7)	--	718 MB
Totals: (max. 6 GB of 8 GB free for process)	1,040,187,392	9.036 GB, default minimum: 1.864 GB

The second state space representation (case 4b), given by Equation (5-12), includes four goals and an action-history of size 4 and follows the Case #1 example from Chapter 5.4.1.1. It only tracks the goal progress of the entire set of screws being removed, and the action-history can be used to explicitly track the order in which screws have been removed. This is of manageable computational size, with $n_s = 2^4 * 8^4 = 65,536$ states.

The third state space representation (case 4c), given by Equation (5-13), includes seven goals and no action-history and follows the Case #1 example from Chapter 5.4.1.1. It tracks the goal progress of the screw positions overall; but, with no action-history, we cannot tell what the explicit ordering of the screw removal process is at every point in time (e.g., if $\{g_{31}^i, g_{32}^i, g_{33}^i, g_{34}^i\} = \{1,0,0,0\}$, then we know the first screw was removed first, but if $\{g_{31}^i, g_{32}^i, g_{33}^i, g_{34}^i\} = \{1,1,0,0\}$, then we don't know whether it was the first screw or the second screw that was removed first). This is the most compact MDP representation, with $n_s=2^7=128$ states.

Both cases 4b and 4c could be represented using either Case #1 or Case #2 of the deterministic domain representation. We use Case #1 here.

5.4.2.2 Transition Probability Function

Transition probabilities for this case study are given in Equation (5-14), and are dependent on aspects of the current state as well as the predicted next action choice a_k :

$$T(s^i, a_k, s^j) = \prod_{z=1}^{n_g} p_z(g_z^j | s^i, a_k) \quad (5-14)$$

Note that the form is derived from Equation (5-4) because dependencies exist for goals with precedence constraints due to the script. For the case study in this section, there are at most only 2 possible transitions for each state given a choice of a_k : either stay the same or change; the no-op action is not included in this state space. All transitions for each goal state have either $p=0$ or $p=1$. Because we may need to look at all the goal states for some cases to determine whether goal transition is possible, we include s^i , not only g_z^i and A^i . However, because our transition probabilities are restricted to 0 or 1, we can still use the equation in the form of Equation (5-14) in this deterministic case because any combination of impossible states will have $p=0$ in at least one multiplicand.

As described above, the transition probability is dependent upon the past action history A^i and the predicted next action choice of the human a_k . There are no explicit mappings between a goal state and an action-choice, but the probabilities are constrained to the possible outcomes of the action-choices for goal-completion to simplify the number of

parameters to optimize. By leveraging this information, we do not need to specify the full tensor, only the possible transitions for each set of actions for each state, thereby reducing the computational complexity. This makes our deterministic probability tensor for each case of size $n_s \times n_a \times 2$ – that is, the number of states (combinatorial set) times twice the number of actions (as there are at maximum only two possible outcomes for each action).

For case 4a (Equation (5-11)), the only possible transition leading to goal completion is an action-history (plus a_k) with a sequence that meets the following criteria:

- *retrieve_screwdriver* (a_2) can only occur after *toolbox_retrieval* (a_1)
- any action *remove_screw_from_panel_position_X* (a_{3X}) can only occur after *retrieve_screwdriver* (a_2) (and before *set_down_screwdriver* (a_4))
- *remove_panel* (a_5) can only occur after all *remove_screw_from_panel_position_X* (a_{3X}) actions and the *set_down_screwdriver* (a_4) action

An explicit choice of ordering for the a_{3X} actions may also be enforced at the scripter's discretion.

For case 4b (Equation (5-12)), the constraints are dependent on the goal state and action-history as follows (also shown in Table 5-9):

- *close_positioning_of_toolbox* (g_2) must be true for *retrieve_screwdriver* (a_2) to be able to transition *holding_screwdriver* (g_4) to true
- *holding_screwdriver* (g_4) must be true and *retrieve_screwdriver* (a_2) must come before all other *remove_screw_from_panel_position_X* (a_{3X}) actions in the action-history and a_k , for a_k to be able to transition *all_screws_removed* (g_3) to true
- *holding_screwdriver* (g_4) must be true for *set_down_screwdriver* (a_4) to be able to transition *holding_screwdriver* (g_4) to false
- *all_screws_removed* (g_3) must be true and *holding_screwdriver* (g_4) must be false for *remove_panel* (a_5) to be able to transition *panel_removed* (g_1) to true

Table 5-9: MDP policy illustrating action-history use, case 4b (4 goals, $n_h=4$)		
(x denotes don't care)		
State $s^i = \{g_1^i, g_2^i, g_3^i, g_4^i, a_1^i, \dots, a_4^i\}$	Policy action a_k ($T(s^i, a_k, s^j)=1$ to new state)	New state $s^j = \{g_1^j, g_2^j, g_3^j, g_4^j, a_1^j, \dots, a_4^j\}$
{x 1 x x x x x}	retrieve_screwdriver (a_2)	{x 1 x 1 x x x x}
{x x x 1 a_2 a_{3m} a_{3n} a_{3o} } a_{3m}, a_{3n}, a_{3o} $\in \{a_{31}, a_{32}, a_{33}, a_{34}\}$	remove_screw_from_panel _position_X (a_{3p}), $a_{3p} \in \{a_{31}, a_{32}, a_{33}, a_{34}\}$ $a_{3m} \neq a_{3n} \neq a_{3o} \neq a_{3p}$	{x x 1 1 x x x x}
{x x x 1 x x x x}	set_down_screwdriver (a_4)	{x x x 0 x x x x}
{x x 1 0 x x x x}	remove_panel (a_5)	{1 x 1 0 x x x x}

For case 4c (Equation (5-13)), the constraints are dependent on the goal state as follows (also shown in Table 5-10):

- *close_positioning_of_toolbox* (g_2) must be true for *retrieve_screwdriver* (a_2) to be able to transition *holding_screwdriver* (g_4) to true
- *holding_screwdriver* (g_4) must be true for *set_down_screwdriver* (a_4) to be able to transition *holding_screwdriver* (g_4) to false
- all *screw_in_position_X_removed* (g_{3X} , $X=\{1,2,3,4\}$) goals must be true and *holding_screwdriver* (g_4) must be false for *remove_panel* (a_5) to be able to transition *panel_removed* (g_1) to true

Table 5-10: MDP policy illustrating action-history use, case 4c (7 goals, $n_h=0$)		
(x denotes don't care)		
State $s^i = \left\{ \begin{matrix} g_1^i, g_2^i, g_{31}^i, g_{32}^i, g_{33}^i \\ g_{34}^i, g_4^i \end{matrix} \right\}$	Policy action a_k ($T(s^i, a_k, s^j)=1$ to new state)	New state $s^j = \left\{ \begin{matrix} g_1^j, g_2^j, g_{31}^j, g_{32}^j, g_{33}^j \\ g_{34}^j, g_4^j \end{matrix} \right\}$
{x 1 x x x x x}	retrieve_screwdriver (a_2)	{x 1 x x x x 1}
{x x x x x x 1}	set_down_screwdriver (a_4)	{x x x x x x 0}
{x x 1 1 1 1 0}	remove_panel (a_5)	{1 x 1 1 1 1 0}

Action-history progression constraints are similar for all policies. Using case 4b as an example: $A^i = \{a_1^i, a_2^i, a_3^i, a_4^i\}$ with a_k as the action-choice becomes $A^j = \{a_2^j, a_3^j, a_4^j, a_k^j\}$.

5.4.2.3 Rewards

The reward functions for our case study are consistent with the form shown in Equation (5-9) and Equation (5-10):

$$R(s^i) = \sum_{z=1}^{n_g} \alpha_z g_z^i \quad (5-15)$$

For all three cases (4a, 4b, 4c) our reward function is the same:

$$R(s^i) = \alpha_1 * g_1^i \quad (5-16)$$

We include goals in cases 4b and 4c that can potentially result in partially-ordered plans (multiple scripts that could work). Using the Case #1 example in Chapter 5.4.1.1 as a guide, only α_1 is set to a nonzero value, so that only g_1^i has an impact. Thus, for case 4a, g_1^i is the only goal to achieve. Because there are no competing objectives we can set $\alpha_1 = 1$.

5.4.2.4 Simulation Results

We encoded the states, actions, transition probability functions, and reward functions for cases 4a, 4b, and 4c as-given above. On a quad-core AMD processor laptop with 8GB memory running Matlab R2012a, case 4c took approximately 3-4 seconds total runtime. The infinite horizon value iteration solver completed the necessary 124 iterations in 2-3 seconds. Case 4b took approximately 11 minutes total runtime. The infinite horizon value iteration solver completed the necessary 23 iterations in a little under 3½ minutes.

For cases 4b and 4c, the results were as-expected for the script sequencing, as shown below:

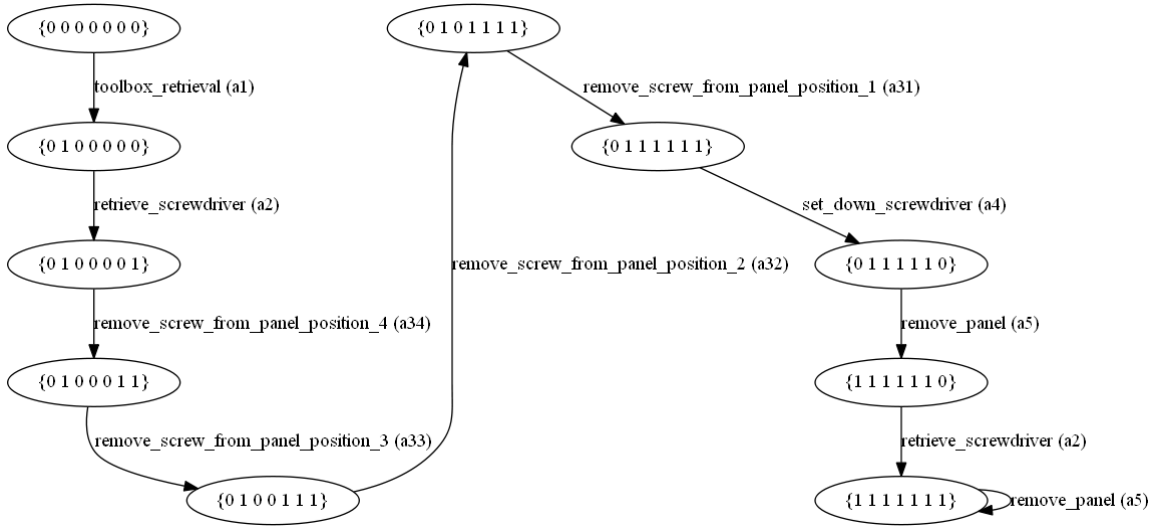


Figure 5-2: State evolution for the optimal MDP policy, case 4c (7 goals, $n_h=0$), starting from $s^i = \{\text{no goals set}\}$, for $s^i = \{g_1^i, g_2^i, g_{31}^i, g_{32}^i, g_{33}^i, g_{34}^i, g_4^i\}$



Figure 5-3: State evolution for the optimal MDP policy, case 4b (4 goals, $n_h=4$), starting from $s^i = \{\text{no goals set, all actions in history } toolbox_retrieval (a_1)\}$, for $s^i = \{g_1^i, g_2^i, g_3^i, g_4^i, a_1^i, a_2^i, a_3^i, a_4^i\}$

In both examples, the scripts are initially followed from the initial state, $s^i = \{\text{no goals set}\}$. Because we do not include no-op in this state space, for the state to become ‘stable’, the policy moves into an absorbing state, which includes holding the screwdriver, before attempting to remove the panel again and again to no further effect. Including a no-op action with slightly lower cost to execute in the state space would resolve this issue.

Considering off-nominal states for case 4b, with $s^i = \{\text{panel_removed, close_positioning_of_toolbox, all_screws_removed, holding_screwdriver, \{action-history\}\}$, the policy chose action a_1 for every (starting) state $s^i = \{0,0,0,0,x,x,x,x\}$ (with x

being any action in the action sequence), and chose action a_2 from every follow-on state $s^i=\{0,1,0,0,x,x,x,x\}$ once the toolbox was close enough to pick up the screwdriver. Screw removal can then be performed in any order, resulting in a partial ordering. For case 4b, the screw removal actions in any order are able to achieve the goal represented by the transition from state $s^i=\{0,1,0,1,x,x,x,x\}$ to $s^i=\{0,1,1,1,x,x,x,x\}$. The next action chosen from state $s^i=\{0,1,1,1,x,x,x,x\}$ is always *set_down_screwdriver* action a_5 and transitions the state to $s^i=\{0,1,1,0,x,x,x,x\}$. From state $s^i=\{0,1,1,0,x,x,x,x\}$ the next selected action is *remove_panel* to $s^i=\{1,1,1,0,x,x,x,x\}$. All sequences worked as intended, following the script within the constraints given.

For case 4c, with $s^i=\{panel_removed, close_positioning_of_toolbox, screw_in_position_1_removed, screw_in_position_2_removed, screw_in_position_3_removed, screw_in_position_4_removed, holding_screwdriver\}$, the policy is also able to always work its way from any state where one or more screws need to be removed $s^i=\{0,1,x,x,x,x,1\}$ to state $s^i=\{0,1,1,1,1,1,1\}$ where the screws are all removed. All other policy-enacted transitions follow the strict set sequence as shown in Figure 5-2, as intended.

5.4.3 Stochastic HIP modeling

While the above examples demonstrate use of the MDP for deterministic scripts, the MDP's strength is in its application to uncertain systems. Uncertainty may be present in human action *selection* (model internal match where the human decides to do something else), *completion* (ability-based, including effects of distraction), and/or *outcome* (external/environmental). With conditional independence, each probability is p_{kz} for the action to have no impact on the goal flag (goal remains the same), or $(1-p_{kz})$ for an action impacting the goal flag. With dependence, structures such as Bayes nets or probability tables may be used.

5.4.4 Case Study #2 – IVA scenario, stochastic system

In this section, an example HIP MDP is developed for a simple IVA scenario, one whose main task is to complete computer work but with uncertainty due to the insertion of eat/drink actions. For illustrative purposes, we assume that the observer always treats the in-progress action as 'unknown', implying that the observer tasked with action-

recognition from sensor data will provide each observed action to the MDP policy executor when recognized, and that the observer also can flag when that action is completed. This would be the case both when the observer recognizes the action early and when it merely observes the consequences of completing a particular action.

We use a domain model with mental concentration (computer work) and pick-and-place (eat/drink) tasks that would be appropriate for astronauts performing intravehicular activity (IVA) as well as humans in their homes on Earth. Our previous experiments described in Chapter 3 provide insight as to human behaviors in such an environment, and here we assume that our basic simulation and experimental results will translate to models of humans performing similar activities in IVA in space.

We define a base case to include a sporadic (interruptive) goal to press only one button. This initially simplifies the human choice preference to be between just blood sugar level/hydration level/work motivation (traditional goals) versus a high-priority button-pushing goal, with the latter objective able to override all other operations, depending on the parameters used in our reward and transition probability function formulation. This also is consistent with our assumption that only one high-priority goal will be active at a time, avoiding the need to carefully model relative priorities over an interruptive goal set.

This scenario parallels on-orbit EVA space-repair at a satellite electronics panel: ‘chip eating’ is a retrieval action, such as consumables that may need to be used to fix internal electronics; ‘soda drinking’ is a pick-and-place action, a simple analogue to the retrieval, use, and stowing of a screwdriver; ‘computer work’ is a cognition-intensive action, such as troubleshooting problems inside a panel through careful visual inspection; and ‘button pushing’ is a task of overriding importance, a time-critical task like noticing and grabbing a toolkit before it floats away.

5.4.4.1 States and Actions

For this case study, we model the above simple inter-vehicle activity (IVA) scenario including the actions of eating (1), drinking (2), interacting with a computer (3), high-priority button-pushing (4), and no-op (5). These five tasks are required for astronauts in IVA as well as for humans on Earth. The state has three mission goals of raise blood

sugar level (1), raise hydration level (2), and complete a general mission-oriented work-effort (3), and one high-priority goal of button-inactive (1) that indicates that a button needs to be pushed (corresponding to a safety-critical mission task that might need to be completed). We have conducted previous human subject experiments of these tasks with a safe robotic manipulator arm that confirm the feasibility of such a shared workspace (see Chapter 3).

Table 5-11 and Table 5-12 describe the human’s actions, goals, and the meanings of the variable status used for our domain. The actions are integers from 1 to 5, corresponding to the labels given above (see Table 5-11).

Table 5-11: Domain representation of actions a_k^i

Discrete Value	Corresponding Action
1	eat_chips (a_1)
2	drink_soda (a_2)
3	computer_work (a_3)
4	push_button (a_4)
5	no_op (a_5)

The mission goals and high-priority goal are binary-valued, with 0 corresponding to incomplete and 1 corresponding to complete (see Table 5-12). A 0 value indicates that a high-priority goal needs to be satisfied (a high cost is incurred for remaining in that state), and a 1 indicates that the button is inactive and does not need to be pushed again. A modest reward is offered for remaining in this safe state.

Table 5-12: Domain representation of goal-objectives

Goal Obj.	Values	Corresponding Action
g_1^i	{0,1}	?blood_sugar_level? (nominal = 1) (g_1)
g_2^i	{0,1}	?hydration_level? (nominal=1) (g_2)
g_3^i	{0,1}	?work_motivation? (lazy/done=1) (g_3)
f_1^i	{0,1}	?button_1_inactive? (inactive=1) (f_1)

Note that we do not explicitly differentiate between physical and mental tasks in our MDP representation, mixing actions such as computer work (math) with eating, drinking, and button-pushing.

For the case study above, there are two basic ways to define the state space:

$$\begin{aligned}
 & s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i, \dots, a_{n_h}^i\} \\
 \text{Case 5a: } & g_z^i \in \{0,1\}, f_z^i \in \{0,1\} \\
 & a_k^i \in \{1, \dots, n_a\}, n_a = 5, a_k \in \{1, \dots, n_a\}
 \end{aligned} \tag{5-17}$$

$$\begin{aligned}
 & s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\} \\
 \text{Case 5b: } & g_z^i \in \left\{0, \frac{1}{n_h+1}, \frac{2}{n_h+1}, \dots, 1\right\}, f_z^i \in \{0,1\} \\
 & n_a = 5, a_k \in \{1, \dots, n_a\}
 \end{aligned} \tag{5-18}$$

For the first state space representation (case 5a), and the value of n_h is constant for any particular MDP policy; the action history lengths explored in this case study range from $n_h=0$ to $n_h=3$. Note that the action $a_{n_h+1}^i$ is always “unknown” and thus not included for simplicity, as we are assuming in this case study that action-recognition cannot supply the in-progress action. For the second state space representation (case 5b), the mission goals are multi-valued, corresponding to n_h+2 terms, and no action-history is included – instead, n_h determines the number of divisions in the goal state; an example formulation where this might be most useful is in tracking a fuel-meter’s state.

Note that for the $n_h=0$ case, however, both case 5a and case 5b simplify to the following representation:

$$\begin{aligned}
 & s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\} \\
 & a_k \in \{1,2,3,4,5\} \\
 & g_z^i \in \{0,1\}, f_z^i \in \{0,1\}
 \end{aligned} \tag{5-19}$$

In all of our simulations, we use the case 5a representation. Equation (5-19) is used for the reward and transition probability tradeoffs, and Equation (5-17) is used for testing the effect of the action-history.

5.4.4.2 Transition Probability Function

Transition probabilities for this case study are given in Equation (5-20) below; goals are conditionally-independent from each other, but are dependent on all other aspects of the current state as well as the predicted next action choice a_k :

$$T(s^i, a_k, s^j) = p(f_1^j | f_1^i, a_k) * \prod_{z=1}^3 p(g_z^j | g_z^i, A^i, a_k) \quad (5-20)$$

For this case study, if a goal has already been completed, it stays in the absorbing state set with 100% probability (1 stays 1, 1 never transitions back to 0). If an action a_k does not impact a goal, then that goal stays in the same state with 100% probability. For convenience, we refer to a transition for a mission goal g_z^i due to a completing action a_k in the $n_h=0$ case as having probability p_{kz} in the no-change case from 0 to 0, and $(1-p_{kz})$ for transitioning from 0 to 1. We assume high-priority goals will complete with 100% probability if action a_k affects that goal, and 0% probability otherwise. We also assume no goal can ever transition back to active (0) once inactive (complete, 1).

For our case study there are at most only 10 possible transitions for each state, and the no-op action is included for use when no goal requires accomplishment. Figure 5-4 shows the transition system for work completion. The *push_button* and *no_op* actions each have only one possible next state with transition probability 1. Figure 5-5 shows the transition system associated with high-priority button deactivation states.

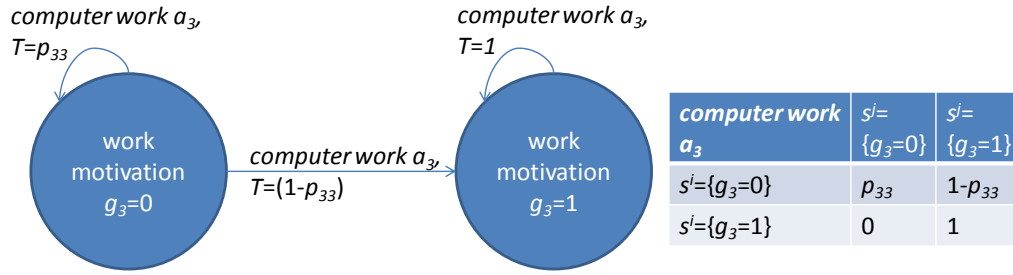


Figure 5-4: State Transition Diagram and transition matrix for work_motivation only, $n_h=0$

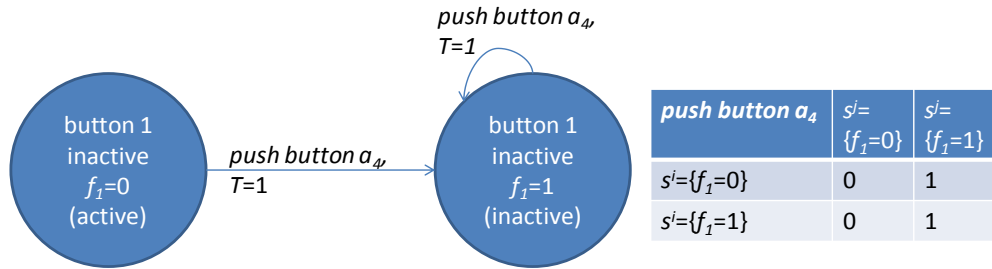


Figure 5-5: State Transition Diagram and transition matrix for button_1_inactive only, $n_h=0$

The *eat_chip* and problem-solving action (*computer_work*) have at most two outcomes each: staying in the same state (no change) or transitioning to corresponding goals being completed as shown in Figure 5-6 and Figure 5-4 respectively. In Figure 5-6, the *drink_soda* action has up to four possible outcomes, due to a coupling of the *drink_soda* action with both the *blood_sugar_level* and *hydration_level* goal objectives. The action of taking a drink, in this case, may raise a person's blood sugar level, if it is a sugary drink. In all case studies, we presume the drink is sugary, as was the Coke consumed during human subject experiments from Chapter 3.

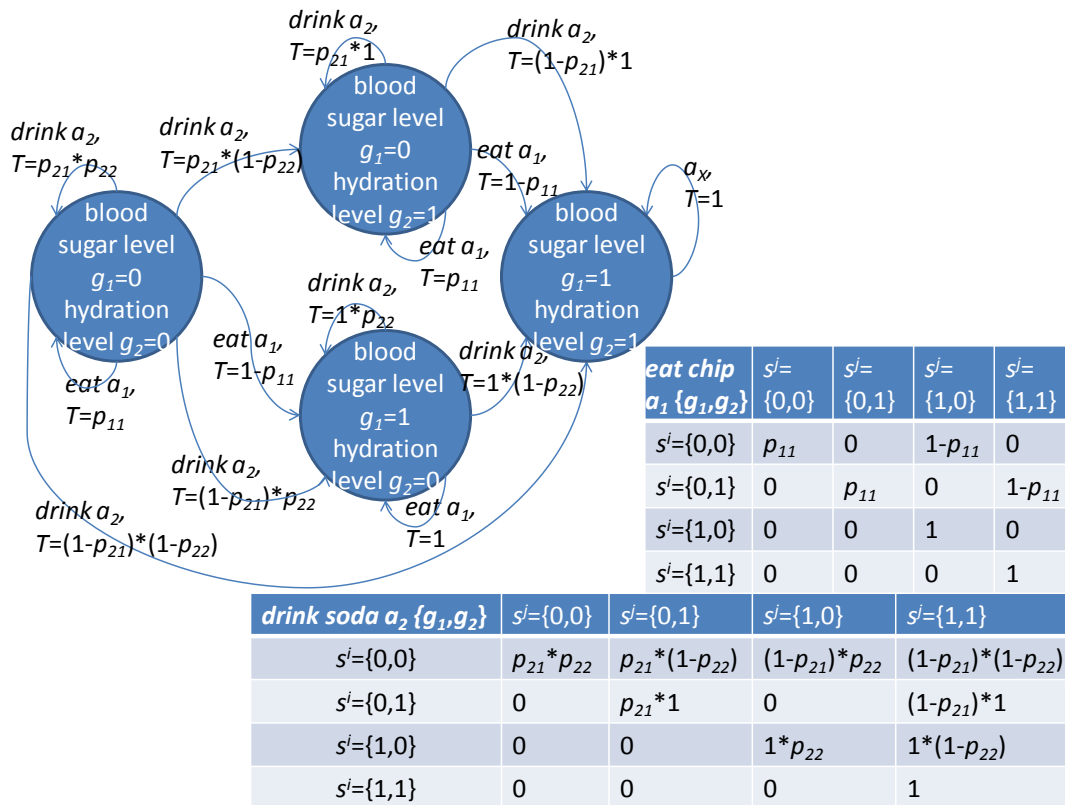


Figure 5-6: State Transition Diagram and transition matrices for blood_sugar_level and hydration_level only, $n_h=0$

A limited example of the general finite state machine diagram for the domain representation is given in Figure 5-7, with $n_h=1$. Recall that case 5a refers to the domain representation given by Equation (5-17) above.

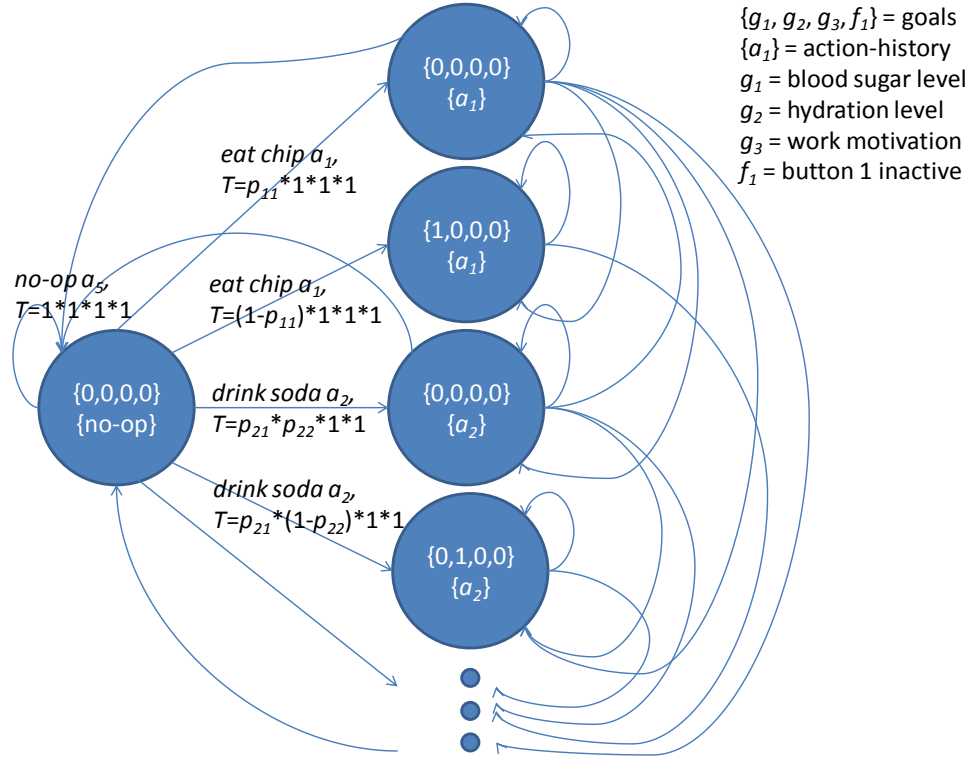


Figure 5-7: Finite State Machine Diagram for case 5a Representation, fully-connected (not all links labeled), $n_h=1$

An example of probability calculation for the goal of sating work motivation (by performing *computer_work*) is given in Equation (5-21) for an action-history of length $n_h=0$:

$$p(g_3^j | g_3^i, A^i, a_k) = (1 - p_{33}) * \frac{(a_k == a_3)}{1}, g_3^i = 0, g_3^j = 1 \quad (5-21)$$

Equation (5-22) gives an example with an action-history of length $n_h=3$:

$$p(g_3^j | g_3^i, A^i, a_k) = \lambda_{33} * \frac{(a_1^i == a_3) + (a_2^i == a_3) + (a_3^i == a_3) + (a_k == a_3)}{3+1}, g_3^i = 0, g_3^j = 1 \quad (5-22)$$

Here, we look to see how many times the *computer_work* action a_3 appears in the history. Every time a match is found, a “1” is tallied; we sum the number of times the action appears. We divide by the number of terms to normalize. We then multiply by a

weighting factor in the range [0 1] to scale this value. If $\lambda_{33} = 1$, then finishing computer work tasks four times in a row will model *work_motivation* as sated with 100% probability.

The general form of Equation (5-22) that we use for this case study is given in Equation (5-23):

$$(1 - p_{kz}) = p(g_z^j = 1 | g_z^i = 0, A^i, a_k) = \frac{\left(\sum_{x=1}^{n_a} \sum_{m=1}^{n_h} \lambda_{xz}(a_m == a_x) \right) + \lambda_{kz}(a_k == a_k)}{n_h + 1} \quad (5-23)$$

$$p_{kz} = p(g_z^j = 0 | g_z^i = 0, A^i, a_k) = 1 - p(g_z^j = 1 | g_z^i = 0, A^i, a_k)$$

For $n_h=0$, this equation simplifies to having a direct relationship with weight λ_{kz} :

$$(1 - p_{kz}) = p(g_z^j = 1 | g_z^i = 0, A^i, a_k) = \frac{0 + \lambda_{kz}(a_k == a_k)}{0 + 1} = \lambda_{kz} \quad (5-24)$$

$$p_{kz} = p(g_z^j = 0 | g_z^i = 0, A^i, a_k) = 1 - p(g_z^j = 1 | g_z^i = 0, A^i, a_k) = 1 - \lambda_{kz}$$

The summation term is used to check each term in the action-history in order; we add λ_{xz} to the numerator if that particular action a_x at history location m has some impact on goal g_z transitioning from 0 to 1. The impact λ_{kz} of choosing action a_k is then added, and the numerator divided by the number of total possible terms in the numerator.

This formulation gives equal weight to each action in the action history and action a_k . Due to the normalizing term in the denominator of Equation (5-23), the transition probability cannot be higher than the largest λ_{kz} for a goal g_z^i ; the highest probability of transition for a particular goal occurs when every action in the action-history is the same as a_k and a_k has the highest likelihood of transition for that goal.

As described above, the transition probability is dependent upon the past action history A^i and the predicted next action choice of the human a_k . There are no explicit mappings between a goal state and an action-choice, but the probabilities are constrained to the known valid action-choices for goal-completion to simplify the number of parameters to

optimize. By leveraging this information, we do not need to specify the full tensor, thereby reducing the computational complexity.

For the case study, we assume that a goal g_z^i may transition from 0 to 1 only if the action-choice a_k impacts that goal (λ_{kz} is nonzero), and will not transition otherwise. In our models there are no delayed effects. Thus, to calculate that conditionally-independent goal's transition probability, when λ_{kz} is zero we use Equation (5-24) that effectively ignores the “history” term, and we use Equation (5-23) when λ_{kz} is nonzero for the multiplicand in Equation (5-20). For $p(f_1^j | f_1^i, a_k)$, we use Equation (5-6) to calculate the multiplicand, recalling that the in-progress action is always ‘unknown’.

As shown, our state-space model includes an action history as well as attributes describing sensed events. By reducing the mapping of objectives directly to actions, we are effectively tracking action-completion in our state.

5.4.4.3 Rewards

$$R(s^i) = \begin{cases} \alpha_1 g_1^i + \alpha_2 g_2^i + \alpha_3 g_3^i + \beta_1 & \text{if } f_j^i = 1 \\ \alpha_1 g_1^i + \alpha_2 g_2^i + \alpha_3 g_3^i - k_1 * \beta_1 & \text{if } f_j^i = 0 \end{cases} \quad (5-25)$$

The above reward function for our case study is consistent with the form shown in Equation (5-9) and Equation (5-10). To equally reward the completion of all goals and an inactive high-priority goal, we set all weighting factors to 1. k_1 is then set to a large positive constant that prioritizes completion of the high-priority goal. In this example so long as $(k_1 * \beta_1) > 3$ the MDP will prioritize high-priority task completion above any mission-related action, even if such an action may contribute to the completion of multiple mission goals.

We tested weights in the range of [0 1] with a change in weight of either $\Delta = 0.25$ or $\Delta = 0.10$.

5.4.4.4 Simulation Results

For the stochastic case, we use illustrative examples to evaluate the impact of reward weightings on the policy, the impact of the transition probabilities on the policy, and the impact of action-history length on the policy.

Changes in reward weighting should impact policy output in accordance with astronaut preferences. We test changes in policy that occur when we trade off weights between:

- unconnected independent nominal goals (*blood_sugar_level* and *work_motivation*)
- independent nominal goals impacted by more than one action (*blood_sugar_level* and *hydration_level*)
- nominal goal impacted by more than one action versus ‘high-priority’ goal (*blood_sugar_level* versus *button_1_inactive*)
- nominal goal impacted by only one action versus ‘high-priority’ goal (*work_motivation* versus *button_1_inactive*)

High-priority goals differ from nominal goals in that they effectively “override” the completion of other goals, in the majority of cases where the (nominal) mission goal weighting terms are much smaller than cost of not fulfilling high-priority goals.

To test the impact of these weightings, we fix the transition probabilities, set $n_h=0$ (no action-history), and zero the reward weights for all non-tested mission-goals. For cases not testing the high-priority goal of button pushing explicitly, we set $\beta_1=1$, $k_I=4$.

The transition probability function is from Equation (5-24) – the probability p_{kz} that action a_k does not transition goal g_z^i from 0 to 1. We use $p_{11}=0.25$, $p_{21}=0.75$, $p_{22}=0.50$, $p_{33}=0.25$, and $p_{44}=0.00$ for the reward weight examples, and set $p_{5X}=1$, indicating that no-op action a_5 never changes the state. All other p_{kz} are set to 1. Once a goal flag becomes 1 it has reached an absorbing state: transitions from $g_z^i=1$ to $g_z^j=1$ are always $p=1$, and transitions from $g_z^i=1$ to $g_z^j=0$ are always $p=0$. For the $n_h=0$ cases, simulation time using the same computing power as in Chapter 5.4.2.4 took a little more than half-a-

second total runtime. The infinite horizon value iteration solver completed after ~220-250 iterations in 0.20-0.30 seconds.

Table 5-13 and Table 5-14 show the impact of reward weightings on policy outcomes. Table 5-13 shows the impact on policy outcome for tradeoffs between the *blood_sugar_level* and *work_motivation* goals.

Table 5-13: Impact of reward weightings, eat_chip (a_1) / blood_sugar_level (g_1) vs. computer_work (a_3) / work_motivation (g_3)						
State Features				Policy action for reward weights 1=eat, 2=drink, 3=work, 4=button press, 5=no-op (constant weights: $\alpha_2=0.00$, $\beta_1=1.00$, $k_1=4$)		
g_1	g_2	g_3	f_1	$\alpha_1=0.50$ $\alpha_3=\mathbf{0.01}$	$\alpha_1=0.50$ $\alpha_3=\mathbf{0.50}$	$\alpha_1=0.50$ $\alpha_3=\mathbf{1.00}$
0	0	0	1	1 (eat)	3 (work)	3 (work)
0	0	1	1	1 (eat)	1 (eat)	1 (eat)
0	1	0	1	1 (eat)	3 (work)	3 (work)
0	1	1	1	1 (eat)	1 (eat)	1 (eat)
1	0	0	1	3 (work)	3 (work)	3 (work)
1	0	1	1	5 (no-op)	5 (no-op)	5 (no-op)
1	1	0	1	3 (work)	3 (work)	3 (work)
1	1	1	1	5 (no-op)	5 (no-op)	5 (no-op)
x	x	x	0	4 (button)	4 (button)	4 (button)

In Table 5-13, near the {0.50,0.50} weight set, there is a tradeoff for the transition probabilities given. While $\alpha_1=0.50$ and $\alpha_3<0.50$, eat is always chosen when $g_1=0$; once $\alpha_3\geq 0.50$, *computer_work* is always chosen when $g_3=0$; these cases are consistent in their optimal action choice. Figure 5-8 and Figure 5-9 show the associated state progression tree.

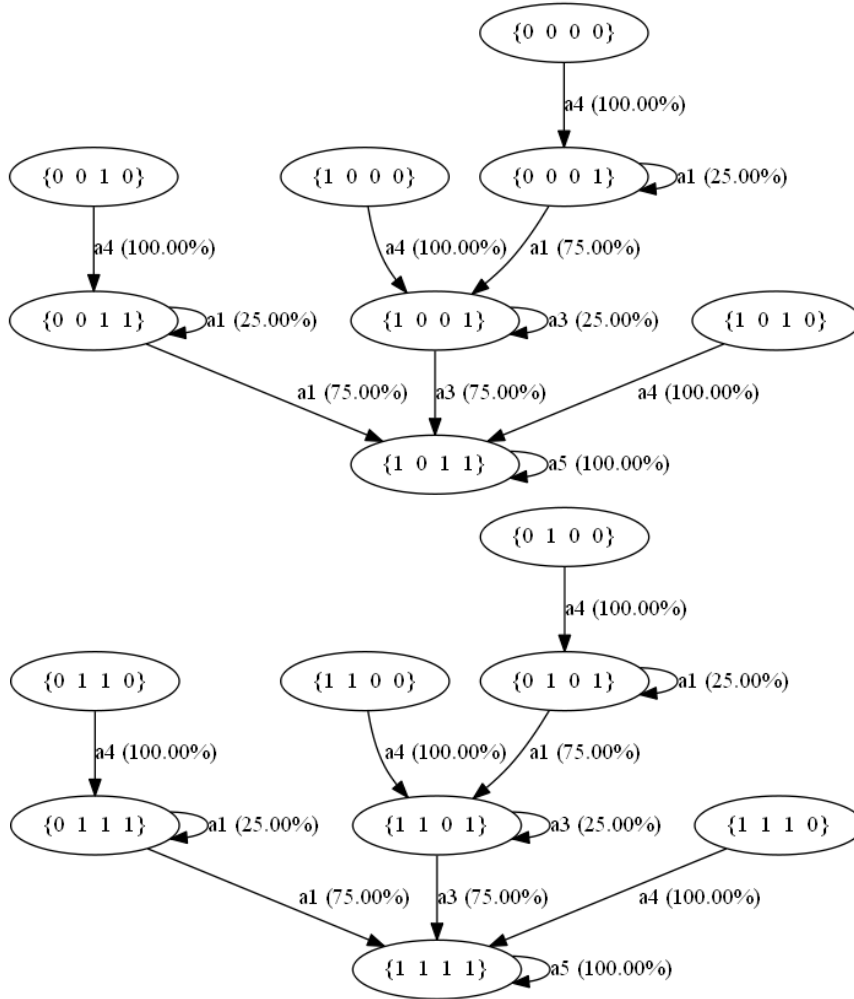


Figure 5-8: State/policy-action progression outcomes, column 1 from Table 5-13
 (blood_sugar_level g_1 vs. work_motivation g_3), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

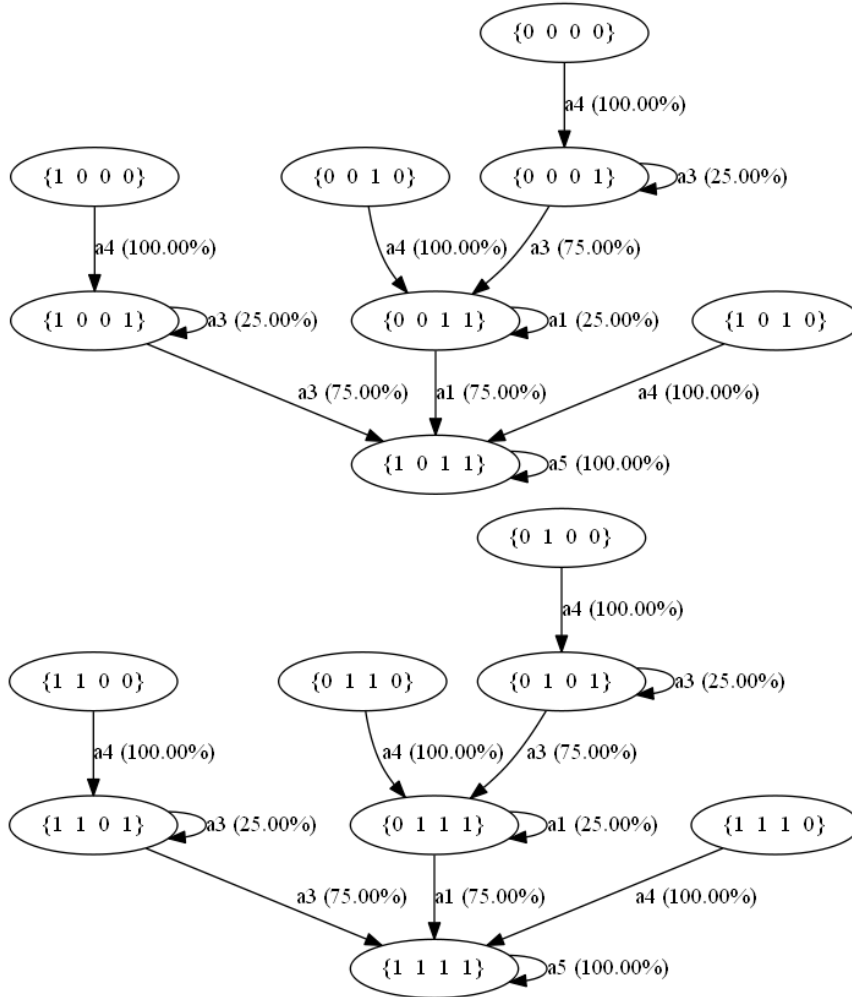


Figure 5-9: State/policy-action progression outcomes, column 2 from Table 5-13
 (blood_sugar_level g_1 vs. work_motivation g_3), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

Table 5-14: Impact of reward weightings, eat_chip (a_1) / blood_sugar_level (g_1) vs. drink_soda (a_2) / hydration_level (g_2)						
State Features				Policy action for reward weights 1=eat, 2=drink, 3=work, 4=button press, 5=no-op (constant weights: $\alpha_1=0.00$, $\beta_1=1.00$, $k_I=4$)		
g_1	g_2	g_3	f_I	$\alpha_1=0.50$ $\alpha_2=\mathbf{0.01}$	$\alpha_1=0.50$ $\alpha_2=\mathbf{0.41}$	$\alpha_1=0.50$ $\alpha_2=\mathbf{1.00}$
0	0	0	1	1 (eat)	2 (drink)	2 (drink)
0	0	1	1	1 (eat)	2 (drink)	2 (drink)
0	1	0	1	1 (eat)	1 (eat)	1 (eat)
0	1	1	1	1 (eat)	1 (eat)	1 (eat)
1	0	0	1	2 (drink)	2 (drink)	2 (drink)
1	0	1	1	2 (drink)	2 (drink)	2 (drink)
1	1	0	1	5 (no-op)	5 (no-op)	5 (no-op)
1	1	1	1	5 (no-op)	5 (no-op)	5 (no-op)
x	x	x	0	4 (button)	4 (button)	4 (button)

Table 5-14 shows the impact on policy outcome for tradeoffs between the *blood_sugar_level* and *hydration_level* goals. Notice that in Table 5-14 with the *drink_soda* action, the tradeoff weights are set to {0.50,0.41}. This is because the *drink_soda* choice does have some likelihood of sating/raising the *blood_sugar_level* goal on its own, and a lower but still nonzero probability of sating both those goals at once. Figure 5-10 and Figure 5-11 show the state progression tree.

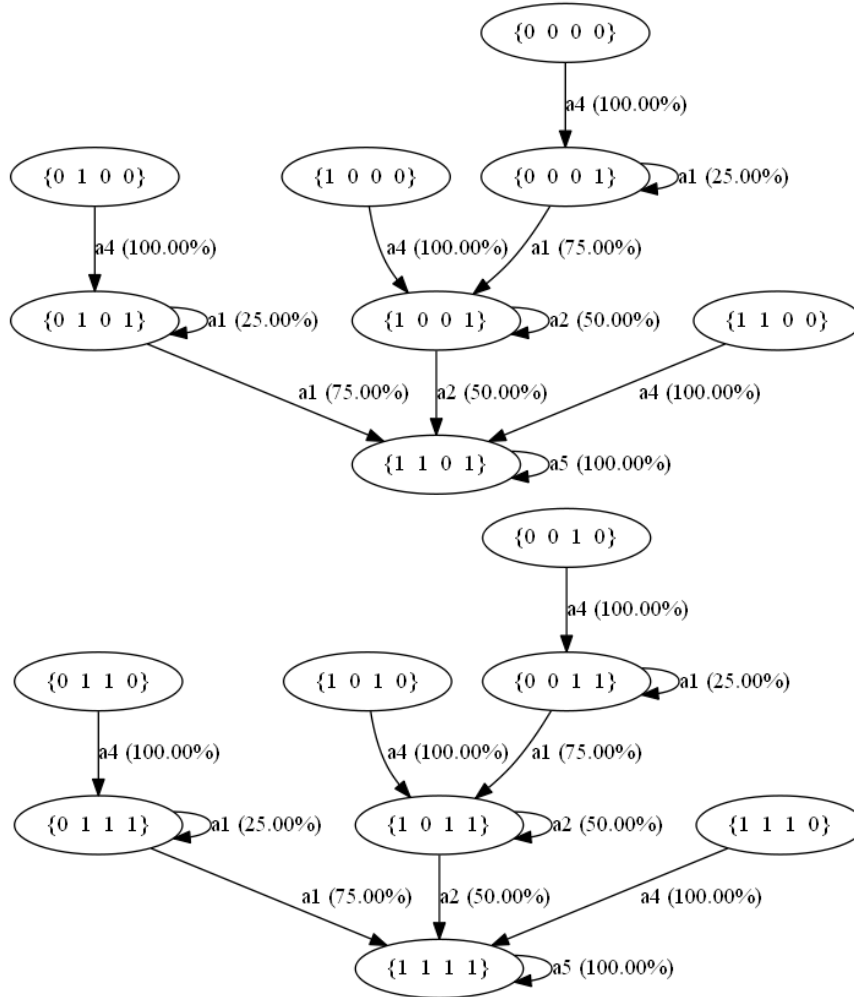


Figure 5-10: State/policy-action progression outcomes, column 1 from Table 5-14 (blood_sugar_level g_1 vs. hydration_level g_2), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

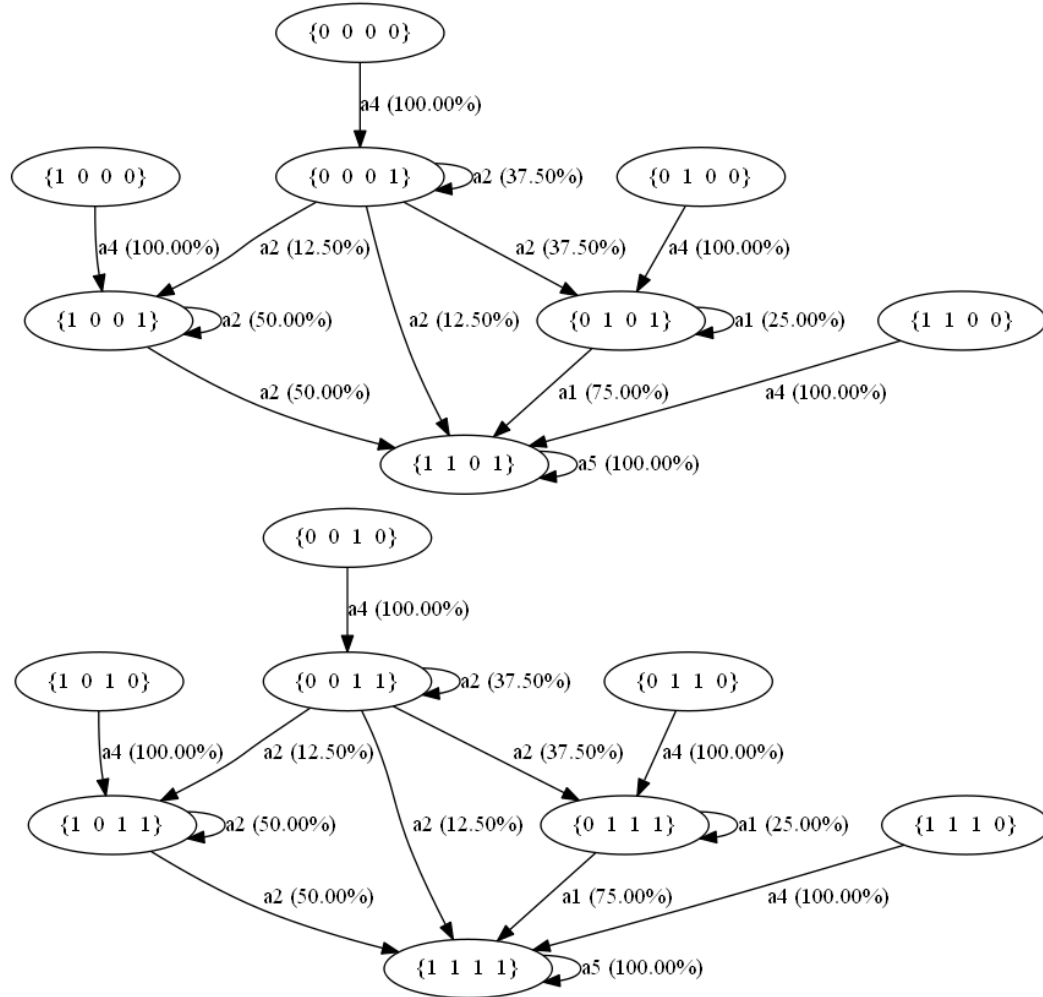


Figure 5-11: State/policy-action progression outcomes, column 2 from Table 5-14 (blood_sugar_level g_1 vs. hydration_level g_2), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

Table 5-15: Impact of reward weightings, eat_chip (a_1) / blood_sugar_level (g_1) vs. push_button (a_4) / button_1_inactive (f_1)						
State Features				Policy action for reward weights 1=eat, 2=drink, 3=work, 4=button press, 5=no-op (constant weights: $\alpha_2=0.00$, $\alpha_3=0.00$)		
g_1	g_2	g_3	f_1	$\alpha_1=0.50$ $\beta_1=0.01, k_1=4$	$\alpha_1=0.50$ $\beta_1=0.075, k_1=4$	$\alpha_1=0.50$ $\beta_1=1, k_1=4$
0	0	0	0	1 (eat)	4 (button)	4 (button)
0	0	1	0	1 (eat)	4 (button)	4 (button)
0	1	0	0	1 (eat)	4 (button)	4 (button)
0	1	1	0	1 (eat)	4 (button)	4 (button)
1	0	0	0	4 (button)	4 (button)	4 (button)
1	0	1	0	4 (button)	4 (button)	4 (button)
1	1	0	0	4 (button)	4 (button)	4 (button)
1	1	1	0	4 (button)	4 (button)	4 (button)
0	0	0	1	1 (eat)	1 (eat)	1 (eat)
0	0	1	1	1 (eat)	1 (eat)	1 (eat)
0	1	0	1	1 (eat)	1 (eat)	1 (eat)
0	1	1	1	1 (eat)	1 (eat)	1 (eat)
1	0	0	1	5 (no-op)	5 (no-op)	5 (no-op)
1	0	1	1	5 (no-op)	5 (no-op)	5 (no-op)
1	1	0	1	5 (no-op)	5 (no-op)	5 (no-op)
1	1	1	1	5 (no-op)	5 (no-op)	5 (no-op)

For the reward tradeoffs done against the high-priority button goal, shown in Table 5-15 and Table 5-16, the tradeoff weighting is $\{0.50, \{0.075, 4\}\}$ for both. This is not surprising, given that the *eat_chip* and *computer_work* actions are both given the same likelihood of satiating the *blood_sugar_level* and *work_motivation* goals, respectively. For the tradeoff of satiating the *blood_sugar_level* goal in Table 5-15, *drink_soda* is never selected in the optimal policy (1) because there is no reward for that goal, and (2) because in all situations, the *drink_soda* action has lower likelihood of transitioning *blood_sugar_level* to being satiated than the *eat_chip* action does.

Table 5-16: Impact of reward weightings, computer_work (a_3) / work_motivation (g_3) vs. push_button (a_4) / button_1_inactive (f_1)						
State Features				Policy action for reward weights 1=eat, 2=drink, 3=work, 4=button press, 5=no-op (constant weights: $\alpha_1=0.00$, $\alpha_2=0.00$)		
g_1	g_2	g_3	f_1	$\alpha_3=0.50$ $\beta_1=0.01$, $k_I=4$	$\alpha_3=0.50$ $\beta_1=0.075$, $k_I=4$	$\alpha_3=0.50$ $\beta_1=1$, $k_I=4$
0	0	0	0	3 (work)	4 (button)	4 (button)
0	0	1	0	4 (button)	4 (button)	4 (button)
0	1	0	0	3 (work)	4 (button)	4 (button)
0	1	1	0	4 (button)	4 (button)	4 (button)
1	0	0	0	3 (work)	4 (button)	4 (button)
1	0	1	0	4 (button)	4 (button)	4 (button)
1	1	0	0	3 (work)	4 (button)	4 (button)
1	1	1	0	4 (button)	4 (button)	4 (button)
0	0	0	1	3 (work)	3 (work)	3 (work)
0	0	1	1	5 (no-op)	5 (no-op)	5 (no-op)
0	1	0	1	3 (work)	3 (work)	3 (work)
0	1	1	1	5 (no-op)	5 (no-op)	5 (no-op)
1	0	0	1	3 (work)	3 (work)	3 (work)
1	0	1	1	5 (no-op)	5 (no-op)	5 (no-op)
1	1	0	1	3 (work)	3 (work)	3 (work)
1	1	1	1	5 (no-op)	5 (no-op)	5 (no-op)

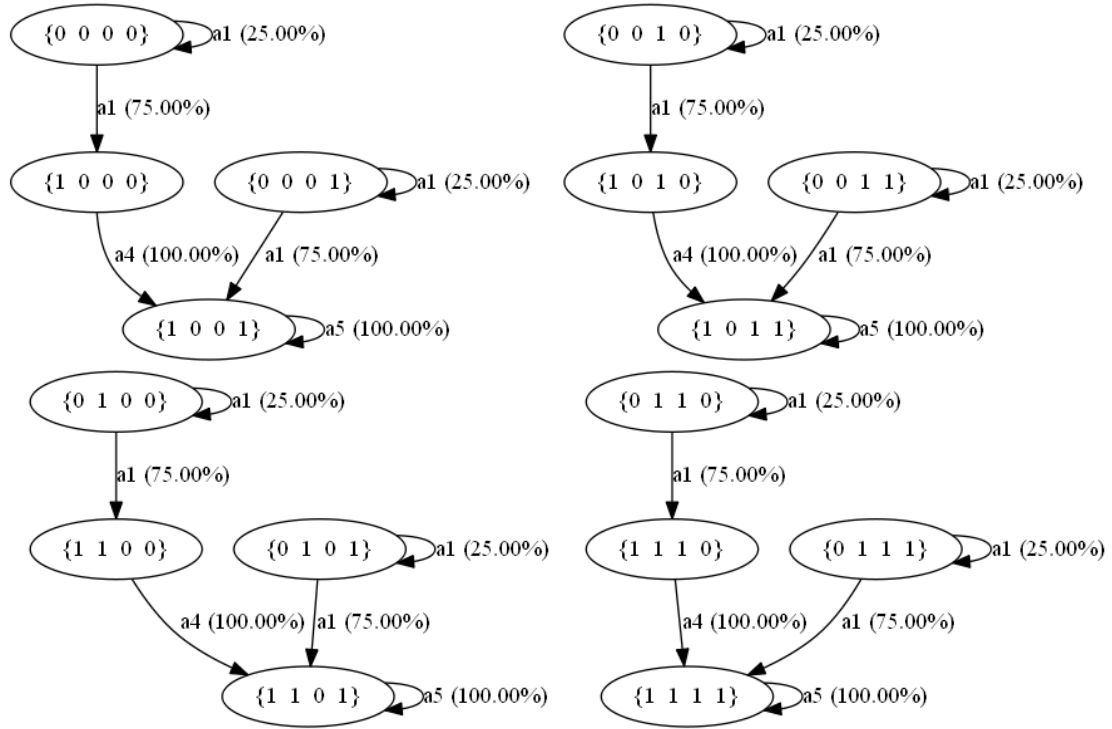


Figure 5-12: State/policy-action progression outcomes, column 1 from Table 5-15 (blood_sugar_level g_I vs. button_1_inactive f_I), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

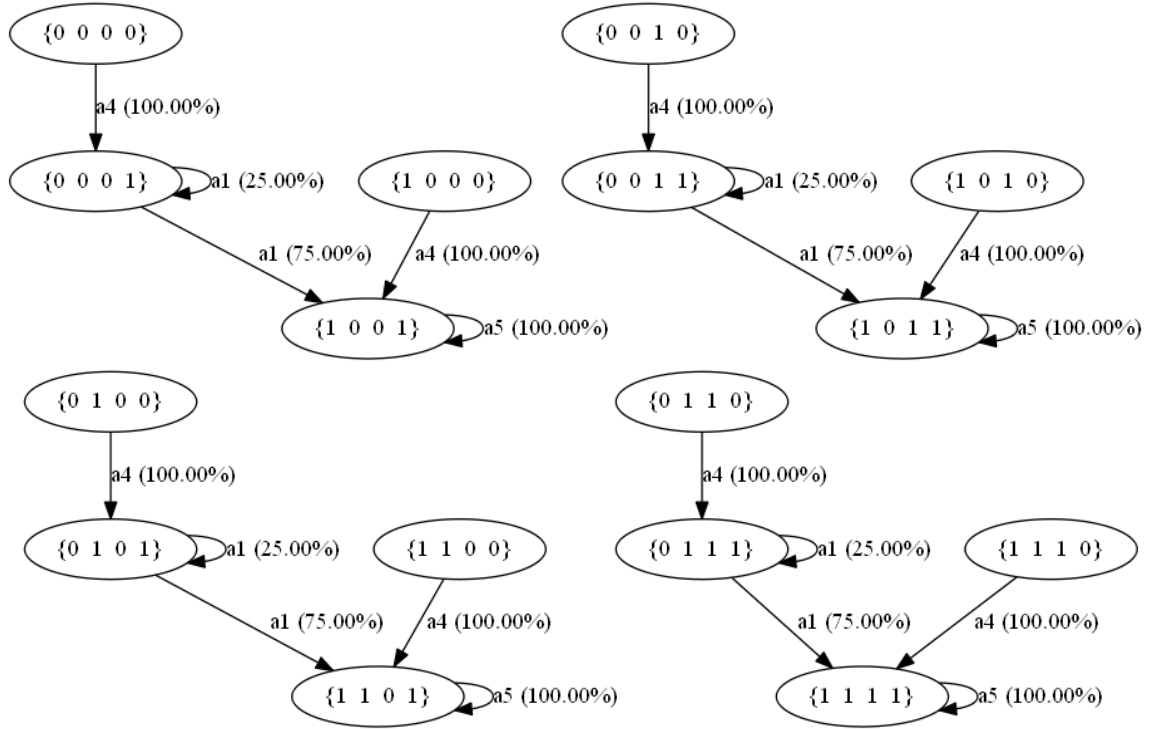


Figure 5-13: State/policy-action progression outcomes, column 2 from Table 5-15 (blood_sugar_level g_I vs. button_1_inactive f_I), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

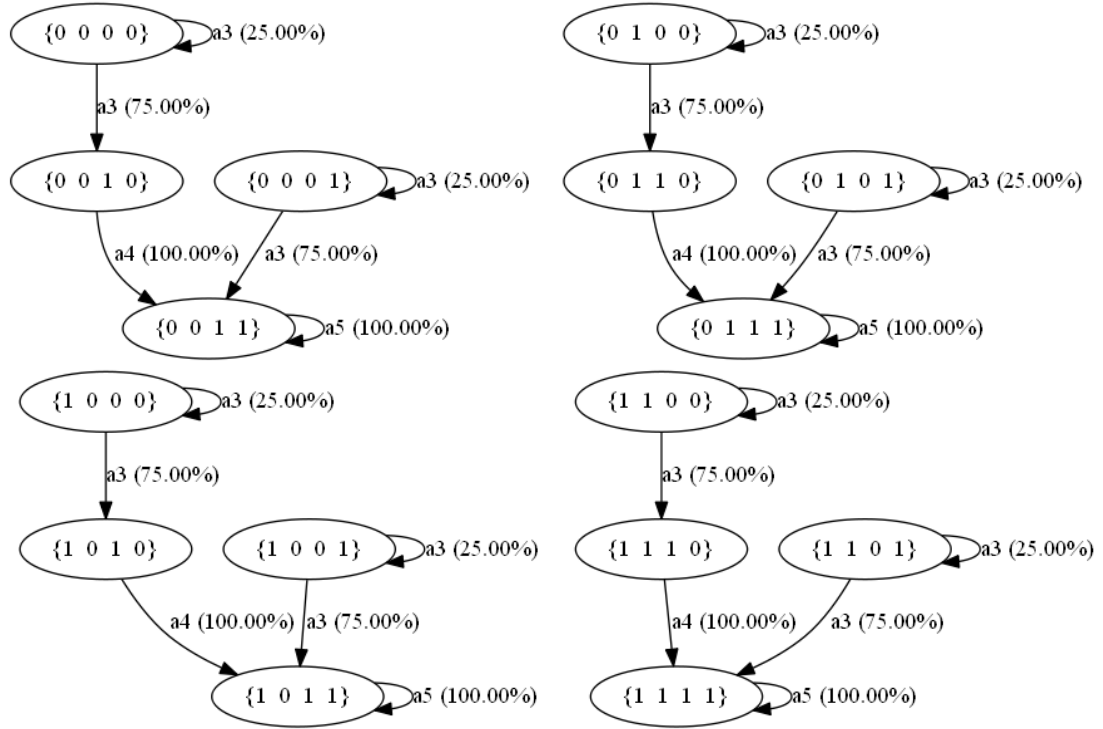


Figure 5-14: State/policy-action progression outcomes, column 1 from Table 5-16 (work_motivation g_3 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

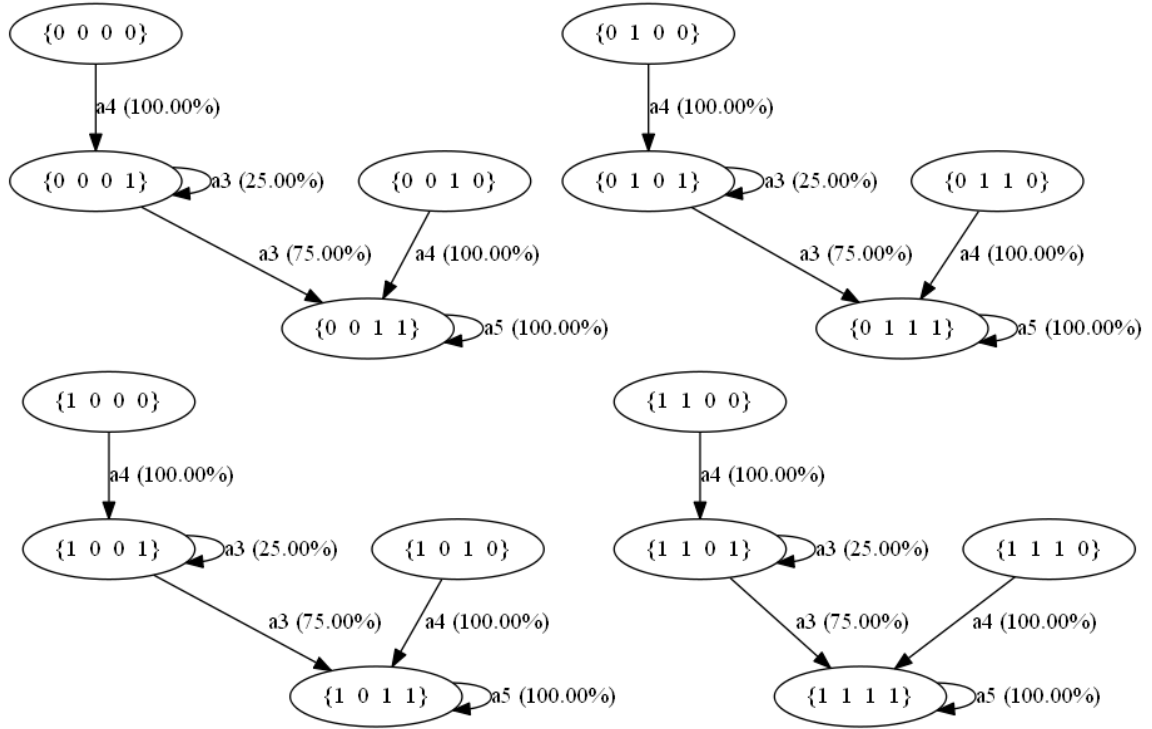


Figure 5-15: State/policy-action progression outcomes, column 2 from Table 5-16

(work_motivation g_3 vs. button_1_inactive f_1), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

Figure 5-12 and Figure 5-13 and show the state progression trees for Table 5-15, and Figure 5-14 and Figure 5-15 show the state progression trees for Table 5-16, respectively.

To test the impact of transition probabilities, we run cases with the reward weightings as above, for $n_h=0$, but different p_{kz} . Changes in the transition probabilities should have a large impact on the choice of policy action for goals with more than one satisficing action (e.g., sating the *blood_sugar_level* goal, with a choice between *eat_chip* and *drink_soda* actions for goal satisfaction). Changes in transition probability also should have an impact on action-choice, in terms of the order of action-choice within the policy when presented with multiple unsatisfied goals (e.g., if *work_motivation* has intermediate reward compared to other goals but a near-zero likelihood of completing, other goals with higher likelihood of completion and lower reward may still be pursued first, as seen via the policy's action-choice). These could either appear to be 'greedy' choices in action-goal accomplishment (short-term) or more 'tactical' choices (longer-term), depending

upon the relative likelihood of transition when balanced with the reward, due to the nature of the value iteration solver we use for the MDPs.

Table 5-17: Impact of transition probabilities, eat_chip (a_1) / blood-sugar_level (g_1) vs. computer_work (a_3) / work_motivation (g_3)						
State Features				Policy action for reward weights 1=eat, 2=drink, 3=work, 4=button press, 5=no-op (constant weights: $\alpha_1=0.50$, $\alpha_2=0.00$, $\alpha_3=0.50$, $\beta_1=1.00$, $k_I=4$) (constant p : $p_{44}=0$, $p_{5X}=1$)		
g_1	g_2	g_3	f_I	Probability set 1 $p_{11}=\mathbf{0.25}$, $p_{21}=0.75$, $p_{22}=0.50$, $p_{33}=\mathbf{0.25}$	Probability set 2 $p_{11}=\mathbf{0.25}$, $p_{21}=0.75$, $p_{22}=0.50$, $p_{33}=\mathbf{0.75}$	Probability set 3 $p_{11}=\mathbf{0.75}$, $p_{21}=0.75$, $p_{22}=0.50$, $p_{33}=\mathbf{0.25}$
0	0	0	1	3 (work)	1 (eat)	3 (work)
0	0	1	1	1 (eat)	1 (eat)	2 (drink)
0	1	0	1	3 (work)	1 (eat)	3 (work)
0	1	1	1	1 (eat)	1 (eat)	2 (drink)
1	0	0	1	3 (work)	3 (work)	3 (work)
1	0	1	1	5 (no-op)	5 (no-op)	5 (no-op)
1	1	0	1	3 (work)	3 (work)	3 (work)
1	1	1	1	5 (no-op)	5 (no-op)	5 (no-op)
x	x	x	0	4 (button)	4 (button)	4 (button)

In Table 5-17, we compare *eat_chip* against *computer_work* by changing the probabilities for the likelihood of completion for *eat_chip* towards the goal of sating/raising the *blood_sugar_level*, and the act of *computer_work* towards *work_motivation*. Note that in all cases, the high-priority goal weight will override other action-choices, for an optimal solution where the button-pressing action will always be picked if the button is active.

For probability set 1, we set the transition probabilities high and to the same value for both eating and working; *eat_chip* will sate *blood_sugar_level* 75% of the time. Note also that *drink_soda* will sate *blood_sugar_level* 50% of the time. Because the goals are weighted equally and so are the transition probabilities, our MDP policy solver defaults to the highest-numbered action as the tie-breaker (a_3) for the cases where both goals are unfulfilled. For states where only one of those two goals (*blood_sugar_level*, *work_motivation*) is not sated, the action with the highest probability of completion that active goal (a_1 , a_3) is chosen.

For probability set 2, *computer_work* has far lower likelihood of completing the *work_motivation* goal – only 25% probability of transition – while *eat_chip* still has 75% chance of satiating *blood_sugar_level*. Because the reward is the same for both satiating *blood_sugar_level* and *work_motivation*, the optimal policy chooses the *eat_chip* action explicitly over the *computer_work* action in all states where both these goals are unfulfilled; it tries to fulfill that higher-likelihood goal earlier.

For probability set 3, *eat_chip* has far lower likelihood of satiating *blood_sugar_level* – only 25% probability of transition – while *computer_work* has 75% chance of completing the *work_motivation* goal. This policy looks similar to the first case, except that the *drink_soda* action is chosen instead of the *eat_chip* action. This is because the *drink_soda* action has a higher likelihood of satiating *blood_sugar_level* (50%) than *eat_chip* does in this case.

All of these policy outcomes are intuitive given our equations and the choice of weights and probabilities. Figure 5-16, Figure 5-17, and Figure 5-18 show the development of the state progression trees for Table 5-17.

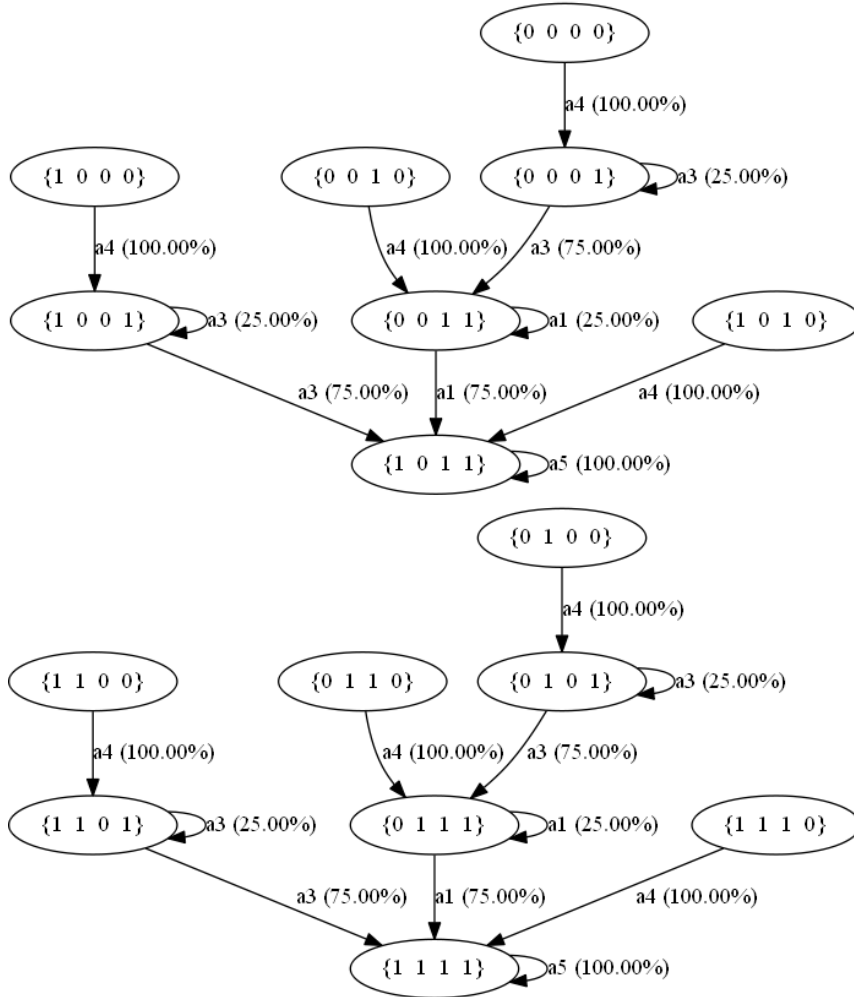


Figure 5-16: State/policy-action progression outcomes, column 1 from Table 5-17
 (blood_sugar_level g_I vs. button_1_inactive f_I), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

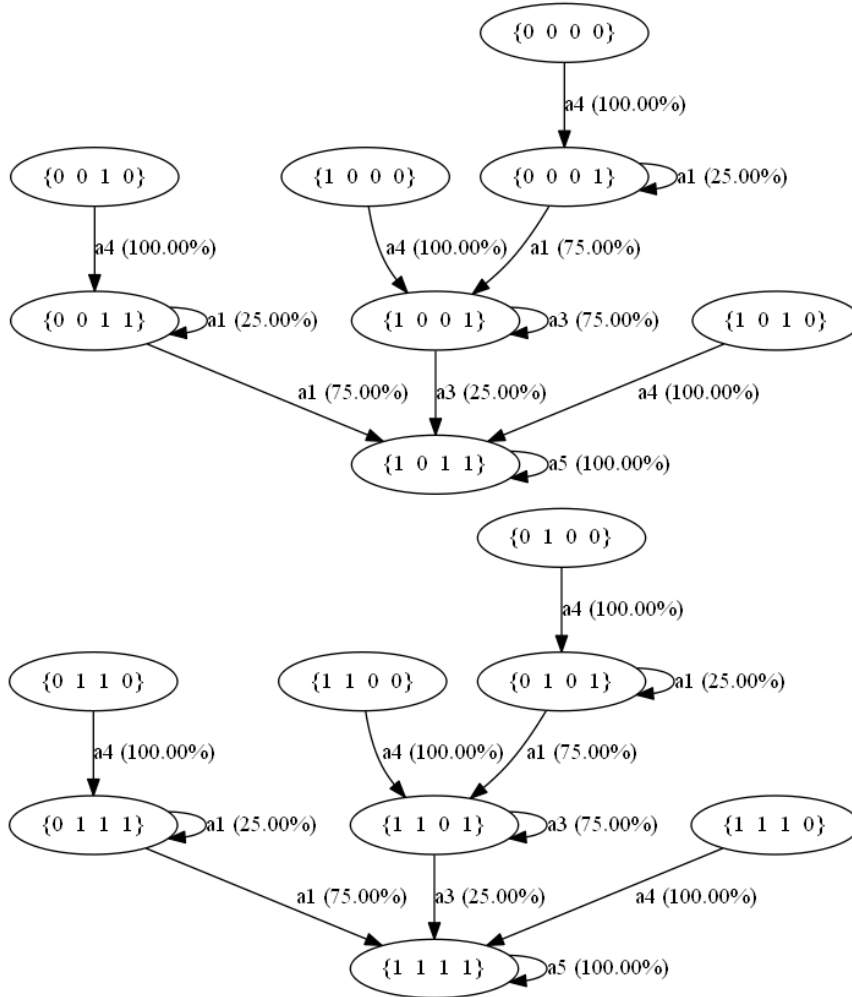


Figure 5-17: State/policy-action progression outcomes, column 2 from Table 5-17
 (blood_sugar_level g_I vs. button_1_inactive f_I), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

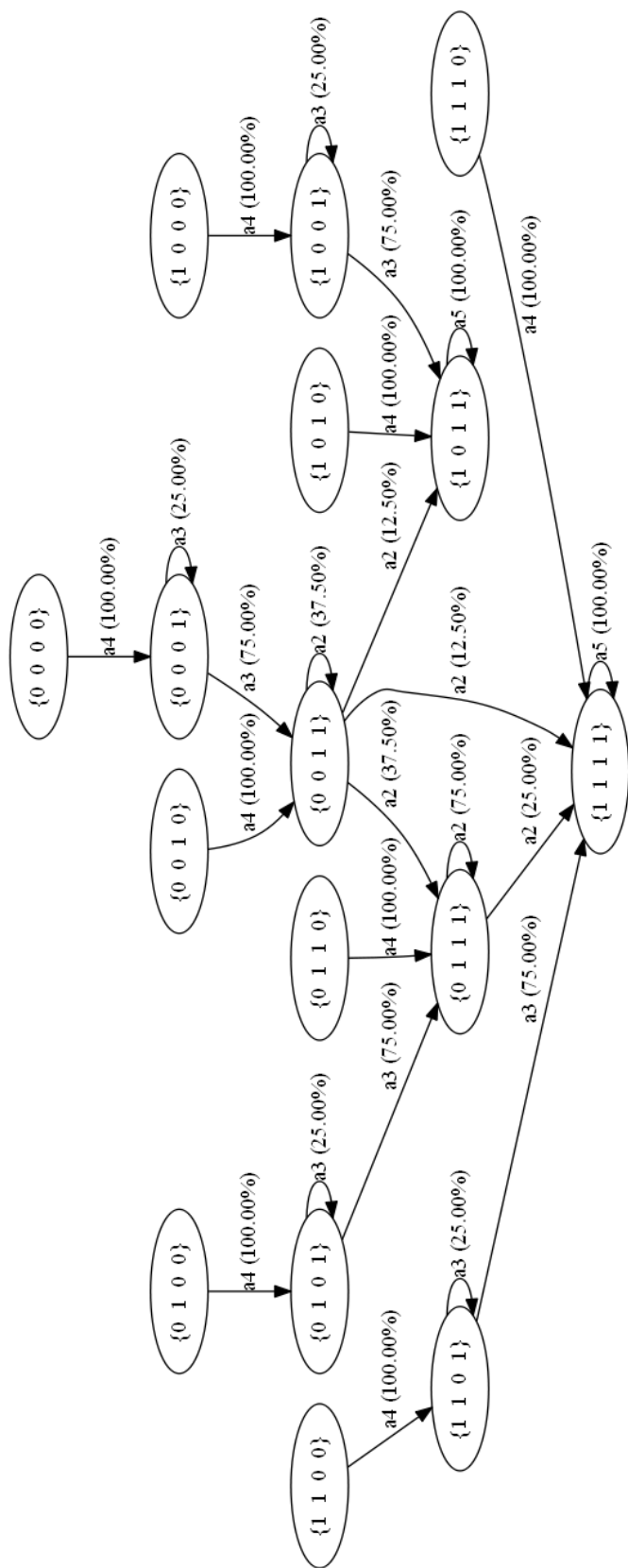


Figure 5-18: State/policy-action progression outcomes, column 3 from Table 5-17 (blood_sugar_level g_I vs. button_1_inactive f_I), $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

In Table 5-18 and Table 5-19, we also test the impact of varying the action-history length for dependent nominal actions (*eat_chip* and *drink_soda*) for similar transition probability weights as used above ($p_{11}=0.25$, $p_{21}=0.75$, $p_{22}=0.50$, $p_{33}=0.25$, $p_{44}=0.00$, $p_{5x}=1$; all other $p_{kz}=1$) for MDPs with action-history of length $n_h=\{0,1,2,3\}$, respectively, but for a more realistic set of non-zero reward and cost weights : $\alpha_1=0.25$, $\alpha_2=0.25$, $\alpha_3=0.50$, $\beta_1=1.00$, $k_I=4$. The α_z , β_z , and k parameter values chosen for the analysis are common sense values, given a lack of statistically-significant human subject experiment data. The run times specified as {history length, total time, infinite horizon value iteration time only, number of iterations} were: $\{n_h=0, < 0.5s, \sim 0.30s, 239\}$, $\{n_h=1, < 0.5s, \sim 0.30s, 59\}$, $\{n_h=2, < 2.1s, \sim 1.0s, 38\}$, and $\{n_h=3, < 10s, \sim 3.8s, 32\}$.

State Features	Policy action for reward weights, percentage chosen for A^i a_1 =eat, a_2 =drink, a_3 =work, a_4 =button press, a_5 =no-op (constants: $\alpha_1=0.25$, $\alpha_2=0.25$, $\alpha_3=0.50$, $\beta_1=1.00$, $k_I=4$, $p_{11}=0.25$, $p_{21}=0.75$, $p_{22}=0.50$, $p_{33}=0.25$, $p_{44}=0$, $p_{5x}=1$)										
	$n_h=0$	$n_h=1$, % chosen for A^i					$n_h=2$, % chosen for A^i				
$\{g_1, g_2, g_3, f_1\}$	--	a_1	a_2	a_3	a_4	a_5	a_1	a_2	a_3	a_4	a_5
$\{0,0,0,1\}$	a_3			1				.16	.84		
$\{0,0,1,1\}$	a_2		1					1			
$\{0,1,0,1\}$	a_3			1			.04		.96		
$\{0,1,1,1\}$	a_1	1					1				
$\{1,0,0,1\}$	a_3			1					1		
$\{1,0,1,1\}$	a_2		1					1			
$\{1,1,0,1\}$	a_3			1					1		
$\{1,1,1,1\}$	a_5					1		.2	.2	.2	.4
$\{x,x,x,0\}$	a_4				1					1	

Table 5-19: Impact of transition probabilities, $n_h=0$ and $n_h=3$						
State Features	Policy action for reward weights, percentage chosen for A^i a_1 =eat, a_2 =drink, a_3 =work, a_4 =button press, a_5 =no-op (constants: $\alpha_1=0.25$, $\alpha_2=0.25$, $\alpha_3=0.50$, $\beta_1=1.00$, $k_I=4$, $p_{11}=0.25$, $p_{21}=0.75$, $p_{22}=0.50$, $p_{33}=0.25$, $p_{44}=0$, $p_{5X}=1$)					
	$n_h=0$	$n_h=3$, % chosen for A^i				
$\{g_1, g_2, g_3, f_1\}$	--	a_1	a_2	a_3	a_4	a_5
$\{0,0,0,1\}$	a_3		.256	.744		
$\{0,0,1,1\}$	a_2		1			
$\{0,1,0,1\}$	a_3	.048		.952		
$\{0,1,1,1\}$	a_1	1				
$\{1,0,0,1\}$	a_3			1		
$\{1,0,1,1\}$	a_2		1			
$\{1,1,0,1\}$	a_3			1		
$\{1,1,1,1\}$	a_5		.2	.2	.2	.4
$\{x,x,x,0\}$	a_4				1	

In Table 5-18 and Table 5-19, we compare the nominal “no action-history” MDP to the distribution of actions over all combinations of A^i for each set of goal states, as shown in the leftmost column. Here, deactivation of the high-priority *button_1_inactive* goal has overriding cost, and sating *work_motivation* has higher reward than sating either *blood_sugar_level* or *hydration_level*, which have equivalent reward to each other. Also, *eat_chip* and *computer_work* have 75% likelihood of transitioning each of their respective goals if the entire action-history plus choice of a_k consists of that action (see Equation (5-23) in the general form for a longer explanation of this). Similarly, *drink_soda* has 50% likelihood of sating (only) *hydration_level* and 25% chance of sating (only) *blood_sugar_level* under similar circumstances. However, with an action-history of non-zero length, probabilities of *eat_chip* and *drink_soda* and *computer_work* span the entire combinatorial set of possibilities within the action-history. In states where the button is inactive, we expect to see: *computer_work* have highest priority for states where *work_motivation* has not yet been satisfied; the *drink_soda* action more likely to be chosen in states where both *blood_sugar_level* and *hydration_level* are not sated (since *drink_soda* is able to transition both those goals at once) and also where *hydration_level* alone is not sated; and *eat_chip* action for states where only *blood_sugar_level* is not sated. For states where none of the three mission goals are

satisfied, we expect that for the probabilities and weights selected, *computer_work* would be chosen most often (with *work_motivation* having higher priority), followed by *drink_soda* (able to impact two goals) and then *eat_chip* (only impacting *blood_sugar_level*).

In comparing the MDP policy outputs for state spaces with differing action-history length in Table 5-18 and Table 5-19, we see that the single-goal incomplete cases are straightforward in that the associated action with the highest probability of transition is chosen across all cases. For states where two goals are not satisfied, the policy output follows our expectations as given. In cases with longer action-histories, the choice of action seems to begin to fragment according to the individual circumstances of each state. However, this is partly due to a higher number of less-likely combinations being included in our rudimentary statistical model (e.g., $A^i = \{a_2, a_2, a_5\}$ for goal state = $\{0, 0, 0, 1\}$ leading to a choice of *drink_soda* (a_2) instead of *computer_work* (a_3) due to multiple occurrences of action a_2 in the action-history). It should be noted, however, that while the transition probability is higher for *eat_chip* sating the *blood_sugar_level* goal than *drink_soda* for *hydration_level*, *drink_soda* actions are more likely to be chosen earlier-on than *eat_chip* actions due to the reward being equal for both being completed, and a high-enough likelihood for both goals being completed at the same time if *drink_soda* is chosen, similar to observed policies from earlier examples. Thus, we expect the percentages of choice across that subset of A^i to vary more consistently between eating and drinking for cases where both of their goals have not been met.

Figure 5-19 and Figure 5-20 shows the development of states according to two of the policies in Table 5-18. Figure 5-21 through Figure 5-26 are zoomed-in portions of Figure 5-20. From every starting state, states progress from no goals satisfied, to one goal satisfied, and so forth, with all paths meeting at the absorbing ‘all goals satisfied’ state. This demonstrates that, even if our model is incorrect and some unexpected transition occurs, so long as unexpected states only progress ‘sideways’ (different goal achieved) or ‘down’ (one or more goals achieved) then we still have a path to goal completion that will eventually be realized.

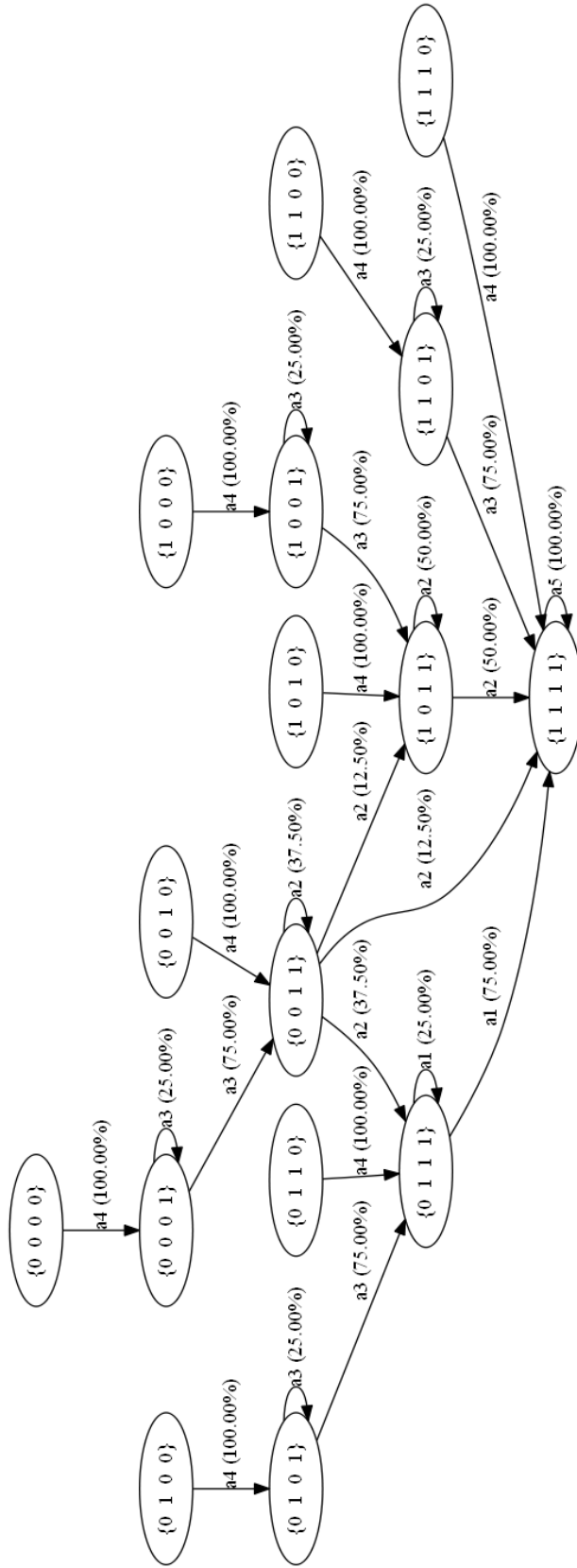


Figure 5-19: State/policy-action progression outcomes, column 1 from Table 5-18, $(n_i=0)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i\}$

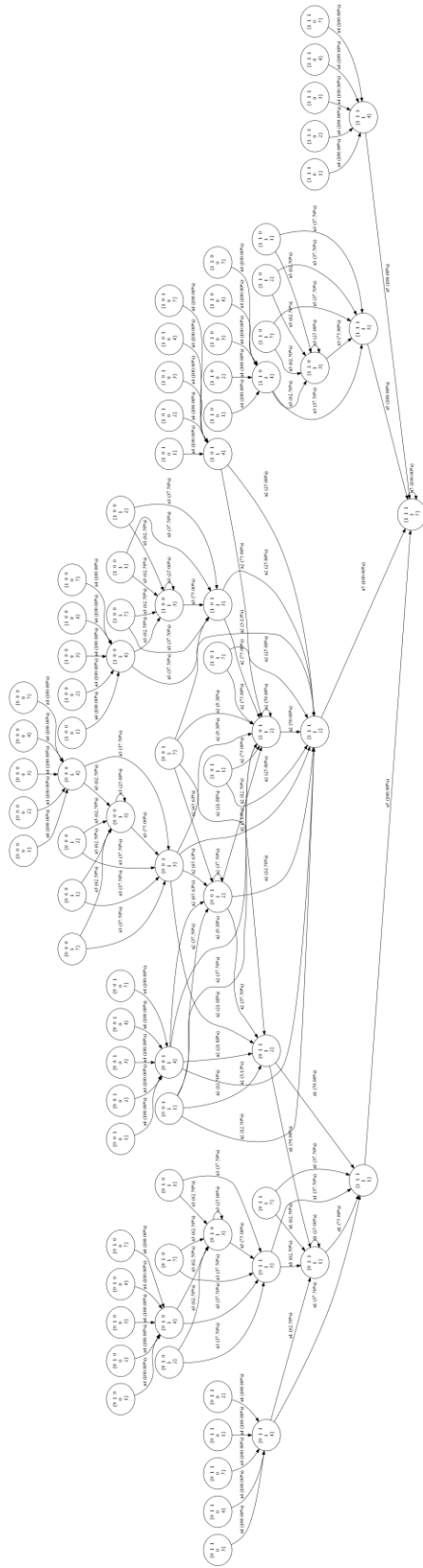


Figure 5-20: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, full diagram (low-resolution overview)

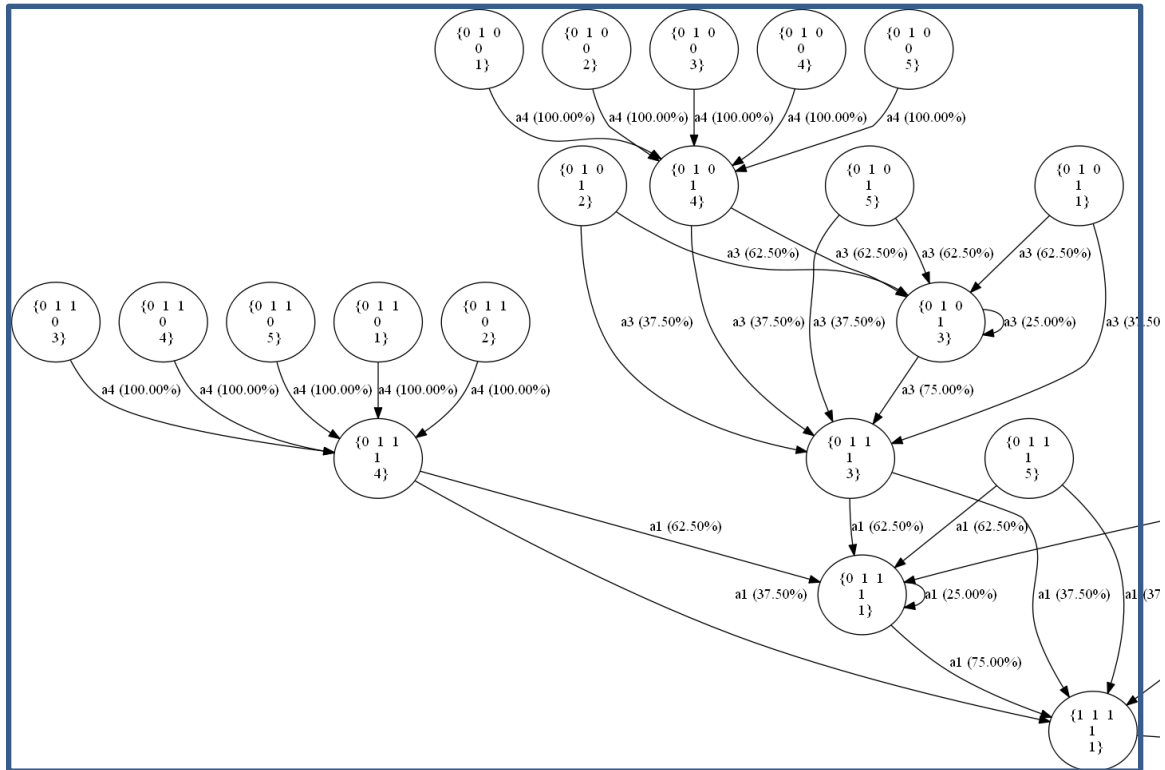


Figure 5-21: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (left-most)

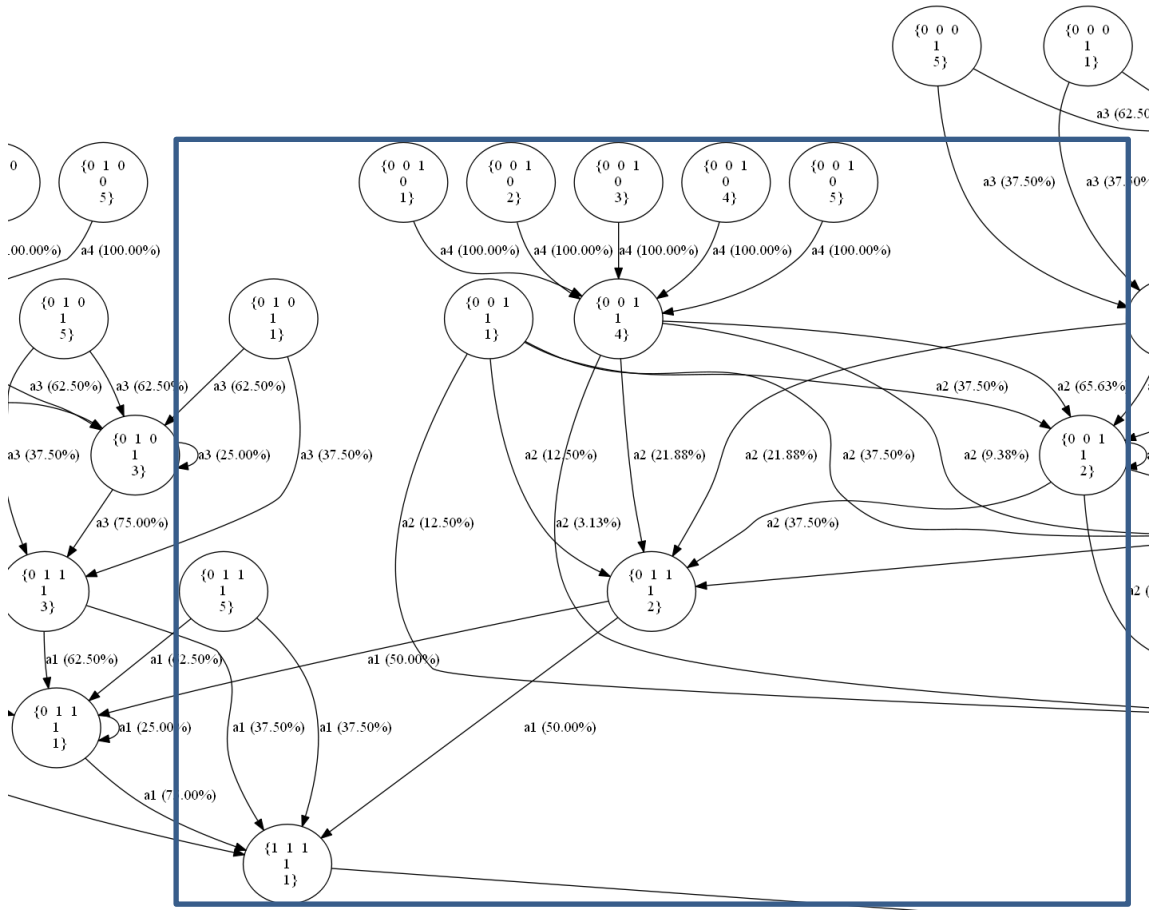


Figure 5-22: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (left-center)

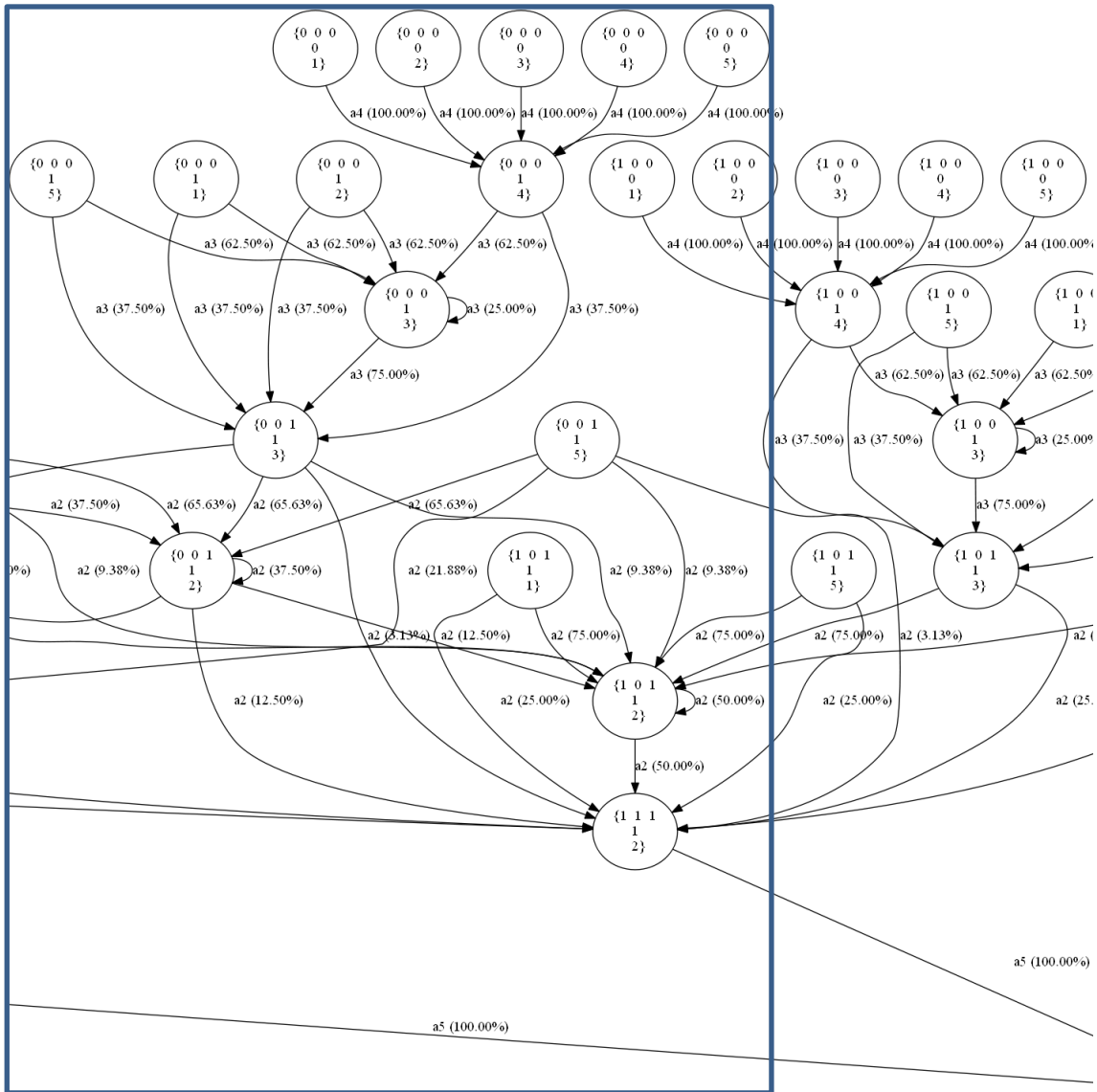


Figure 5-23: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (center)

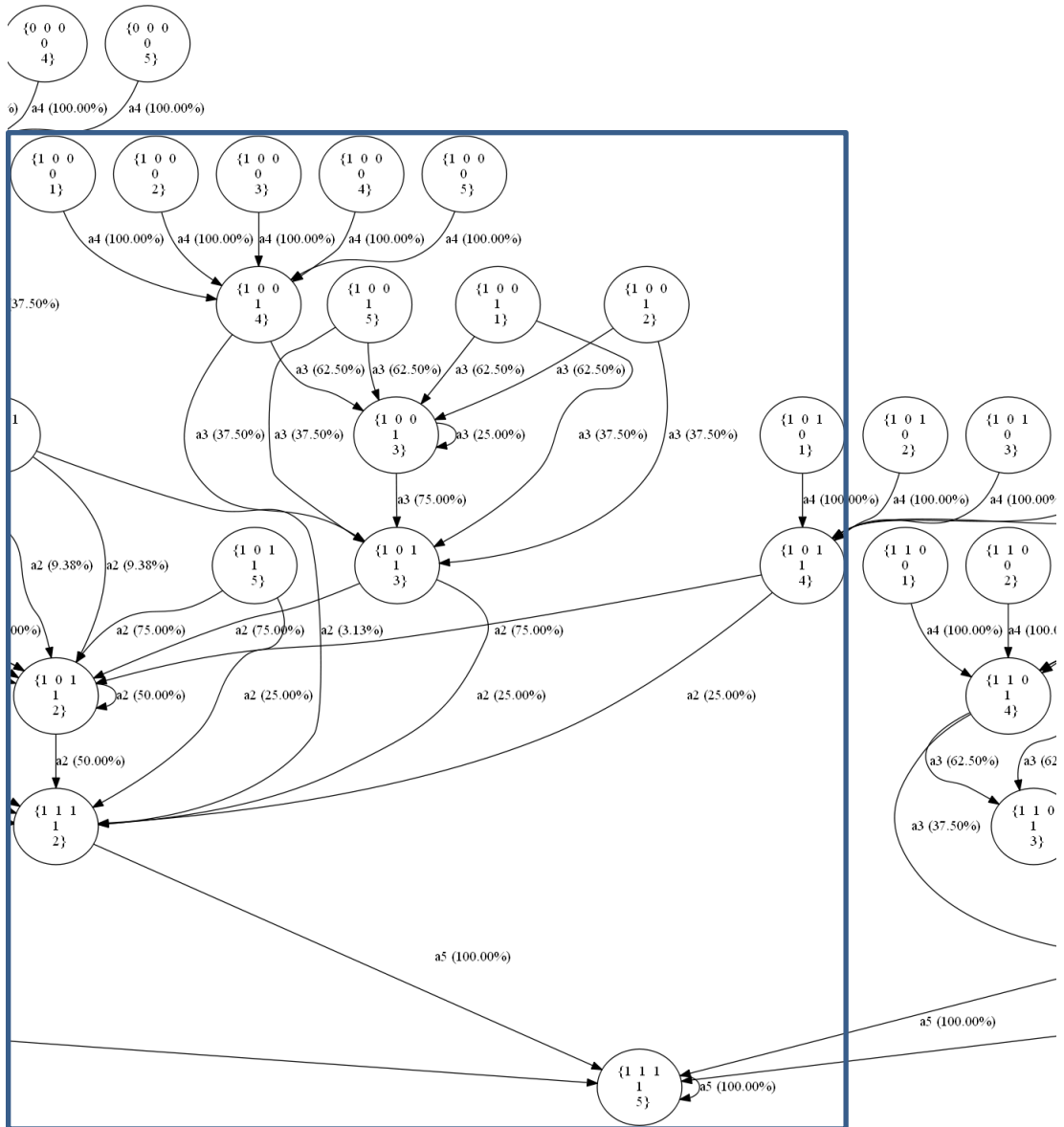


Figure 5-24: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (right-center)

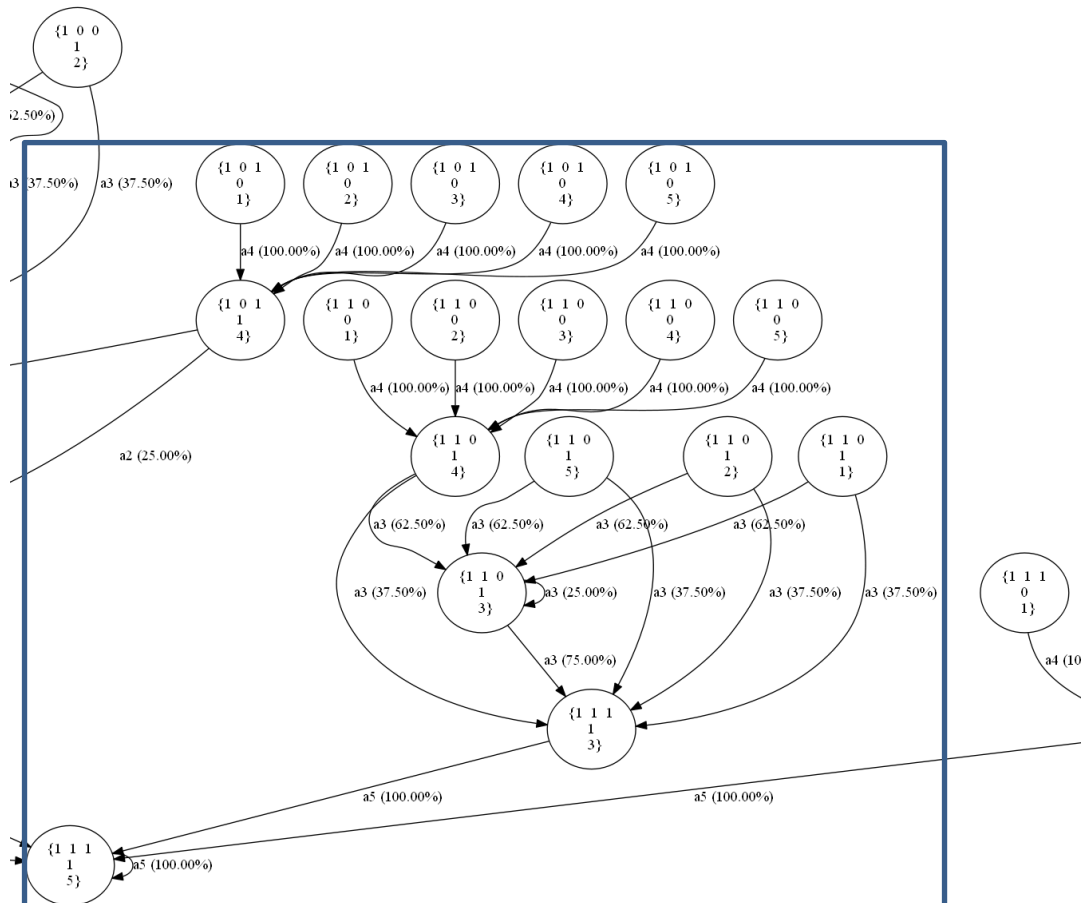


Figure 5-25: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (right of right-center)

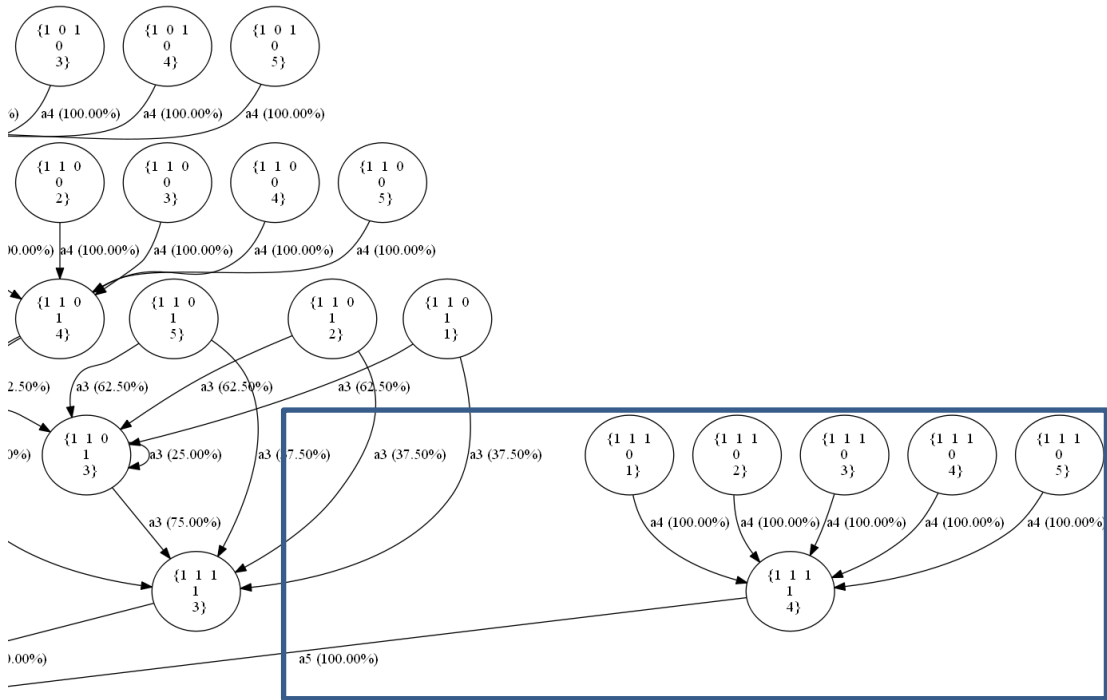


Figure 5-26: State/policy-action progression outcomes, column 2 from Table 5-18, $(n_h=1)$, $s^i = \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i\}$, partial diagram (right-most)

5.4.5 Inclusion of action-recognition input $a_{n_h+1}^i$ for one-step predictive lookahead

We have discussed several cases where action-recognition was assumed to not have provided a prediction of the in-progress action. However, this is a simplification – there will be times that action-recognition will recognize the currently in-progress action with sufficient confidence that we can treat it as observed. Thus, we can extend the equations above to include not only the “unknown” case at the very beginning of a human’s movement, but also this one-step lookahead case.

As an example, Equation (5-17) becomes:

$$\begin{aligned} s^i &= \{g_1^i, g_2^i, g_3^i, f_1^i, a_1^i, \dots, a_{n_h}^i, a_{n_h+1}^i\} \\ g_z^i &\in \{0,1\}, f_z^i \in \{0,1\} \\ a_k^i &\in \{1, \dots, n_a, n_a + 1\}, n_a = 5, a_k \in \{1, \dots, n_a\} \end{aligned} \quad (5-26)$$

with no-op (5) signifying lack of motion as before, and the addition of a new action, “unknown” (6). The “unknown” action implies that motion has been observed as occurring – signifying that a new action has been initiated but that the motion itself has not been identified, categorized, or labeled with sufficient confidence to feed back to the MDP executor. The reward function remains the same as before, being a function of goals, not actions or action-history, while Equation (5-20) becomes:

$$T(s^i, a_k, s^j) = p(f_1^j | f_1^i, a_{n_h+1}^i) * \prod_{z=1}^3 p(g_z^j | g_z^i, A^i, a_{n_h+1}^i) \quad (5-27)$$

Note that a_k is replaced by a_{n_h+1} in this equation. In this formulation, a_{n_h+1} is treated as a_k was previously. This is because, when a_{n_h+1} is known, it becomes the in-progress action that will have an immediate impact once completed, while a_k is a future-predicted action that has delayed impact. Our MDP should react just the same way as it did before, as we were effectively finding our best guess at the in-progress action, action a_k , as discussed in Chapter 5.2.2.

For the “new” cases where we transition with a_{n_h+1} known, we model the system in the following manner: no past action in A^i is “no-op” or “unknown” due to our earlier assumption of observability, a_{n_h+1} is instead set. If a_{n_h+1} is known, transitions represent delay, e.g. $p(g_z^j | g_z^i, A^i, a_{n_h+1})$ and not $p(g_z^j | g_z^i, A^i, a_{n_h+1}, a_k)$. If a_{n_h+1} is indicated by the observer as unknown, transitions are not delayed, e.g. $p(g_z^j | g_z^i, A^i, a_k)$, and a_{n_h+1} remains unknown. Goal states must be updated according to the given a_{n_h+1} , which by definition has not yet had an observed impact on the goal states in observed state s^i .

5.5 Conclusions and Discussion

We have presented a Markov Decision Process (MDP) formulation for predicting human astronaut intent during a space mission. We rely on the presence of scripts and a well-characterized environment to build action sequences and to presume an observer capable of action-recognition. The optimal MDP policy generates human action predictions that are then used by a robotic manipulator arm for decision-making. We have presented initial simulation results where we generate and examine policies developed from formulated models to better understand the characteristics of this ‘simulated human’ model. The models appear to have reasonable and intuitive policy results. We expect that the accuracy of the HIP system’s predictions will fall within the robustness constraints necessary to be useful for real-time task scheduling for a robot. Ground-based examples related to EVA and IVA are explored here, but it is assumed that, once astronauts become comfortable working side-by-side with robots that use HIP as part of a total HRI system (e.g., on space station), they will be much more ready to also accept them as companions on EVA.

As demonstrated above, we can encode a structured environment in the form of deterministic scripts into a MDP and recreate expected policies. We can then inject uncertainty into those transition probabilities, and use a reduced set of parameters to tune our models intuitively to create models with a sliding scale of internal uncertainty. This will allow us to predict and account for somewhat irrational or unexpected behavior in human agents who might be acting nearby or within a robot’s shared workspace. Policies that are partially-ordered also seem to be able to ‘recover’ when starting from unexpected

states, despite being suboptimal due to the unmodeled transitions. In terms of parameter choice, there are tipping points in policy outcomes between pairs of reward weightings from our simulation results. Drawing from this, it would likely be a good choice to determine in isolation a human’s likelihood for selecting and completing each action-sequence, then determine the relative reward weightings for pairs of goals one set at a time, and only then combine all the model parameters together. This would also help to identify unexpected dependencies between goals, if they exist. Work to further mature the HIP MDP defined in this chapter is summarized below.

5.5.1 Future work: evaluation and comparison against other methods

In this thesis, we do not evaluate the MDP for HIP in comparison to any other methods. Instead, we describe the structure of how to formulate HIP as a separate MDP. The most commonly used alternative formulation is the POMDP. In future work, it would be instructive to compare and contrast how the observability assumption in the HIP MDP impacts constraints (a negative for the MDP) used in reducing complexity relative to the POMDP. To compare the accuracy of HIP to other methods, it could also be beneficial to conduct human subject experiments. These experiments could be used to learn MDP parameters as well as to compare HIP methods. To evaluate the HIP MDP alone, learned models could be tested with additional human subject experiments. While tailoring MDP models to specific human preferences might be unrealistic for terrestrial applications, this would be feasible for space applications given the small size of the astronaut corps.

Benefits from performing human subject experiments would be twofold: we could explicitly evaluate the utility of our method, and we could determine whether the assumptions we have made to enable the MDP formulation are accurate. For instance, currently we assume the human remains sated short-term once they have finished eating. However, it is reasonable to assume that on a longer time scale, a human would see that goal as requiring satiation again. Not allowing these transitions back to an incomplete state makes the MDP policies suboptimal. We would need to observe the human long-term to see how important these divergences are and whether it might be necessary to update the general HIP formulation.

5.5.2 Future work: handling of model uncertainty

To prevent the necessity to revert to a POMDP formulation, we restricted the state-space to directly observed and “recognized action” state features in s^i . We presumed an external observer could recognize actions underway and/or completed by the human, thereby enjoying the significant representational and computational advantage of a MDP relative to a POMDP. We presume action-recognition and sensor deconfliction within the observer module, although in reality such capabilities would themselves carry some uncertainty in their conclusions. For more accuracy, there may be critical cases where we must include a belief-state associated with the outcome of the observed action a_{n_h+1} within the model, and recast the problem in a limited POMDP form.

With the current MDP formulation, we do not directly model the partial-completion of a goal; instead, we effectively calculate partial goal-completion using the action-history. A more expressive representation of goal might be needed for some task/goal models. Also, we currently disallow a fulfilled goal to transition to an unfulfilled state, a simplification that can be relaxed by allowing such transitions in the MDP probability tensor. This constraint should be relaxed, especially for progressively achieved (dynamic) goals (e.g., raising the blood sugar level or hydration level, which could conceivably transition to a less-complete state as energy is burned and the food and drink is digested).

Uncertainty may be present in human action *selection* (model internal match where the human decides to do something else), *completion* (ability-based, including effects of distraction), and/or *outcome* (external/environmental). Currently, our model only includes some sources of uncertainty in completion. We do not include *selection* cases where we allow nonzero probability of transitioning to a certain class of states that go off-script by breaking assumed constraints, e.g., an astronaut might be able to hold their screwdriver while removing a panel. We also do not include *selection* cases to account for the model possibly generating the wrong action; we currently assume that every action a_k will be present in the n_h 'th action-history attribute in the next state s^j . If we wished to include uncertainty in the action model, this might require including a third parameter, representing a nontrivial probability that the n_h 'th attribute in the action-

history for state s^j is some other action with some other goal-state effect. This set of possible transitions would need to include not just the set of states where $A^i \rightarrow A^j$ for a_k , but all sets of states $A^i \rightarrow A^j$ for each and every possible choice of action for a_k .

We originally chose a MDP modeling method because it keeps the computational complexity low relative to a POMDP. We assume that the other safety measures, namely a reactive capability to avoid collision even when the human's action is unexpected, will allow the observer to update quickly enough that the HIP MDP policy output meets given robustness and accuracy constraints.

5.5.3 Future work: simulated human vs. human matching models

The HIP MDP outputs expected human actions, but the human is not guaranteed to always stick to the expected behaviors. Thus, the selection of internal structure for the transition probabilities should represent the possibility of deviation from the MDP-optimized action through uncertainty in action outcome, given that the MDP always anticipates that it can execute an action but does not require the outcome of this action to be deterministic. We currently handle this possibility by incorporating into the action history the human's *sensed* response, the subjective action choice they made as specified by the action-recognition observer, rather than inserting the action retrieved from the policy into the history without feedback.

We have chosen functional forms for the reward function and probability tensor that are intuitive. However, if we want to use this model to actually predict a human, we need experimentally-informed data to populate our HIP model. There are many learning techniques that could be used to find best-fit model parameters. We could run a set of human subject experiments to determine those probabilities from observation, and then survey the astronauts regarding what importance they assign to the tasks to provide better reward weightings, thus giving us a 'seeded' model from which to start our learning process for our human intent model(s). We define two model types: a simulated human model that is a baseline for performing offline testing and can be used to seed the HIP module at the start of a scenario, and a human-matching model that can be updated online through model learning of the parameters to a better fit, requiring recalculation of the optimal policy output.

A simulated human model could be developed by aggregating statistics obtained through human subject experiments run prior to a mission in a scenario similar to the mission. A human-matching model would need to be more refined, and the output of this policy is what would be needed to supply input to the robot action-choice (RAC) MDP module online. The latter would be expected to give exactly the same predicted output as the human actually decides; the former would be expected to give output that corresponds to the statistical norm, for use in testing and to seed the human-matching model with an initial baseline. The parameters captured by such a learning process include reward weightings, the parametric values in the transition probability function, or the transition probability tensor directly. Such experimentally-derived models could be used for action-recognition and for specifying or augmenting the HIP MDP parameters online.

5.5.4 Future work: computation of action history length

For model simplicity, we assumed that actions older than the action-history are no longer relevant. However, there are times when this simplifying assumption may not be valid, such as circumstances where actions made towards completing one goal could be interrupted and resumed later. In this case, a very long action-history might need to be maintained to keep track of the progression towards goal-completion. In the worst-case scenario, the history might need to cover the entire length of the mission. The inclusion of intermediate goal states reduces dependence on long-term action history (as briefly illustrated in Chapter 5.4.2.1).

Some actions could also belong to more than one set of action-sequences, and thus help satisfy (or impact) more than one goal at a time. An example is picking up and holding a tool that could be used for more than one purpose. These cases would require a longer action-history so ‘forgetting’ important past events would not be a concern. The possible difficulties in, and impact of, correctly and consistently including additional MDP model complexity would be interesting to explore.

Chapter 6

Robot Planning for Optimal Human-Robot Interaction

6.1 Introduction

Given human intent information, the robot must decide how to act to achieve its own goals without negatively impacting its companion. This chapter describes a Markov Decision Process (MDP) for this process of robot action choice (RAC), which we define as *decision-making for robot mission completion given as input the robot state, goals already accomplished, observed environment state, and predicted human intent*. Assuming robot tasks are independent of and do not have higher-priority than human tasks, the robot must plan its actions to minimize probability of conflict with the human as well as achieving its own task-level goals.

Consistent with previous chapters, we assume no explicit communication between agents occurs. By assuming intent predictions from the HIP MDP described in Chapter 5 are “observations” of human intent, RAC can also make the assumption of full observability in its state, enabling use of the simpler MDP rather than a POMDP formulation for decision-making. This is because the observer module updates its output in real-time, and the HIP policy’s output updates near-instantaneously from that. HIP does not need to be perfect, as the reactive layer will preserve safety even in cases where the robot attempts actions that introduce conflicts. HIP does need to update its predictions to ensure RAC is receiving the best HIP estimate at each decision epoch. The RAC MDP therefore can define its fully-observable state from HIP input and the current state of the environment, while internally keeping track of the robot’s own goal state and action history as needed.

Use of an MDP for robot action choice (RAC) is not itself new [86]. The original contributions of this dissertation are instead the simplification of RAC for HRI, in a way that exploits the availability of human input. The assumption of full observability improves computational tractability relative to POMDP models. Inclusion of limited

memory in the form of a selective action history also can improve performance through improved consistency in action sequences. Incorporation of novel metrics into the reward function enables RAC to account for human and robot deconfliction (safety) as well as robot goal achievement. RAC safety metrics inspired by Kulic’s danger index [21] are designed to support a direct tradeoff between the safety of the human and the efficiency of the human-robot team.

Below, we first outline the general RAC MDP formulation and its instantiations for a single astronaut-robot case study that will be carried through the chapter. The scenario follows a similar physical setup to the human subject experiments in Chapter 3 that was investigated from the HIP perspective in Chapter 5. In the scenario, the (human) astronaut’s actions include eating chips, drinking soda, and solving math problems while seated at a workstation, while the robot’s actions involve pressing buttons in the shared workspace. We then present a series of simulation results for RAC alone versus RAC that uses HIP-supplied information provided through the integrated architectural framework described in Chapter 4, followed by an analysis of the safety-efficiency tradeoff for RAC policies using different weights.

6.2 Markov Decision Process (MDP) for Robot Action Choice (RAC)

Recall that a MDP is specified as the tuple: [39,35]

$$MDP = \{S, A, T(s^i, a_k, s^j), R(s^i, a_k)\} \rightarrow \pi(s^i) \quad (6-1)$$

which includes a set of n_s discrete states S , a set of n_a actions A , state dependent rewards $R(s^i, a_k)$ representing the reward of executing action a_k in state s^i , and transition probability tensor $T(s^i, a_k, s^j)$ representing the likelihood of transitioning from state s^i to s^j given action a_k . a_k is the action the robot takes in state s^i given policy optimal $\pi(s^i)$. The MDP assumes state s^i is fully-observable. For RAC, we presume that human intent predictions are observations available within state s^i .

We abstract action and state sets to reduce model complexity: most state feature values are either binary (0 or 1) or a finite set of values with low cardinality. The discretized model for our problem formulation is as follows.

6.2.1 States and Actions

The robot must model both itself and the human to make good decisions about how to act and react safely and efficiently in the shared work environment. The robot's overarching goal is to complete its own goal tasks without the physically-proximal human changing his/her behavior due to the robot.

For this reason, each state s^i in the set of all robot states $S = \{s^1, s^2, \dots, s^{n_s}\}$ includes three main components: H^i describing the state of the human companion, R^i describing the state of the robot, and a state feature called the discretized danger index attribute d^i that expresses a snapshot of safety for the human-robot system.

$$s^i = \{H^i, R^i, d^i\}, i = 1, \dots, n_s \quad (6-2)$$

Each RAC policy action a_k is chosen from:

$$a_k \in {}^R A, {}^R A = \{1, \dots, {}^R n_a\} \quad (6-3)$$

6.2.1.1 Human state features

Human state for RAC only must represent ongoing or upcoming human action choices that could impact robot state or action choice. Specifically, we only need information translatable to potential physical conflicts to make sure the robot doesn't distract or pose a risk of collision to its human companion. The robot does not need to know the human's task-level goals, as it is working independently. In our current formulation, the robot always defers to the human as needed to avoid conflict.

Human state for RAC thus can be specified by the human's actions ${}^H A^i$, as shown in Equation (6-4). Each action in turn maps to action-zone(s) which represent locations in physical space expected to be occupied or transited by the human:

$$H^i = \{{}^H A^i\} \quad (6-4)$$

Action-zones are annotations to the state. To simplify the problem of translating task-level actions to zones, we assume the human and robot stay in or move through a

common set of zones for the duration of their work. This is consistent for our space application involving a fixed-base manipulator and an anchored astronaut (simulated by a seated human on Earth). With this assumption, knowing the human’s current and upcoming task-level actions gives an understanding of his/her physical movements, which in turn allows specification of the conflicts that occur through action-zone occupation. The human is expected to occupy zones where task-level goals are currently being completed based on direct observation (*obs*) and also to occupy zones that must be transited to reach the site(s) where the next action predicted by HIP will be completed. RAC therefore models two human actions: the in-progress (directly observed) action and next action predicted by HIP:

$${}^H A^i = \{ {}^H a_{obs}^i, {}^H a_{HIP}^i \}, {}^H a_{obs}^i \in \{0, {}^H A\}, {}^H a_{HIP}^i \in {}^H A, {}^H A = \{1, \dots, {}^H n_a\} \quad (6-5)$$

${}^H a_{obs}^i$ is the in-progress action output by the observer module (called $a_{n_h+1}^i$ in Chapter 5), and ${}^H a_{HIP}^i$ is the action output from the HIP policy (a_k from Chapter 5). As explained in Chapter 4, if the human’s motion indicates that he/she has just initiated a new action, the observer identifies this new in-progress action as “unknown” (value 0), and the HIP output is effectively the in-progress action that has not yet been recognized, with no future predicted intent available to RAC. After the observer recognizes the in-progress action above a threshold of certainty/confidence, the HIP output shifts to the next predicted action of the human. We make the simplifying assumption that RAC can treat *both* of these attributes as 100% certain, a critical assumption enabling use of an MDP instead of a POMDP.

Note that ${}^H A$ does not include 0 in its count, as the value 0 is a reserved number; HIP never outputs an ${}^H a_{HIP}^i$ that is “unknown” (value 0) – the HIP MDP will always give its best guess as its action output.

6.2.1.2 Robot state features

The robot state features include:

$$R^i = \{ G^i, F^i, a^i, b^i \} \quad (6-6)$$

which describe the robot’s mission goals G^i and high-priority goals F^i , current action a^i , and current action-status b^i , respectively.

The robot’s mission goals $G^i = \{g_1^i, g_2^i, \dots, g_{n_g}^i\}$ and high-priority goals $F^i = \{f_1^i, f_2^i, \dots, f_{n_f}^i\}$ mirror human goal sets from the HIP MDP and are binary-valued. We do not include a robot action-history in RAC because it is unnecessary. The robot can choose its next action. It is not attempting to recognize a past-sequence by cross-referencing a script; it is instead determining the optimal action to take.

For RAC, the robot’s current (in-progress) action a^i must be included in the state since actions may be sufficiently long-term or with unpredictable duration to warrant interruption. The status of robot action completion must therefore be sensed by the observer, which in turn enables RAC policies to purposely continue or abort (interrupt) an ongoing action.

The human and robot have their own separate tasks to complete. In this work, we assume our robot will never fail to complete a task (i.e., it is guaranteed to reach a “halting state” for each action), although the outcome of executing an action may be uncertain.

$$\begin{aligned} a^i &\in {}^R A, {}^R A = \{1, \dots, {}^R n_a\} \\ b^i &\in \{0,1\}, a_k \in {}^R A \end{aligned} \tag{6-7}$$

Action a^i is in-progress if the action-status $b^i = 0$ or assumed to have just completed if $b^i = 1$. We require the action-status so the robot is able to model where the robot (arm) is currently located after an action has completed. The value 1 is a reserved number in ${}^R A$; this is the no-op action. In the simple case, $a_k \in {}^R A, a^i \in {}^R A$. However, a^i does not need to include the no-op action explicitly, as the external sensors the observer module uses will not be able to distinguish between a planned wait action (no-op) and a pause in an in-progress action. The observer module can recognize human motion, robot motion, and changes in the environment, so keeping track of the difference between a wait versus a pause within RAC is unnecessary. The human and robot actions are annotated with zone information, and the zones give an understanding of physical

location and motion mapped to each action in the workspace. Thus, for the purposes of maintaining safety, the differences between a no-op action (holding at a particular location) and a pause (during a particular motion) are negligible, so long as the observer updates the next action before a change in motion diverges significantly from the current physical location. Thus, ${}^R A$ in Equation (6-7) can be simplified to reduce the size of the state space:

$$\begin{aligned} a^i &\in {}^R A, {}^R A = \{2, \dots, {}^R n_a\} \\ a_k &\in \{1, {}^R A\} \end{aligned} \quad (6-8)$$

6.2.1.2.1 Action-zone annotation

Each human or robot action has one or more expected trajectories through physical space, representing motions or even just the volume of space occupied, as an agent completes an action in-place. In this work, actions map onto action-zones in 3D space, but this is not a one-to-one correspondence. To determine how the robot must act to accomplish its goals while avoiding interference with its human companion, physical conflict must be modeled in each action the robot and human executes. To model such conflicts, we define *action-zones* each agent is expected to occupy as each goal-seeking action is executed.

There are three aspects to each action: an action-zone that defines the physical space occupied (or being transited), directionality associated with the motion within the action-zone, and intent that implies the action's effect on goal completion. Note that these action-zones may overlap and may be occupied by either human or robot operating in the shared workspace. However, human and robot action-zone spaces may be defined distinctly, with translation as needed. Zones for each robot and human action modeled in the RAC MDP are given by:

$$\begin{aligned} {}^R Z^i &= \{ {}^R Z_1^i, {}^R Z_{a_k}^i \}, \\ {}^R Z_k^i &\in {}^R Z, {}^R Z = \{0, 1, \dots, {}^R n_z\} \\ {}^H Z^i &= \{ {}^H Z_{obs}^i, {}^H Z_{HIP}^i \} \\ {}^H Z_k^i &\in {}^H Z, {}^H Z = \{0, 1, \dots, {}^H n_z\} \end{aligned} \quad (6-9)$$

We assume zone data ${}^H Z^i$ and ${}^R Z^i$ are fully-observable and identified as part of the action-recognition process. The zones represent annotations to RAC MDP state thus are not explicitly listed in the MDP to minimize state complexity. Defining physical action-zones and their correspondence to the human and robot actions provides a method to calculate danger index and to better estimate the reward and transition probability functions than would be possible without this spatial data.

Action-zone definitions are discussed further in subsequent case studies (Chapter 6.4.1).

6.2.1.3 Conflict (danger) state feature

The discretized danger index, d^i [21] indicates the expected level to which the robot's presence and motion may pose risk to or introduce interference with its human companion. d^i is a discrete value, assigned corresponding to the bin (interval) in set D containing the current floating-point danger index:

$$d^i \in D, D = \{0, \dots, D_{max}\} \quad (6-10)$$

As an example, for a binary-valued index, a discretized danger index $d^i = 0$ would correspond to the interval $[0 \ DI_1]$, where DI_1 is the threshold value for that piece of the continuum of possible values of the danger index DI , as given in Chapter 2.2.2. This state implies that there is no chance of unsafe physical conflict in the near-future, so the robot can effectively ignore what the human is doing at-present. $d^i = 1$ would then correspond to the interval $(DI_1 \ DI_2]$, when the robot must begin to worry about collision occurring. DI_2 would be the highest danger index value allowable for safe human-robot operations with an HIP+RAC implementation.

A translation from physical space properties to danger index levels is calculated offline and dependent on the speed, inertia, and response time of the manipulator arm or more generally robot motion for each action. In the simplest case, and in our case studies, the danger index is a binary attribute and is effectively a flag that states whether the robot is moving too close to the human. This could be further discretized. For this HIP+RAC method, one might set thresholds of $DI_1=0.3$ (as Kulic chooses this value in [30] to avoid false positives) and a $DI_2=0.8$ (rather than the upper limit of 1), as values exceeding DI_2

would cause the reactive controller to take over until a sufficiently low DI is again restored. For safer, more conservative operations, lower threshold values could be chosen.

Recall that DI , the raw value of the danger index computed in real-time, corresponds to the maximum state of danger at a snapshot in time; usually this is defined by the closest approach point at the highest speed over the expected action trajectory. If the human and robot are motionless relative to each other, DI will be 0 regardless of the distance between them (due to the velocity factor $f_V(v)$; refer to Chapter 2.2.2).

The danger index can also be used in the lower-level reactive controller to prevent a collision, even if the RAC MDP policy selects a conflicting action. This will delay or interrupt the robot's action but will preserve safety. We assume that this capability is available to the robot engaged in HIP+RAC. Details of this process are left for future work.

6.2.2 Transition Probabilities

Transition probabilities are calculated from the robot's action-choice a_k and goal accomplishment status flags in the robot's state. These values are then modified by the sensed and predicted human intent to reflect the possibility of collision avoidance reactions. The robot executes action a_k output from the RAC policy at each timestep. We can account for the robot's reactive avoidance of conflict with the human in the MDP transition probability tensor.

The transition probability tensor for the MDP is given by:

$$T(s^i, a_k, s^j) = p(s^j | s^i, a_k) \quad (6-11)$$

which for RAC expands to:

$$T(s^i, a_k, s^j) = p(H^j, R^j, d^j | H^i, R^i, d^i, a_k) \quad (6-12)$$

Applying the chain rule $p(A, B) = p(A) p(B/A)$ enables a series of simplifications. First separate human state from robot and danger state:

$$T(s^i, a_k, s^j) = p(H^j | H^i, R^i, d^i, a_k) * p(R^j, d^j | H^i, R^i, d^i, a_k, H^j) \quad (6-13)$$

Next separate robot and danger state:

$$T(s^i, a_k, s^j) = p(H^j | H^i, R^i, d^i, a_k) * p(R^j | H^i, R^i, d^i, a_k, H^j) * p(d^j | H^i, R^i, d^i, a_k, H^j, R^j) \quad (6-14)$$

This yields a product of three functions computing probabilities of the next human state, robot state, and collision danger state, respectively. Each of these probability terms is described below.

6.2.2.1 Human state probability

Because we assume that the human is not impacted by the robot, we can write the probability of H^j as:

$$p(H^j | H^i, R^i, d^i, a_k) = p(H^j | H^i) \quad (6-15)$$

Observed (*obs*) and predicted (*HIP*) human state H^j will evolve in accordance with human action input computed in the current, observed human state H^i .

Table 6-1: Use of Human State Information in RAC

Case	Discrete Value		Moves to Case
	${}^H a_{obs}^i$	${}^H a_{HIP}^i$	
1	X	Y	1, if X has not completed (${}^H a_{obs}^i = X \neq 0$) 2, if X completes
2	0	Y	2, if Y has not completed (${}^H a_{HIP}^i = Y \neq 0$) 1, if Y completes (${}^H a_{obs}^j = Y$ is expected)

As shown in Table 6-1, two cases are present with respect to available human state information. In case 1, both observed action value X and HIP-supplied action Y are available. In case 2, HIP provides an action prediction but the observer has not yet recognized the current (in-progress) action. When transitioning, two outcomes are possible: (a) the human state remains the same, or (b) the human state transitions, in

which case the predicted intent remains the same (Y) until the action is recognized. We assume that the observer and HIP give perfect information to RAC in our case studies. Note, however, that we could relax our assumption that HIP is perfect so long as we incorporate non-zero probability of an incorrect prediction.

Because we assume that the human state information given to RAC is observable, this state transition demonstrates the progression of the human state within RAC with respect to the robot's actions. Relative to the rest of the RAC state features, if the human state remains the same after an internal MDP transition, then this indicates that the robot has transitioned its own state before the human finished their current action. If the human state changes asynchronously, the robot must now reconsider the best action to pursue or continue accounting for the new (or avoided) conflict situations.

An algorithm to compute $p(H^j|H^i)$ is given below. $p_{H a_{obs}^i}$ is the probability that the action $^H a_{obs}^i$ will stay in a case 1 state (not transition to case 2); $p_{H a_{HIP}^i}$ is the probability that the action $^H a_{HIP}^i$ will stay in a case 2 state (not transition to case 1). The algorithm is as follows: if $^H a_{obs}^i$ is not unknown (0), we assume $^H a_{obs}^i$ and $^H a_{HIP}^i$ remain the same with probability $p_{H a_{obs}^i}$, and that $^H a_{HIP}^i$ doesn't change while $^H a_{obs}^i$ becomes unknown ($^H a_{obs}^j=0$) with probability $1 - p_{H a_{obs}^i}$; if $^H a_{obs}^i$ is unknown, we assume that $^H a_{obs}^i$ and $^H a_{HIP}^i$ remain the same with probability $p_{H a_{HIP}^i}$, and that $^H a_{HIP}^i$ is accurate and is the new $^H a_{obs}^j$ while $^H a_{HIP}^j$ could transition to any action in $^H A$ with equal probability $\frac{1-p_{H a_{HIP}^i}}{H_{n_a}}$.

Algorithm $p(H^j|H^i)=p({}^H a_{obs}^j, {}^H a_{HIP}^j|{}^H a_{obs}^i, {}^H a_{HIP}^i)$:

```

if ( ${}^H a_{obs}^i \neq 0$ ) and ( ${}^H a_{HIP}^j = {}^H a_{HIP}^i$ )
  if ( ${}^H a_{obs}^j = {}^H a_{obs}^i$ )
    return  $p_{{}^H a_{obs}^i}$ 
  else if ( ${}^H a_{obs}^j = 0$ )
    return  $1 - p_{{}^H a_{obs}^i}$ 
  else return 0
else if ( ${}^H a_{obs}^i = 0$ )
  if ( ${}^H a_{obs}^j = 0$ ) and ( ${}^H a_{HIP}^j = {}^H a_{HIP}^i$ )
    return  $p_{{}^H a_{HIP}^i}$ 
  else if ( ${}^H a_{obs}^j = {}^H a_{HIP}^i$ )
    return  $\frac{1-p_{{}^H a_{HIP}^i}}{Hn_a}$ 
  else return 0
else return 0

```

Figure 6-1: Algorithm for calculating $p(H^j | H^i)$

In our case studies, we assume that every $p_{{}^H a_{obs}^i} = p_{{}^H a_{HIP}^i} = 0.50$; this represents a model of equal probability that the human will complete their latest action before, or during/after, the robot attempts the action. This models the possibility of the robot noticing a change in human state while both agents are in the midst of working to complete actions.

6.2.2.2 Robot state probability

The second term of Equation (6-14) can be simplified. Recall the components of R^i :

$$p(R^j|H^i, R^i, d^i, a_k, H^j) = p(G^j, F^j, a^j, b^j | H^i, G^i, F^i, a^i, b^i, d^i, a_k, H^j) \quad (6-16)$$

Rewrite as:

$$\begin{aligned}
p(R^j|H^i, R^i, d^i, a_k, H^j) &= p(G^j, F^j, b^j | H^i, G^i, F^i, a^i, b^i, d^i, a_k, H^j) \\
&* p(a^j | H^i, G^i, F^i, a^i, b^i, d^i, a_k, H^j, G^j, F^j, b^j) \quad (6-17)
\end{aligned}$$

Because we define a^j as independent of everything but a_k and assume that a^i , as a_k will always complete if it is not interrupted, and a_k can be no-op:

$$p(R^j | \dots) = p(G^j, F^j, b^j | H^i, G^i, F^i, a^i, b^i, d^i, a_k, H^j) * p(a^j | a^i, a_k) \quad (6-18)$$

with:

$$p(a^j | a^i, a_k) = \begin{cases} 1 & \text{if } a_k = \text{"no-op"} \text{ and } a^j = a^i \\ 1 & \text{if } a^j = a_k \\ 0 & \text{otherwise} \end{cases} \quad (6-19)$$

Recall that b^j is the in-progress or complete status of a^i . A constraint of our modeling method is that every time a robot action is completed we should see a goal state change, except when a_k is no-op. We also assume that the robot will not fail to complete a task for the action taken. Thus, if a goal didn't transition, then the action did not complete. This lets us simplify b^j , as it is only dependent upon change in goal status and the a_k that causes it, becoming 1 when a_k finishes. Therefore, we can rewrite Equation (6-18) as:

$$p(R^j | \dots) = p(G^j, F^j | H^i, G^i, F^i, a^i, b^i, d^i, a_k, H^j) * p(b^j | G^i, F^i, G^j, F^j, b^i, a_k) * p(a^j | a^i, a_k) \quad (6-20)$$

with:

$$p(b^j | G^i, F^i, G^j, F^j, b^i, a_k) = \begin{cases} 1 & \text{if } a_k = \text{"no-op"} \wedge (b^j = b^i) \wedge (G^i = G^j) \wedge (F^i = F^j) \\ 1 & \text{if } b^j = 0 \wedge ((G^i, a_k) \rightarrow G^j) \wedge ((F^i, a_k) \rightarrow F^j) \\ 1 & \text{if } b^j = 1 \wedge ((G^i, a_k) \rightarrow G^j) \wedge ((F^i, a_k) \rightarrow F^j) \\ 0 & \text{otherwise} \end{cases} \quad (6-21)$$

The effect(s) of a_k on G^i and F^i , whether in-progress ($b^j = 0$) or complete ($b^j = 1$), can be read from a lookup table. This is a deterministic model although the MDP formulation would also support uncertainty.

The $p(G^j, F^j | \dots)$ term from Equation (6-20) represents the likelihood of goal completion of the robot given the current state (i), human state, and the robot's action-choice. d^i , the danger index, can be removed as a dependency if d^i is binary-valued, as this will change according to a_k , so d^i will not directly influence goal states $\{G^j, F^j\}$.

$$p(G^j, F^j | H^i, G^i, F^i, a^i, b^i, d^i, a_k, H^j) = p(G^j, F^j | H^i, G^i, F^i, a^i, b^i, a_k, H^j) \quad (6-22)$$

Also, the previous robot action does not determine the goal outcome as that action is no longer in progress – only a_k does.

$$p(G^j, F^j | \dots) = p(G^j, F^j | H^i, G^i, F^i, a_k, H^j) \quad (6-23)$$

Equation (6-24) gives an example formulation. Here, the probability is 1 only when transitioning to states s^j where: the action is no-op and there is no change in goal state, the action a_k is likely to complete successfully (will not be interrupted due to upcoming conflict with the human) and the future goal it impacts is set to true (completed), or the action a_k is not likely to complete successfully (high chance of conflict) and no goals change. It is assumed that goal(s) not acted upon will remain the same. For likelihood of completion, conflict checks are performed on the assumption that both the in-progress action is finished and the robot's associated goal has been achieved.

$$p(G^j, F^j | H^i, G^i, F^i, a_k, H^j) = \begin{cases} 1 & \text{if } a_k = \text{"no-op"} \wedge (G^i = G^j) \wedge (F^i = F^j) \\ 1 & \text{if completion of } a_k \text{ expected} \wedge ((G^i, a_k) \rightarrow G^j) \wedge ((F^i, a_k) \rightarrow F^j) \\ 1 & \text{if } a_k \text{ not expected} \wedge (G^i = G^j) \wedge (F^i = F^j) \\ 0 & \text{otherwise} \end{cases} \quad (6-24)$$

We can generally assume conditional independence ($p(A,B)=p(A)*p(B)$) between goals in G^i and F^i . The joint probability is therefore the product of the individual probabilities of the goals:

$$p(G^j, F^j | H^i, G^i, F^i, a_k, H^j) = \prod_{z=1}^{n_g} p(g_z^j | H^i, g_z^i, a_k, H^j) \\ * \prod_{z=1}^{n_f} p(f_z^j | H^i, f_z^i, a_k, H^j) \quad (6-25)$$

These terms are used to determine the likelihood of goal completion of the robot given the current goal state $\{G^i, F^i\}$, the current and assumed-next-transitory state of the human $\{H^i, H^j\}$ that might cause conflict and make a goal's completion less-likely, and the robot's action-choice a_k .

6.2.2.3 Collision danger state probability

Consider the $p(d^j/\dots)$ term from Equation (6-14).

$$p(d^j | H^i, R^i, d^i, a_k, H^j, R^j) \\ = p(d^j | H^i, G^i, F^i, a^i, b^i, d^i, a_k, H^j, a^j, b^j) \quad (6-26)$$

The discretized danger index d^i is the maximum state of danger existing between the human and robot in the current MDP state. Thus, d^j is a calculation dependent upon relative human and robot motion in state j .

$$p(d^j | H^i, R^i, d^i, a_k, H^j, R^j) \\ = p(d^j | H^i, H^j, a^j, b^j) \quad (6-27)$$

In the simplest case when d^i is binary-valued, $p(d^j | H^i, H^j, a^j, b^j)$ is 1 only when a conflict will not occur and d^j is 0, or 1 when a conflict will occur and d^j is 1, a case in which risk is mitigated by MDP policy or (if considered not possible by mitigation) at the

reactive control layer. We assume that risk actively mitigated by MDP policy action will be performed in time for the reactive controller to not need to take over.

6.2.2.4 Complete transition probability function

To summarize, RAC MDP transition probabilities are given by:

$$\begin{aligned}
T(s^i, a_k, s^j) &= p(H^j | H^i) \\
&* p(G^j, F^j | H^i, G^i, F^i, a_k, H^j) \\
&* p(b^j | G^i, F^i, G^j, F^j, b^i, a_k) * p(a^j | a^i, a_k) \\
&* p(d^j | H^i, H^j, a^j, b^j)
\end{aligned} \tag{6-28}$$

With conditionally independent robot goals this simplifies to:

$$\begin{aligned}
T(s^i, a_k, s^j) &= p(H^j | H^i) \\
&* \prod_{z=1}^{n_g} p(g_z^j | H^i, g_z^i, a_k, H^j) * \prod_{z=1}^{n_f} p(f_z^j | H^i, f_z^i, a_k, H^j) \\
&* p(b^j | G^i, F^i, G^j, F^j, b^i, a_k) * p(a^j | a^i, a_k) \\
&* p(d^j | H^i, H^j, a^j, b^j)
\end{aligned} \tag{6-29}$$

In this formulation, we assume the reachable set of states s^j from a particular s^i are always going to be combinations between the various end-states of each human and robot action.

6.2.3 Rewards

The RAC MDP reward is dependent upon goal completion, safety of an action in terms of potential for collision, and robot energy consumed by an action:

$$\begin{aligned}
R(s^i, a_k) = & w_1 * R_1(G^i, F^i) + w_2 * R_2(d^i, a^i, b^i, a_k, H^i) \\
& + w_3 * R_3(a^i, b^i, a_k)
\end{aligned} \tag{6-30}$$

The first term R_1 gives a reward for advancing overall robot goal-completion, R_2 is the safety cost, and R_3 subtracts from reward depending on energy cost.

Goal completion reward is given by:

$$R_1(G^i, F^i) = \sum_{z=1}^{n_g} \alpha_z * r_{11}(g_z^i) + \sum_{z=1}^{n_f} \beta_z * r_{12}(f_z^i) \tag{6-31}$$

r_{11} rewards accomplishment of goals in the set G^i . Term r_{12} rewards high-priority goal completion and imposes cost for incomplete but active high-priority goals. α_z and β_z are the weights on r_{11} and r_{12} , respectively.

Equation (6-32) gives the general form of the reward terms:

$$\begin{aligned}
r_{11}(g_z^i) &= \begin{cases} 1 & \text{if } g_z^i = 1 \\ 0 & \text{otherwise} \end{cases} \\
r_{12}(f_z^i) &= \begin{cases} 1 & \text{if } f_z^i = 1 \text{ (complete)} \\ -k & \text{if } f_z^i = 0 \text{ (active / not complete)} \end{cases}
\end{aligned} \tag{6-32}$$

Because the goals are binary-valued, we can use them directly to calculate the reward R_1 as we did in Chapter 5. Equation (6-33) gives an example formulation for our case studies:

$$\begin{aligned}
r_{11}(g_z^i) &= g_z^i \\
r_{12}(f_z^i) &= (1 + k_z) * f_z^i - k_z = \begin{cases} 1 & \text{if } f_z^i = 1 \\ -k_z & \text{if } f_z^i = 0 \end{cases}
\end{aligned} \tag{6-33}$$

This is the same form as Equation (5-10).

If we wanted to equally weigh all goals and non-active high-priority goals, we would set all weighting factors α_z, β_z to 1. So long as $k_z * \beta_z > \max_z(\alpha_z)$, with $|k_z| > 1$, the MDP will prioritize high-priority task completion above each mission-goal action.

The second term R_2 penalizes risky states by assigning high costs to low human-robot separation distances (influenced by speed), and lesser costs to a choice of a_k that might cause conflict with future predicted actions without mitigation by the reactive controller.

$$R_2(d^i, a^i, b^i, a_k, H^i) = r_{21}(d^i) + r_{22}(d^i, a^i, b^i, a_k, H^i) + r_{23}(a_k, a^i, H^i) \quad (6-34)$$

Term r_{21} is a weighted penalty for the (binary) discretized danger index:

$$r_{21}(d^i) = \begin{cases} -w_d & d^i > 0 \\ 0 & d^i = 0 \end{cases} \quad (6-35)$$

This penalizes the robot for close distances or unsafe operating speeds. If d^i is 0, the safety is dependent upon the relative action status, because the robot arm might be motionless and waiting rather than far outside an unsafe shared zone. An example of this function as used in our case studies is Equation (6-36):

$$r_{21}(d^i) = -w_d * d^i \quad (6-36)$$

w_d is a positive weighting factor. As d^i is a discretized term (ranges of DI map to set values), we can use the scaled value directly in our reward function.

Term r_{22} is a cost term that measures the change in safety in choosing a_k , relative to the in-progress action. The general form of r_{22} is:

$$r_{22}(d^i, a^i, b^i, a_k, H^i) = \begin{cases} -w_{22} & b^i = 0, a^i \neq a_k, a_k \neq \text{"no-op"}, d^i \text{ expected to increase given } H^i \\ 0 & \text{otherwise} \end{cases} \quad (6-37)$$

If our safety constraints are of overriding importance, weights w_{22} and w_2 should be chosen greater than w_1 to ensure the maximum total reward is achieved by also maximizing R_1 . Determining the expectation of an increase in discretized danger index requires comparison between a^i and a_k with H^i ; the likelihood of collision is then assessed to determine relative change in d^i . For binary d^i , there can only be an increase if $d^i=0$ and $d^i=1$, or if the type of collision could have greater negative consequence (physical versus mental). An example algorithm for calculating of the likelihood of increase is given below.

Algorithm likelihood of increase for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$:

```

if  $^H a_{obs}^i \neq 0$ 
    if  $conflictR(a_k, ^H a_{obs}^i) > conflictR(a^i, ^H a_{obs}^i)$ 
        return 1
    else if  $conflictR(a_k, ^H a_{obs}^i) = conflictR(a^i, ^H a_{obs}^i)$ 
        if  $conflictR(a_k, ^H a_{HIP}^i) > conflictR(a^i, ^H a_{HIP}^i)$ 
            return 1
        else return 0
    else if  $conflictR(a_k, ^H a_{obs}^i) < conflictR(a^i, ^H a_{obs}^i)$ 
        return 0
else if  $^H a_{obs}^i = 0$ 
    if  $conflictR(a_k, ^H a_{HIP}^i) > conflictR(a^i, ^H a_{HIP}^i)$ 
        return 1
else return 0

```

Figure 6-2: Algorithm for calculating likelihood of increase for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$

We use an assignment of values for known conflicts shown in Equation (6-38). The type of expected conflict, if any, is read from a lookup table of conflict information that is calculated from the zone information that annotates each action; see the case study presented in Chapter 6.4.1 for more detail.

$$conflictR(a_y, ^H a_x^i) = \begin{cases} 2 & \text{physical conflict} \\ 1 & \text{mental conflict} \\ 0 & \text{no conflict} \end{cases} \quad (6-38)$$

Term r_{23} is a cost term indicative of the risk of choosing a_k given the upcoming predicted human actions in H^i ; the higher the risk of future conflict, the higher the cost. There are two components to r_{23} : weight w_{231} enacts a cost for conflict with the in-progress action, and w_{232} enacts a cost for conflict with the future-predicted action. (Thus, if ${}^H a_{obs}^i$ is unknown, ${}^H a_{HIP}^i$ is the in-progress action, while the future-predicted action is unknown, value 0.) An example calculation that uses the lookup function *conflictR* as given above is shown below:

Algorithm $r_{23}(a_k, a^i, H^i)$ for $H^i = \{{}^H a_{obs}^i, {}^H a_{HIP}^i\}$:
 if ${}^H a_{obs}^i \neq 0$ and $a_k = \text{"no-op"}$
 return $-w_{231} * \text{conflictR}(a^i, {}^H a_{obs}^i) - w_{232} * \text{conflictR}(a^i, {}^H a_{HIP}^i)$
 else if ${}^H a_{obs}^i \neq 0$ and $a_k \neq \text{"no-op"}$
 return $-w_{231} * \text{conflictR}(a_k, {}^H a_{obs}^i) - w_{232} * \text{conflictR}(a_k, {}^H a_{HIP}^i)$
 else if ${}^H a_{obs}^i = 0$ and $a_k = \text{"no-op"}$
 return $-w_{231} * \text{conflictR}(a^i, {}^H a_{HIP}^i) - w_{232} * \text{conflictR}(a^i, {}^H a_{obs}^i)$
 else if ${}^H a_{obs}^i = 0$ and $a_k \neq \text{"no-op"}$
 return $-w_{231} * \text{conflictR}(a_k, {}^H a_{HIP}^i) - w_{232} * \text{conflictR}(a_k, {}^H a_{obs}^i)$

Figure 6-3: Algorithm for calculating $r_{23}(a_k, a^i, H^i)$ for $H^i = \{{}^H a_{obs}^i, {}^H a_{HIP}^i\}$

Note that we compare a_k against each human action separately because we do not know whether the robot's action a_k will complete before or after the human's in-progress action has been completed – the amount of passage of time is not explicitly identified in the MDP models we use. This is preferable to the alternative, however, because we need to model time fluidly; humans can take non-trivial varying amounts of time to complete their tasks. The same lookup table used for r_{22} can be used to determine the conflicting actions and the risk of collision between them.

The third term R_3 penalizes the relative energy consumption for an action as a cost in the reward function. We assume that we know the physical location for each action and completion status pair $\{a^i, b^i\}$. If we know which goal we want to complete next, we know where we need to go next (new endpoint), we already know where we currently are (last endpoint known) and that we are stopped (motion stopped).

$$R_3(a^i, b^i, a_k) = r_{31}(a^i, b^i, a_k) - r_{32}(a_k) \quad (6-39)$$

Term r_{31} defines the action switching cost, encouraging actions to run to completion once begun. There is usually a balance between this and the cost associated with the current state of the actions and the discretized danger index in R_2 .

$$r_{31}(a^i, b^i, a_k) = \begin{cases} 0 & a_k = \text{no-op} \\ -w_{31} & b^i = 0, a^i \neq a_k \\ 0 & \text{otherwise} \end{cases} \quad (6-40)$$

Term r_{32} is the cost of motion, and is read from a lookup table; only the action a_k needs be known for this determination.

6.3 Metrics for RAC MDP Performance Evaluation

For each case study, we evaluate MDP formulations by comparing policy outputs as parameters – such as reward weightings – are varied for a consistent set of transition probabilities.

Our main evaluation metric is the percentage of an action-choice a_k – the number of times a_k is chosen by the policy divided by the number of states, given a particular reward weighting or other parameter choice.

When evaluating the policies resulting from different parameter choices, we discuss:

- Changes in optimal action-choice as determined by the policy for the various G^i and F^i goal state transitions over all possible states and the likelihood of conflict over time for selected scenarios given H^i
- Changes in optimal action-choice comparing each of the reward terms individually (R_1, R_2, R_3) and the total reward ($R_1 + R_2 + R_3$)

We can evaluate the relative impact of changes in the model by varying the values of two parameters in the HIP model at a time, solving for the optimal policies for each set of parameters, and looking for the tradeoffs.

Below, we examine the policy outcomes for RAC using differing amounts of human prediction information. We also examine the tradeoff between the various reward function term weightings and how this impacts the conflicts that occur – or are avoided. There are actions-to-goals that may cause physical conflict, mental conflict, or no conflict.

6.4 Case Studies

The IVA scenarios following from HIP studies investigated in Chapter 5 are presented here. These studies are similar to those used in the human subject testing in the Chapter 3 human subject experiments. Each scenario is presented below, following a discussion of action-zone definition for the case studies. We compare the policy output of RAC MDP's that do and do not use HIP-supplied information, and then look into the tradeoff between safety (number and type of expected collisions) and efficiency (policy value) in policies with tradeoffs in reward weights w_1, w_2, w_3 .

6.4.1 Encoding Zone Information within an RAC MDP state space

Actions and zones are interrelated. We assume progressions through or transitions between zones are predictable for a given action. Once the motion is matched as belonging to a zone with sufficient certainty, the correct action will be identified.

To simplify the problem of tracking risk of collision for task-level actions, we assume the human and robot stay in or move through a common set of zones in physical space for the duration of their work. This is consistent with our space application involving a fixed-base manipulator and an anchored astronaut (simulated by a seated human on Earth). With this assumption, knowing the human's current and upcoming task-level actions can be translated to a prediction of his/her physical movements, which in turn allows specification of the conflicts expected to occur through action-zone occupation. The human is expected to occupy zones where task-level goals are currently being completed based on direct observation (*obs*) and also to occupy zones that must be transited to reach the site(s) where the next action predicted by HIP will be completed. A similar translation can also be applied to the robot.

A zone is a discretization of 3-D space near the human and robot. An action-zone is a discretization of 3-D space occupied by the human and robot during an action. Each action-zone can be comprised of one or more partitions in physical space and represents one of two quantities: a snapshot of the volume an agent occupies once its goal-seeking action is “complete”, or a trajectory envelope bounding the space through which the agent is or will be moving. The latter is computed based on both a starting location and an ending location. In our case studies, we assume each agent has a known fixed base location within an inertial coordinate system, enabling unambiguous and compact definition of the collectively reachable workspace. Each action in the MDP state has an action-zone annotation. This annotation is used to help calculate the probability of collision, or increase/decrease in discretized danger index state. Danger is then estimated with respect to relative separation between human and robot.

Two options for representing action-zones are possible:

- 1) 2D or 3D space is partitioned: in this protocol, human and robot occupation regions are defined. For any action, a binary mapping of expected occupation (1=occupied, 0=vacant) for human and robot is generated. Distances between zones can be tabulated based on worst-case (closest possible approach) or centroid distances.
- 2) Envelopes of space can be generated around an action’s expected trajectory: with this representation, there are two sets of integer-valued attributes. One set indicates presence of the human in a set of zones, the other indicates the presence of the robot in a set of zones.

Examples of using the two methods for a case of pressing two buttons are given below.

Each action has a unique spatiotemporal effect on the state, both on the goal state and on the space the agent performing the action occupies while in the goal state and while in transit to the goal state. We depend upon the collision avoidance algorithm using Kulic’s danger index at the reactive control layer [30], which can override the MDP and remove itself from the area occupied by the human to assure safe operations in a worst-case scenario. However, the tradeoffs for doing so are extra energy use for the avoidance

trajectory and slowing or temporarily stopping the robot's motion toward the goal state to avoid the collision. The possibility of a collision avoidance action must be included in the state; in our case, it is included through a nonzero possibility that a goal-seeking action will terminate without successful completion, leaving the robot in a safe zone rather than in the intended goal completion zone. Thus, to preserve safety and for completeness in our closed-world model, our collision-avoidance strategy for RAC MDP requires us to include at least one position outside of human space to which the robot arm can retract safely for each robot action (in-progress or completed). Trajectories to each 'safe pose' must be nonconflicting regardless of the human action currently active at the time. We discuss ways of encoding or including this in the below case study.

Methods of calculating conflict potential are discussed for each action-zone definition below.

6.4.1.1 Action-zone Definition #1 – regular grid, pattern of 2D or 3D shapes

Figure 6-4 shows an action zone example in 2-D using definition #1.

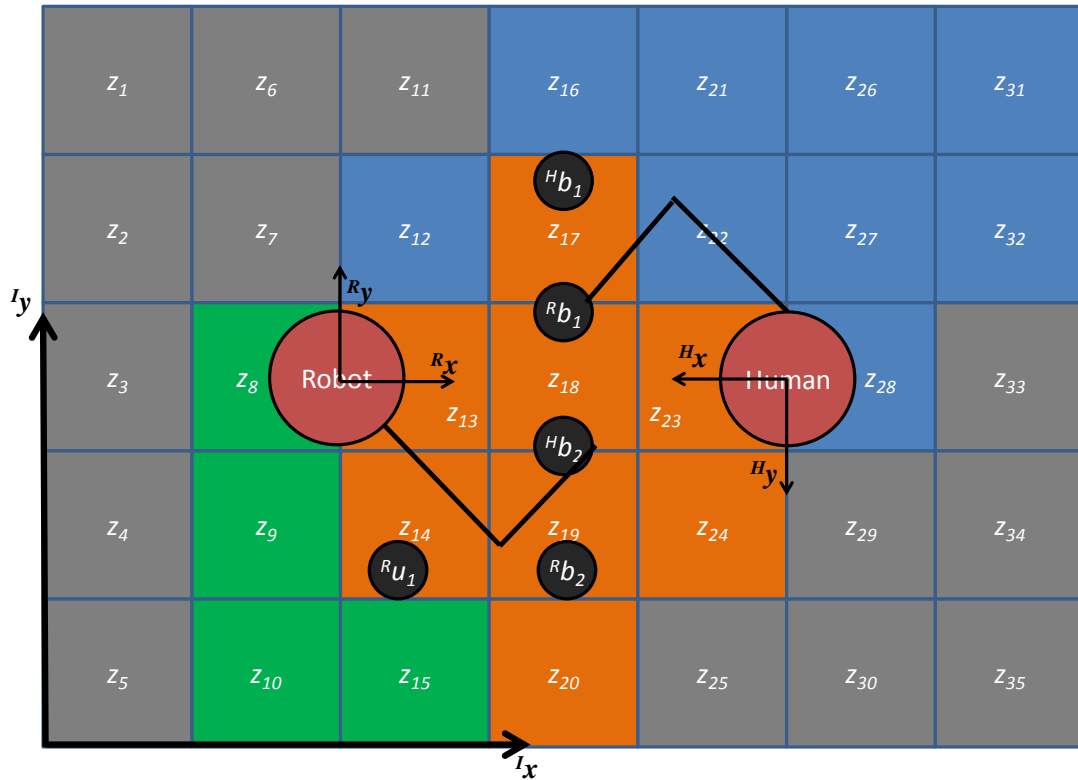


Figure 6-4: Zone partitioning using Definition #1, 2-D case, overhead view; fixed-base frames for agents are shown; zones are designated orange = occupiable by both, green = reachable by robot only, blue = reachable by human only, grey = non-occupiable

The numbered zones are unique discrete regions in inertial space specified in an inertial coordinate system $\{^I x, ^I y\}$. The set of zones that must be defined for each agent covers the reachable workspace plus additional space occupied by the agent (e.g., body, arm(s)) while manipulating itself through the reachable workspace. As shown in Figure 6-4, zones colored green and orange are reachable by the robot, while zones colored blue and orange are reachable by the human. Grey zones are not reachable by either agent – they cannot be occupied by either agent under the fixed-based assumption). The orange zones cover the regular grid reachable by any agent. A subset of these zones – $\{z_{17}, z_{18}, z_{19}\}$ – are areas that are potentially in conflict, due to the expected motions in each agent’s task space. The human is potentially tasked with pressing a button at location $^H b_1$ or $^H b_2$, and the robot is tasked with pressing buttons at $^R b_1$ or $^R b_2$ or may retract to a position outside of human space at location $^R u_1$.

For this definition, occupation regions for an action-zone are defined by a given value set of zones. Zones are set to 1 if they correspond to the space that the agent will reside at or sweep through for the trajectory of motion associated with that action, and 0 otherwise. Because the zones are defined as a common set, calculating a binary (yes/no) possibility of collision can be done by comparing a human's action-zone $^H a_{z,x}$ for an action $^H a_x$ and a robot's action-zone $^R a_{z,y}$ for an action a_y . By taking the union of the zone values of $^H a_{z,x}$ and $^R a_{z,y}$ and counting how many zones are 1, we immediately see the overlapping areas of conflict that are reachable by both. A nonzero overlap means there is collision potential, and the more discretized zones that overlap, the higher the possibility of collision and the more dangerous it is to have the robot perform that particular action a_y while the human performs its action $^H a_x$. This gives the worst-case assumption, as we are matching possible overlap without regard for circumstances of timing. Once the zones associated with an action-zone are determined to be in conflict or not, we can then determine which actions conflict with each other, knowing their action-zones. In the most conservative case, if there is a nonzero overlap between the action-zones annotating the robot action and human action, we would set the discretized danger index $d^i=1$ (or expect $d^i=1$).

6.4.1.2 Action-zone Definition #2 – custom grid, based on movement-envelopes and areas of conflict

A custom grid is an extension of the regular grid concept. The action-zones in this circumstance are considered to be a cohesive whole, a region in 2-D or 3-D space, rather than a binary set of occupied/not occupied regions. Thus, zones and action-zones are equivalent. This can be calculated for an action in one of two ways:

- 1) A discretization with a very small regular grid size is done as in Definition #1. The occupied regions (set to 1) for the action's trajectory are calculated from kinematic models or human subject test data. The occupied regions are aggregated to form a cohesive region in space that becomes the action-zone for that action. This is essentially the same as Definition #1, but reducing the size of the action-zone specification by only remembering the occupied zone regions for each action. The resulting action-zone is then defined as a polyhedron.

- 2) A motion trajectory in space for an action is expanded outward by a safe error margin to create an envelope in space. This envelope in space is the action-zone. The resulting action-zone is generally defined as an envelope around a B-spline curve that best-matches the action's trajectory.

We calculate collision avoidance via occupation potential for 1) in a similar way as Definition #1, by comparing the subset of zones that overlap. For 2), we calculate the separation distances over the course of the entire action using the technique described in Ref. [30] with Kulic's danger index.

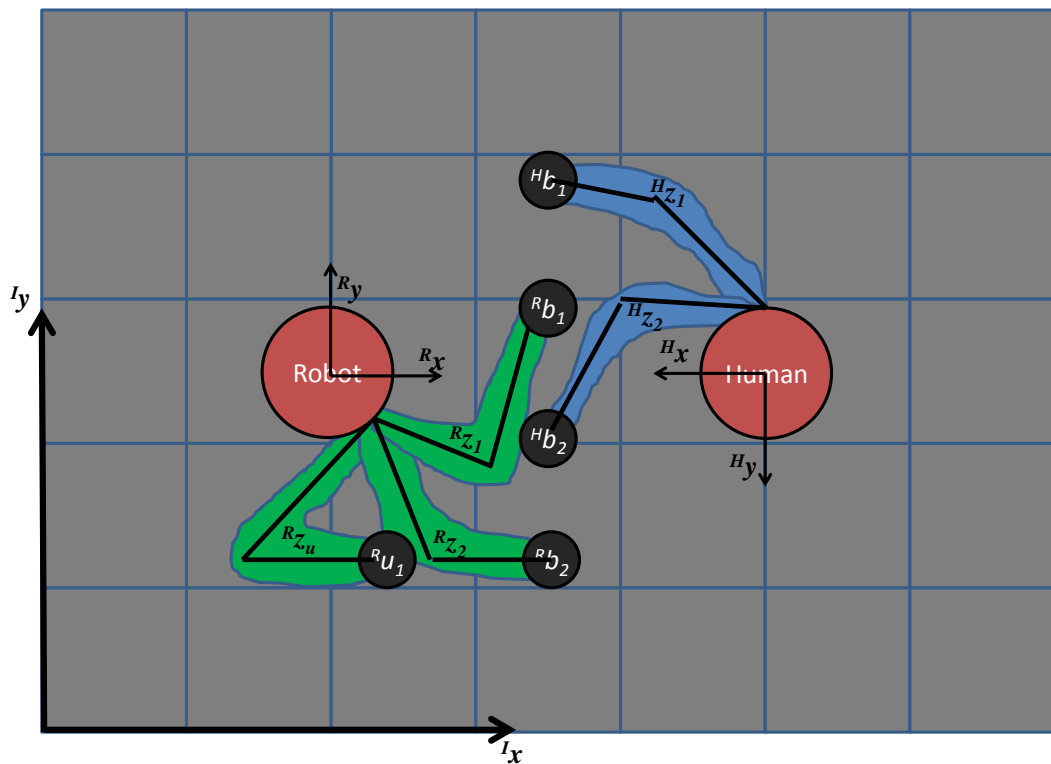


Figure 6-5: Action-zone partitioning using Definition #2, 2-D case, overhead view, stationary arm positions

For the robot and its three positions as shown in Figure 6-5, there are two zones that demarcate a stationary arm position at a button that must be pressed, one zone that demarcates a retracted unstow position that will not cause conflict with the human, and three zones that demarcate the area that the robot arm will pass through as it moves along a given trajectory between two of the positions to perform a particular action – a sweep

zone. Each sweep zone captures an agent's locations at action onset and completion, and the restricted trajectory between them. Thus, stationary zones are a subset of at least one sweep zone, e.g. ${}^R z_1$ and ${}^R z_2$ in Figure 6-5 are a subset of the sweep zone ${}^R z_{12}$ in Figure 6-6. For the human and its two buttons shown, there are two stationary zones and one sweep zone. A zone ${}^R z_0$, not shown, is defined as any other robot motion or location in the robot reachable space that does not match goal-seeking behavior and is not shared by the human; zone ${}^H z_0$ is the human-equivalent of this. The robot's unstowed waiting pose is a unique action-zone: it is an area that the robot arm can reach that is outside the human's reachable workspace, where no impingement or conflict with the human is possible. Figure 6-5 shows all of the end poses for both agents. Figure 6-6 shows an example of two sweep zones with overlap. The region in orange delineates the area of conflict that is part of both zones.

Since there is overlap between zones ${}^R z_{12}$ and ${}^H z_{12}$, we would expect it likely for a binary discretized danger index d^i to become =1 at some point during a robot motion ${}^R b_2$ to ${}^R b_1$ if the human simultaneously moves from ${}^H b_1$ to ${}^R b_2$. In fact, d^i may transition from 0 to 1 well before entering the (orange-colored) area of conflict as the motion is sensed if the danger index (DI) rises above the threshold to which we attribute a high value of d^i . Recall that Equation (2-17), used to compute DI , consists of a product of three terms: a distance factor f_D , a velocity factor f_V , and an inertia factor f_I . According to the limits discussed in Chapter 6.2.1.3, if $DI > 0.3$ while the robot traverses within the known sweep zone area, the binary d^i is set high (e.g., when all three factors have value ~ 0.67 or higher). Further, a $DI \geq 0.8$ would force the reactive controller to move into collision avoidance mode where MDP policy actions are temporarily ignored (e.g., when $f_I=0.67$ and both f_D and f_V have value ~ 1.1 or higher).

This danger index 'buffer' is what gives the robot the opportunity to react at the policy-execution level, but it also may require more immediate reaction to remain clear. We generally assume in our case studies that collision avoidance mode, if enacted, does not move us out of the current zone, but along a retreat path within the targeted zone. However, so long as we can identify the motions and map them to actions with certainty, nearby human and robot motions that give a high value of DI do not need to impact our

robot unless an area of conflict is upcoming. If the human is not performing a conflicting action, d^i can remain in a low state without negative consequence.

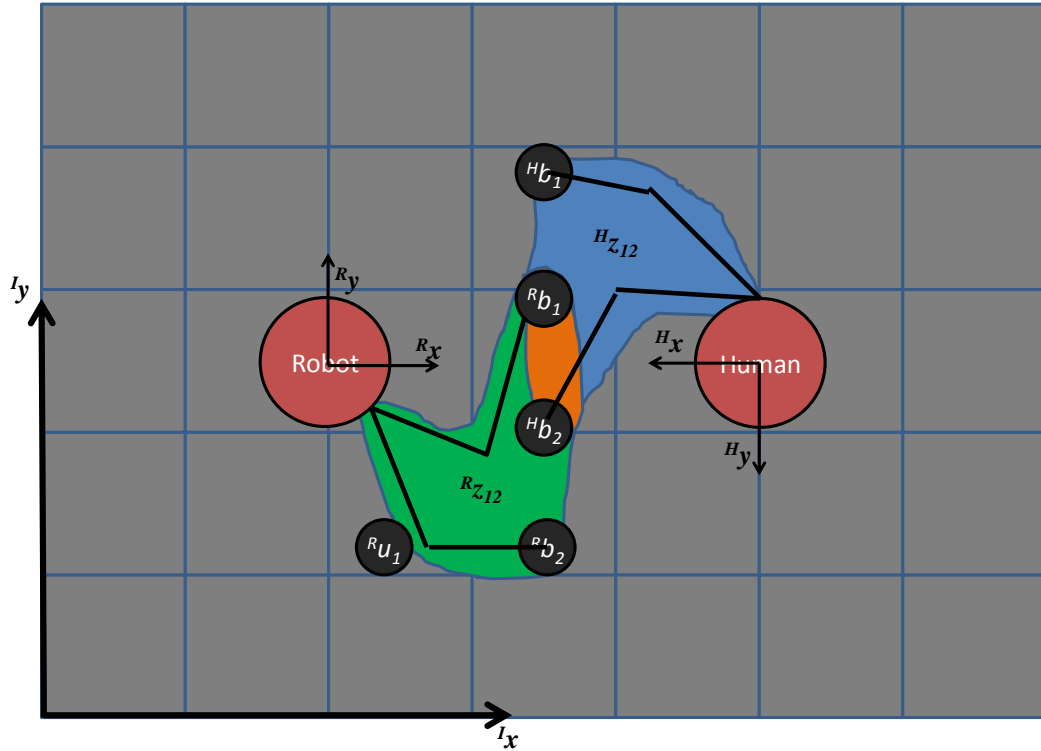


Figure 6-6: Action-zone partitioning using Definition #2, 2-D case, overhead view, example with possible-conflict case

Table 6-2 lists the robot zones from Figure 6-5 and Figure 6-6 in summary form.

Table 6-2: Zone partitioning using Definition #2, robot's zones as related to button 1 (R_{b1}), button 2 (R_{b2}), and unstow position (R_{u1})				
Zone #	Relation	R_{b1}	R_{b2}	R_{u1}
R_{z12}	transit	X	X	
R_{z2u}	transit		X	X
R_{z1u}	transit	X		X
R_{zu}	at (endpoint)			X
R_{z1}	at (endpoint)		X	
R_{z2}	at (endpoint)	X		

The human zones from Figure 6-5 and Figure 6-6 are listed in Table 6-3 in brief form.

Zone #	Relation	Hb_1	Hb_2
${}^H_{z_{12}}$	transit	X	X
${}^H_{z_1}$	at (endpoint)	X	
${}^H_{z_2}$	at (endpoint)		X

Zones with the same subscripts between the human and robotic agents do not necessarily cover the same areas (e.g., ${}^H_{z_{12}}$ does not demarcate the same 2-D space as ${}^R_{z_{12}}$), nor do they necessarily overlap each other (e.g., ${}^H_{z_1}$ versus ${}^R_{z_1}$). These zones do not move during the RAC MDP epoch, and are anchored in physical space relative to the inertial frame due to the fixed-base assumption, though transformations can be made to other base frames. During RAC MDP operations, the safety of the human (due to the robot's motion) is only communicated through the discretized danger index term, while the action-zones are underlying physical concepts that are state annotations to the actions. These zones are not impacting the complexity of the MDP except through how these two sets of attributes are chosen and defined.

Table 6-4 and Table 6-5 show how action-zones in this example map to the human and robot actions, and Table 6-5 shows the conflicts between zones. The initial occupation region and transit are captured in the initial state (in-progress or $b^i=0$), but not the final state after the action completes ($b^i=1$).

Table 6-4: Domain Representation of human actions ${}^H a_x^i, (\{ {}^H a_{obs}^i, {}^H a_{HIP}^i \})$

Discrete Value	Corresponding Action	Corresponding Action-Zone
1	no_op	??
2	move_b2_press_b1 (${}^H a_2$)	<i>in-progress</i> $\rightarrow {}^H_{z_{12}}$ <i>completed</i> $\rightarrow {}^H_{z_1}$
3	move_b1_press_b2 (${}^H a_1$)	<i>in progress</i> $\rightarrow {}^H_{z_{12}}$ <i>completed</i> $\rightarrow {}^H_{z_2}$

Table 6-5: Domain Representation of (robot) actions a_k

Discrete Value	Corresponding Action	Corresponding Action-Zone	Conflicts with Human Action-zone
0	unknown (a_0)	??	??
1	move_unstow_press_b1 (a_1)	$b^i=0 \rightarrow^R z_{1u}$ $b^i=1 \rightarrow^R z_1$	$H z_{12}$ $H z_{12}$
2	move_b2_press_b1 (a_1)	$b^i=0 \rightarrow^R z_{12}$ $b^i=1 \rightarrow^R z_1$	$H z_{12}$ $H z_{12}$
4	move_b1_press_b2 (a_2)	$b^i=0 \rightarrow^R z_{12}$ $b^i=1 \rightarrow^R z_2$	$H z_{12}$ --
5	move_unstow_press_b2 (a_2)	$b^i=0 \rightarrow^R z_{2u}$ $b^i=1 \rightarrow^R z_2$	-- --
6	move_b1_unstow (a_3)	$b^i=0 \rightarrow^R z_{1u}$ $b^i=1 \rightarrow^R z_u$	$H z_{12}$ --
7	move_b2_unstow (a_3)	$b^i=0 \rightarrow^R z_{2u}$ $b^i=1 \rightarrow^R z_u$	-- --
8	no_op (a_4)	varies	varies

6.4.1.3 Zones for Use in Case Studies

In our IVA case studies, we use the experimental setup of Chapter 3: a human is eating chips, drinking soda, and solving math problems, while the robot attempts to press buttons without causing conflict.

We use Definition #2 for the action-zones in our case studies. However, for simplicity, the case studies presented in this chapter do not contain separate “transit” zones, instead only capturing human and robot locations at action completion. This is because, for our specific case study, conflict only occurs at the edges of the shared workspace close to the completed $b^i=1$ pose locations. The robot moves quickly away from potential conflict ($d^i=1$), so long as the next action a_k chosen is non-conflicting. Thus, only one general trajectory is associated with each action. As in Chapter 3, we assume that “no-op” for the human is the pose used for solving math problems; for the robot, “no-op” is the last known zone the robot was in (a^i).

Table 6-6: Human zone partitioning using Definition #2				
Zone #	Relation	<i>chips</i>	<i>soda</i>	<i>math</i>
H_{z_1}	to	X		
H_{z_2}	to		X	
H_{z_3}	to			X
H_{z_4}	at	X		
H_{z_5}	at		X	
H_{z_6}	at			X

Table 6-7: Robot zone partitioning using Definition #2					
Zone #	Relation	b_1	b_2	<i>unstow</i>	Conflicts with Human Action-zone
R_{z_1}	to	X			H_{z_3}, H_{z_6} (<i>math</i> , mental conflict)
R_{z_2}	to		X		H_{z_1}, H_{z_4} (<i>chips</i> , physical conflict)
R_{z_3}	to			X	--
R_{z_4}	at	X			H_{z_3}, H_{z_6} (<i>math</i> , mental conflict)
R_{z_5}	at		X		H_{z_1}, H_{z_4} (<i>chips</i> , physical conflict)
R_{z_6}	at			X	--

The human and robot zones, and potential conflicts between them, are specified in Table 6-7 for the case study presented below.

6.4.2 Case Study #1 – IVA scenario, with and without human state input

In this case study, a simplified RAC MDP is developed to determine the relative performance of using RAC with no knowledge of human location versus perfect HIP information.

6.4.2.1 States and Actions

States and actions for this IVA case are simple and consistent with those from previous examples. Table 6-8 shows the human action set.

Table 6-8: Domain Representation of human actions ${}^H a_x^i, (\{ {}^H a_{obs}^i, {}^H a_{HIP}^i \})$

Discrete Value	Corresponding Action	Corresponding Action-Zone
0	unknown (a_0)	?? (worst-case against robot action chosen)
1	eat_chips (a_1)	${}^H z_1$
2	drink_soda (a_2)	${}^H z_2$
3	computer_work (a_3)	${}^H z_3$
4	no_op (a_4)	${}^H z_3$

In experiments from Chapter 3, a first-in-first-out (FIFO) queue served as the human’s and robot’s action “scripts”, with scenarios scripted such that some had conflicts while others did not. Goals on the queue were removed once completed, and goals were *temporarily* skipped if they were ‘blocked’ due to a physical or mental conflict with the human (e.g., the robot physically blocks the human from reaching a target or visually distracts within or occludes an essential viewing area).

The above scenario was created by manually specifying the RAC script and reaction “policy”. Here we ask the RAC MDP to build the policy that offers the most reward based on real-time observer and HIP MDP policy inputs. To specify the RAC MDP described in this chapter, FIFO queue priorities are directly mapped to the relative reward weightings. The ‘return to unstow position’ goal is a lower priority mission goal that is nominally overridden by any button-pushing high-priority goal. Constructing the problem in this manner requires that a new RAC MDP be specified and executed for every queue combination; this is expected as button events are not predictable, and the reward weights are relative. Table 6-9 describes the RAC goals used for our domain.

Table 6-9: Domain Representation of g_z^i, f_z^i goal-objectives			
Goal Objective	Discrete Value	Goal Definition	Corresponding Action
g_1^i	{0,1}	?at_unstowed_location?	return_to_unstow
f_1^i	{0,1}	?b1_inactive?	press_b1
f_2^i	{0,1}	?b2_inactive?	press_b2

Table 6-10: Domain Representation of (robot) actions a_k			
Discrete Value	Corresponding Action	Corresponding Action-Zone	Generally Conflicts With Human Action
1	no_op	varies (${}^R z^{i-1}$)	varies (possibly ${}^H a_x^{i-1}$)
2	press_b1	$b^i=0 \rightarrow {}^R z_1$ $b^i=1 \rightarrow {}^R z_4$	math, ${}^H a_x^i = 3$ math, ${}^H a_x^i = 3$
3	press_b2	$b^i=0 \rightarrow {}^R z_2$ $b^i=1 \rightarrow {}^R z_5$	eat chip, ${}^H a_x^i = 1$ eat chip, ${}^H a_x^i = 1$
4	return_to_unstow	$b^i=0 \rightarrow {}^R z_3$ $b^i=1 \rightarrow {}^R z_6$	n/a n/a

Table 6-10 describes the robot’s actions with action-zone mappings. The in-progress action a_1^i is a subset of this set used for a_k – it does not include the no-op action (value 1). In this case study, we reduce the combinatorial action transition set to those that do not induce collisions along one or more paths. While button presses occur in shared workspace, the robot can also transition to an unstowed position that is outside the human’s work envelope. When moving to “unstow”, it does not matter where the robot is currently located – it will always be moving away from collision. Movements to and from button 1 create only “mental conflict”, not physical collision potential. We assume that any movement from button 2 elsewhere will move the robot away from physical conflict, while any motion toward button 2 will cause conflict under certain circumstances. ‘No-op’ means that the robot doesn’t move from its last position. Thus, ‘no-op’, in this instance, can also be folded into the ‘completed’ or ‘in-progress but waiting’ states (distinguished by b^i) for any action that the robot has performed; the robot does not need to treat this as its own separate action.

We define the RAC state space as follows when presuming full observability:

$$\begin{aligned}
s^i &= \left\{ \left\{ {}^H a_{obs}^i, {}^H a_{HIP}^i \right\}, \left\{ g_1^i, f_1^i, f_2^i, a^i, b^i \right\}, d^i \right\} \\
a^i &\in \{2,3,4\}, b^i \in \{0,1\}, a_k \in \{1,2,3,4\} \\
g_1^i &\in \{0,1\}, f_z^i \in \{0,1\}, d^i \in \{0,1\} \\
{}^H a_{obs}^i &\in \{0,1,2,3,4\}, {}^H a_{HIP}^i \in \{1,2,3,4\}
\end{aligned} \tag{6-41}$$

Without data ${}^H a_{HIP}^i$ from the HIP MDP, the state space becomes:

$$\begin{aligned}
s^i &= \{ \{ {}^H a_{obs}^i \}, \{ g_1^i, f_1^i, f_2^i, a^i, b^i \}, d^i \} \\
a^i &\in \{2,3,4\}, b^i \in \{0,1\}, a_k \in \{1,2,3,4\} \\
g_1^i &\in \{0,1\}, f_z^i \in \{0,1\}, d^i \in \{0,1\} \\
{}^H a_{obs}^i &\in \{0,1,2,3,4\}
\end{aligned} \tag{6-42}$$

where ${}^H a_{obs}^i = 0$ requires assumption of the worst-case conflict over all human action choices.

With no feedback on human state, the RAC MDP becomes:

$$\begin{aligned}
s^i &= \{ \{ g_1^i, f_1^i, f_2^i, a^i, b^i \} \} \\
a^i &\in \{2,3,4\}, b^i \in \{0,1\}, a_k \in \{1,2,3,4\} \\
g_1^i &\in \{0,1\}, f_z^i \in \{0,1\}
\end{aligned} \tag{6-43}$$

The conservative policy in this case only allows actions that never conflict with the human's workspace, reverting to the status quo where human and robot occupy separate spaces. In one series of simulations, we assume in RAC that no human exists in the workspace to generate a worst-case conflict policy in which the human is unmodeled.

6.4.2.2 Transition Probability Function

Transition probabilities for this case study are given in Equation (6-44).

$$\begin{aligned}
T(s^i, a_k, s^j) &= p(H^j | H^i) \\
&* p(g_1^j | H^i, g_1^i, a_k, H^j) * \prod_{z=1}^2 p(f_z^j | H^i, f_z^i, a_k, H^j) \\
&* p(b^j | G^i, F^i, G^j, F^j, b^i, a_k) * p(a^j | a^i, a_k) \\
&* p(d^j | H^i, H^j, a^j, b^j)
\end{aligned} \tag{6-44}$$

Goals are presumed conditionally-independent from each other but are dependent on other aspects of the current state as well as the action choice a_k .

The goal transitions can be simplified for this case study. If a high-priority goal has been completed, the system stays in the absorbing (completed goal) state with probability 1. If an action a_k does not impact a mission-goal, then that mission goal stays in the same state with probability 1. The mission goal g_1^i is a special case: it is only complete while the robot arm remains at that location. Mission goal g_1^i transitions back to 0 whenever the arm attempts to complete another goal. This low-priority (mission) goal gives the robot a preference for staying out of the way if nothing else needs to be done, if the reward is nonzero. High-priority goals are achieved then can be “forgotten.” It should be noted that this is inconsistent with Chapter 3 where high-priority (interrupt) goals could need attention (action) multiple times over a specific test scenario. For a more realistic scenario, instead of having the high-priority goal states absorbing, we could have included a 10% probability at each epoch that the state could become 0 again once set.

For the case studies where $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$ or $H^i = \{^H a_{obs}^i\}$, the transition probability function is:

$$\begin{aligned}
 T(s^i, a_k, s^j) &= p(H^j | H^i) \\
 &* p(g_1^j | g_1^i, a_k) * \prod_{z=1}^2 p(f_z^j | H^i, f_z^i, a_k, H^j) \\
 &* p(b^j | G^i, F^i, G^j, F^j, b^i, a_k) * p(a^j | a^i, a_k) \\
 &* p(d^j | H^i, H^j, a^j, b^j) \tag{6-45}
 \end{aligned}$$

For the $H^i = \emptyset$ case study, the transition probability function is:

$$T(s^i, a_k, s^j) = p(g_1^j | g_1^i, a_k) * \prod_{z=1}^2 p(f_z^j | f_z^i, a_k) \\ * p(b^j | G^i, F^i, G^j, F^j, b^i, a_k) * p(a^j | a^i, a_k) \quad (6-46)$$

$p(g_1^j | H^i, g_1^i, a_k, H^j)$ reduces to $p(g_1^j | g_1^i, a_k)$, for the case study with $g_1 = at_unstowed_location$, because no trajectory leading to g_1^i will cause conflict, regardless of the human's state.

Figure 6-7 and Figure 6-8 show the algorithmic computation of probability values for g_1^j and b^j , respectively. These functions are a compact representation of the associated conditional probability tables.

Algorithm $p(g_1^j | g_1^i, a_k)$:
 if (a_k is no-op)
 if ($g_1^j = g_1^i$)
 return 1
 else return 0
 else if (a_k is return_to_unstow)
 if ($g_1^i = g_1^j = 1$)
 return 1
 else if ($g_1^i = 0, g_1^j = 1$)
 return 1
 else return 0
 else if (a_k is not return_to_unstow)
 if ($g_1^i = g_1^j = 0$)
 return 1
 else if ($g_1^i = 1, g_1^j = 0$)
 return 1
 else return 0
 else return 0

Figure 6-7: Algorithm for calculating $p(g_1^j | g_1^i, a_k)$

Algorithm $p(b^j | G^i, F^i, G^j, F^j, b^i, a_k)$:

```

if ( $a_k$  is no-op)
    if ( $b^j = b^i$  and  $G^i = G^j$  and  $F^i = F^j$ )
        return 1
    else return 0
else if ( $b^j = 0$ )
    if (evaluation of  $a_k$  is causing  $G^i \rightarrow G^j$  and  $F^i \rightarrow F^j$ )
        return 1
    else return 0
else if ( $b^j = 1$ )
    if (completion of  $a_k$  results in  $G^i \rightarrow G^j$  and  $F^i \rightarrow F^j$ )
        return 1
    else return 0
else return 0

```

Figure 6-8: Algorithm for calculating $p(b^j | G^i, F^i, G^j, F^j, b^i, a_k)$

Values used internal to these algorithms include and are consistent with previous discussions:

$$p(a^j | a^i, a_k) = \begin{cases} 1 & \text{if } a_k = \text{"no-op"} \text{ and } a^j = a^i \\ 1 & \text{if } a^j = a_k \\ 0 & \text{otherwise} \end{cases} \quad (6-47)$$

$$conflict(a_y, {}^H a_x^i) = \begin{cases} 2 & \text{physical conflict} \\ 1 & \text{mental conflict} \\ 0 & \text{no conflict} \end{cases} \quad (6-48)$$

Conditional probabilities for human state H^j are specified in Figure 6-9 and Figure 6-10, for cases with the observer (obs) supplying input with and without HIP, respectively.

The case-specific algorithms are:

Algorithm $p(H^j | H^i)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$:
if ($^H a_{obs}^i \neq 0$) and ($^H a_{HIP}^j = ^H a_{HIP}^i$)
 if ($^H a_{obs}^j = ^H a_{obs}^i$)
 return 0.5
 else if ($^H a_{obs}^j = 0$)
 return 0.5
 else return 0
else if ($^H a_{obs}^i = 0$)
 if ($^H a_{obs}^j = 0$) and ($^H a_{HIP}^j = ^H a_{HIP}^i$)
 return 0.5
 else if ($^H a_{obs}^j = ^H a_{HIP}^i$)
 return $\frac{0.5}{4}$
 else return 0
else return 0

Figure 6-9: Algorithm for calculating $p(H^j | H^i)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$

Algorithm $p(H^j | H^i)$ for $H^i = \{^H a_{obs}^i\}$:
if ($^H a_{obs}^i \neq 0$)
 if ($^H a_{obs}^j = ^H a_{obs}^i$)
 return 0.5
 else if ($^H a_{obs}^j = 0$)
 return 0.5
 else return 0
else if ($^H a_{obs}^i = 0$)
 if ($^H a_{obs}^j = 0$)
 return 0.5
 else if ($^H a_{obs}^j \neq 0$)
 return $\frac{0.5}{4}$
 else return 0
else return 0

Figure 6-10: Algorithm for calculating $p(H^j | H^i)$ for $H^i = \{^H a_{obs}^i\}$

Figure 6-11 and Figure 6-12 similarly specify conditional probability computation for high-priority goal set f_z^j .

Algorithm $p(f_z^j | H^i, f_z^i, a_k, H^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$ and $H^i = \{^H a_{obs}^i\}$:

```

if ( $a_k$  is no-op)
    if ( $f_z^j = f_z^i$ )
        return 1
    else return 0
else if ( $f_z^i = f_z^j = 1$ )
    return 1
else if ( $f_z^i = 0$ )
    if ( $f_z^j = 1$ )
        if ( $a_k$  impacts  $f_z^i$  and  $noconflict(H^i, a_k, H^j)$ )
            return 1
        else return 0
    else if ( $f_z^j = 0$ )
        if  $not(a_k$  impacts  $f_z^i$  and  $noconflict(H^i, a_k, H^j))$ 
            return 1
        else return 0
    else return 0
else return 0

```

Figure 6-11: Algorithm for calculating $p(f_z^j | H^i, f_z^i, a_k, H^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$ and $H^i = \{^H a_{obs}^i\}$

Algorithm $p(f_z^j | f_z^i, a_k)$ for $H^i = \emptyset$:

```

if ( $a_k$  is no-op)
    if ( $f_z^j = f_z^i$ )
        return 1
    else return 0
else if ( $f_z^i = f_z^j = 1$ )
    return 1
else if ( $f_z^i = 0$ )
    if ( $f_z^j = 1$ )
        if ( $a_k$  impacts  $f_z^i$ )
            return 1
        else return 0
    else if ( $f_z^j = 0$ )
        if not( $a_k$  impacts  $f_z^i$ )
            return 1
        else return 0
    else return 0
else return 0

```

Figure 6-12: Algorithm for calculating $p(f_z^j | f_z^i, a_k)$ for $H^i = \emptyset$

Figure 6-13 specifies binary discretized danger index conditional probability computation; if no human state data is available, the discretized danger index is not observable thus not part of the state as shown above in Equation (6-43). If a conflict is expected, the discretized danger index is expected to be 1 in the next state. Coupled with the computation given in Figure 6-11, which does not allow nonzero goal transitions to a completed state when conflict is expected, this models the expected ‘pause’ state that the danger index safety implementation inside the reactive controller would induce if the robot moved too close to the human on a collision trajectory.

Algorithm $p(d^j | H^i, H^j, a^j, b^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$ and $H^i = \{^H a_{obs}^i\}$:

```

if ( $d^j=0$ ) and ( $noconflict(H^i, a^j, H^j)$ )
    return 1
else if ( $d^j=1$ ) and ( $not(noconflict(H^i, a^j, H^j))$ )
    return 1
else return 0

```

Figure 6-13: Algorithm for calculating $p(d^j | H^i, H^j, a^j, b^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$ and $H^i = \{^H a_{obs}^i\}$

Algorithm $noconflict(H^i, a, H^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$:

```

if ( $^H a_{obs}^i \neq 0$ )
    if ( $^H a_{obs}^j \neq 0$ )
        if ( $conflict(a, ^H a_{obs}^i) > 0$ )
            return false
        else return true
    else if ( $^H a_{obs}^j = 0$ )
        if ( $conflict(a, ^H a_{obs}^i) > 0$  or  $conflict(a, ^H a_{HIP}^j) > 0$ )
            return false
        else return true
else if ( $^H a_{obs}^i = 0$ )
    if ( $^H a_{obs}^j = 0$ )
        if ( $conflict(a, ^H a_{HIP}^i) > 0$ )
            return false
        else return true
    else if ( $^H a_{obs}^j \neq 0$ )
        if ( $conflict(a, ^H a_{HIP}^i) > 0$  or  $conflict(a, ^H a_{obs}^j) > 0$ )
            return false
        else return true
else return false

```

Figure 6-14: Algorithm for calculating $noconflict(H^i, a, H^j)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$

Algorithm *noconflict*(H^i, a, H^j) for $H^i = \{^H a_{obs}^i\}$:

```

if ( $^H a_{obs}^i \neq 0$ )
    if ( $^H a_{obs}^j \neq 0$ )
        if ( $conflict(a, ^H a_{obs}^i) > 0$ )
            return false
        else return true
    else if ( $^H a_{obs}^j = 0$ )
        if ( $conflict(a, ^H a_{obs}^i) > 0$  or  $conflict(a, ^H a_{obs}^j) > 0$ )
            return false
        else return true
    else return true
else if ( $^H a_{obs}^i = 0$ )
    if ( $^H a_{obs}^j \neq 0$ )
        if ( $conflict(a, ^H a_{obs}^i) > 0$ )
            return false
        else return true
    else if ( $^H a_{obs}^j \neq 0$ )
        if ( $conflict(a, ^H a_{obs}^j) > 0$ )
            return false
        else return true
    else return true
else return false

```

Figure 6-15: Algorithm for calculating *noconflict*(H^i, a, H^j) for $H^i = \{^H a_{obs}^i\}$

Figure 6-14 and Figure 6-15 show computation of the *noconflict* function used in the Figure 6-13 algorithm to compute d^j probabilities. *noconflict*(H^i, a, H^j) checks a_k against the human actions that have been accomplished to see if they might have introduced conflict with each other (e.g., if $^H a_{obs}^i$ is not “unknown” and does not change, then the human has not begun action $^H a_{HIP}^i$ in the interim between states s^i and s^j , thus a_k only needs to be compared against $^H a_{obs}^i$ for collision avoidance). *noconflict*(H^i, a, H^j) for $H^i = \{^H a_{obs}^i\}$ assumes each action will not immediately be recognized; this allows us to distinguish between repeated actions because $^H a_{obs}^i$ will be equal to 0 before any previous action is repeated. A similar assumption is made for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$.

The above algorithms provide a method to functionally compute RAC MDP state transition probabilities in lieu of a full conditional probability tensor. These functions are supplemented by a “default” function to trivially model each other “unreachable” state not represented above as an absorbing state.

6.4.2.3 Rewards

Recall that the RAC MDP reward function is given by:

$$R(s^i, a_k) = w_1 * R_1(G^i, F^i) + w_2 * R_2(d^i, a^i, b^i, a_k, H^i) + w_3 * R_3(a^i, b^i, a_k) \quad (6-49)$$

The first term R_1 gives a reward for advancing overall robot goal-completion, R_2 is a cost associated with the safety of the human, and R_3 subtracts a varying cost dependent upon the choice of action a_k (e.g., energy required to accomplish a_k).

For the case studies where $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$ or $H^i = \{^H a_{obs}^i\}$, the reward terms are:

$$R_1(G^i, F^i) = \alpha_1 * g_1^i + \sum_{z=1}^2 \beta_z * ((1 + k_z) * f_z^i - k_z) \quad (6-50)$$

$$R_2(d^i, a^i, b^i, a_k, H^i) = -w_d * d^i + r_{22}(d^i, a^i, b^i, a_k, H^i) + r_{23}(a_k, a^i, H^i) \quad (6-51)$$

$$R_3(a^i, b^i, a_k) = r_{31}(a^i, b^i, a_k) - r_{32}(a_k) \quad (6-52)$$

For the $H^i = \emptyset$ case study, the reward terms are:

$$R_1(G^i, F^i) = \alpha_1 * g_1^i + \sum_{z=1}^2 \beta_z * ((1 + k_z) * f_z^i - k_z) \quad (6-53)$$

$$R_2 = 0 \quad (6-54)$$

$$R_3(a^i, b^i, a_k) = r_{31}(a^i, b^i, a_k) - r_{32}(a_k) \quad (6-55)$$

Note that there is no R_2 term where $H^i = \emptyset$ because this model does not include human state (no H^i, d^i).

In Equation (6-51):

$$r_{22}(d^i, a^i, b^i, a_k, H^i) = \begin{cases} -w_{22} & b^i = 0, a^i \neq a_k, a_k \neq \text{"no-op"}, \text{danger_increase_flag} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (6-56)$$

An algorithm to set the danger_increase_flag in Equation (6-56) is specified in Figure 6-16 and Figure 6-17. Note that $a_k = \text{"no-op"}$ would keep the robot in the same zone (thus, no increase in d^i).

Algorithm danger_increase for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$:
 if $^H a_{obs}^i \neq 0$
 if $conflictR(a_k, ^H a_{obs}^i) > conflictR(a^i, ^H a_{obs}^i)$
 return 1
 else if $conflictR(a_k, ^H a_{obs}^i) = conflictR(a^i, ^H a_{obs}^i)$
 if $conflictR(a_k, ^H a_{HIP}^i) > conflictR(a^i, ^H a_{HIP}^i)$
 return 1
 else return 0
 else if $conflictR(a_k, ^H a_{obs}^i) < conflictR(a^i, ^H a_{obs}^i)$
 return 0
 else if $^H a_{obs}^i = 0$
 if $conflictR(a_k, ^H a_{HIP}^i) > conflictR(a^i, ^H a_{HIP}^i)$
 return 1
 else return 0

Figure 6-16: Algorithm for calculating a flag representing danger increase potential for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$

Algorithm danger_increase for $H^i = \{^H a_{obs}^i\}$:
 if $conflictR(a_k, ^H a_{obs}^i) > conflictR(a^i, ^H a_{obs}^i)$
 return 1
 else
 return 0

Figure 6-17: Algorithm for calculating a flag representing danger increase potential for $H^i = \{^H a_{obs}^i\}$

The conflict function used in this algorithm is defined as:

$$conflictR(a_y, ^H a_x^i) = \begin{cases} 2 & \text{physical conflict} \\ 1 & \text{mental conflict} \\ 0 & \text{no conflict} \end{cases} \quad (6-57)$$

Reward term r_{31} from Equation (6-52) and Equation (6-55) is given by:

$$r_{31}(a^i, b^i, a_k) = \begin{cases} 0 & a_k = \text{no-op} \\ -w_{31} & b^i = 0, a^i \neq a_k \\ 0 & \text{otherwise} \end{cases} \quad (6-58)$$

where w_{31} is a user-specified weight.

Term r_{32} from these same equations is the cost of motion, and is read from a lookup table (vector w_{32} of n_a values).

Reward term r_{23} from Equation (6-51) is computed as shown in Figure 6-18 and Figure 6-19. It also relies on the definition of conflict from Equation (6-57).

Algorithm $r_{23}(a_k, a^i, H^i)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$:
if $^H a_{obs}^i \neq 0$ and $a_k = \text{"no-op"}$
 return $-w_{231} * conflictR(a^i, ^H a_{obs}^i) - w_{232} * conflictR(a^i, ^H a_{HIP}^i)$
else if $^H a_{obs}^i \neq 0$ and $a_k \neq \text{"no-op"}$
 return $-w_{231} * conflictR(a_k, ^H a_{obs}^i) - w_{232} * conflictR(a_k, ^H a_{HIP}^i)$
else if $^H a_{obs}^i = 0$ and $a_k = \text{"no-op"}$
 return $-w_{231} * conflictR(a^i, ^H a_{HIP}^i) - w_{232} * conflictR(a^i, ^H a_{obs}^i)$
else if $^H a_{obs}^i = 0$ and $a_k \neq \text{"no-op"}$
 return $-w_{231} * conflictR(a_k, ^H a_{HIP}^i) - w_{232} * conflictR(a_k, ^H a_{obs}^i)$

Figure 6-18: Algorithm for calculating $r_{23}(a_k, a^i, H^i)$ for $H^i = \{^H a_{obs}^i, ^H a_{HIP}^i\}$

Algorithm $r_{23}(a_k, a^i, H^i)$ for $H^i = \{^H a_{obs}^i\}$:
if $a_k = \text{"no-op"}$
 return $-w_{231} * conflictR(a^i, ^H a_{obs}^i)$
else if $a_k \neq \text{"no-op"}$
 return $-w_{231} * conflictR(a_k, ^H a_{obs}^i)$

Figure 6-19: Algorithm for calculating $r_{23}(a_k, a^i, H^i)$ for $H^i = \{^H a_{obs}^i\}$

To trade relative importance of the R_1, R_2, R_3 terms, each expression can be normalized then multiplied by an overall weighting factor:

$$R(s^i, a_k) = \frac{w_1}{\max |R_1|} * R_1(G^i, F^i) + \frac{w_2}{\max |R_2|} * R_2(d^i, a^i, b^i, a_k, H^i) + \frac{w_3}{\max |R_3|} * R_3(a^i, b^i, a_k) \quad (6-59)$$

We adopt the formulation from Equation (6-59) in our case study below and test weights in the range of [0 1] with increment in weight value of either $\Delta = 0.25$ or $\Delta = 0.10$.

6.4.2.4 Simulation Results

To demonstrate use of the RAC MDP for our case study, we examine the effect of different reward weights on optimal policy actions. We compare a RAC MDP that uses both observer and HIP data (which we denote HIP+RAC), a RAC MDP that uses only

observer data (similar to how the robot reacted in Chapter 3, which we denote RAC-only), and a RAC MDP where no human state information is taken into account (blind-RAC).

We use the transition probabilities given above in Chapter 6.4.2.2 and the conflicts described in Table 6-10 in Chapter 6.4.2.1. As the transition probability function is currently formulated, buttons are never expected to turn back on once the robot presses them off. This means that we generally assume that the ‘normal’ starting scenario involves a state where both buttons need to be pressed. Because these two button press high-priority goal features are presumed conditionally-independent, the policy will select a first action based on a relative reward of pressing one button versus the other.

We use the following reward weights for this case study:

$$\alpha_1 = 0, \beta_1 = 2, \beta_2 = 1, k_1 = 3, k_2 = 2, w_1 = 1, w_2 = 1, w_3 = 1,$$

$$w_d = 1, w_{22} = 1, w_{231} = 10, w_{232} = 5, w_{31} = 0.1, w_{32} = [0, 0.2, 0.2, 0.2] \quad (6-60)$$

We set $\alpha_1 = 0$, indicating no preference to move the robot to the “unstow” position unless cost of conflict in other locations exceeds the cost of moving to unstow. The β_z values we have chosen weight button 1 to be of more importance than button 2. The k_z values reinforce this, making button 1 more costly to ignore than button 2. We weight $w_1=w_2=w_3=1$ so that the R_1 , R_2 , and R_3 terms are all treated equally and the inner balance of each term is seen. $w_d=1$ assigns cost to the danger index going high ($d^i=1$) for a conflict occurring. w_{22} penalizes an expected increase in d^i . w_{231} and w_{232} explicitly weight the risk of conflict of a_k with $^H a_{obs}^i$ and $^H a_{HIP}^i$, respectively, as shown in Figure 6-18. We weight $w_{231} > w_{232}$ to imply that avoiding conflict with the in-progress action is more important than avoiding conflict with the action predicted by HIP. w_{31} is the switching cost for executing a_k that is not a continuation of a_1 . w_{31} is small because we want a_k to change if there is an appreciable change of conflict. The lookup table for r_{32} holds the weights w_{32} for the *no_op*, *press_b1*, *press_b2*, and *return_to_unstow* actions, respectively. Note that no-op takes no effort to maintain in a space environment, as it is

not working against gravity; for simplicity, all other actions are expected to require the same level of energy expenditure.

We expand the subset of matching states of the blind-RAC policy to draw analogy to those from the HIP+RAC policy, and perform a similar expansion of states from the RAC-only policy. We then compare the action-choice a_k for the different policies.

For each state in HIP+RAC, the policy action selected is compared with the action that would be selected in this state for RAC-only and blind-RAC. Table 6-11 and Table 6-12 summarize the results.

Table 6-11: Collision spread according to policy, robot action a_k , in-progress action

Action (mc=mental) (pc=physical)	HIP + RAC policy		RAC-only policy		blind-RAC policy	
	# conflict states / total states	# states action chosen	# conflict states / total states	# states action chosen	# conflict states / total states	# states action chosen
no_op	--	0	--	0	--	0
press_b1 (mc)	0%	720	2.5%	768	12.5%	960
press_b2 (pc)	0%	480	1.25%	480	6.25%	480
return_to_unstow	--	720	--	672	--	480
	# no conflict states	total # states	# no conflict states	total # states	# no conflict states	total # states
	100%	1920	96.25%	1920	81.25%	1920

Table 6-12: Collision spread according to policy, robot action a_k , future-predicted action

Action (mc=mental) (pc=physical)	HIP + RAC policy		RAC-only policy		blind-RAC policy	
	# conflict states / total states	# states action chosen	# conflict states / total states	# states action chosen	# conflict states / total states	# states action chosen
no_op	--	0	--	0	--	0
press_b1 (mc)	15%	720	17.5%	768	20%	960
press_b2 (pc)	10%	480	10%	480	10%	480
return_to_unstow	--	720	--	672	--	480
	# no conflict states	total # states	# no conflict states	total # states	# no conflict states	total # states
	75%	1920	72.5%	1920	70%	1920

Given our reward structure that encourages pressing buttons even when the robot is “blind” to the human, it is not surprising that while conflicts in the current action are always avoided in HIP+RAC, physical conflicts are encountered for RAC-only with even more for blind-RAC. As shown in Table 6-12, some conflicts with the future (HIP) action are allowed in the HIP+RAC policy, but these are not as common as for RAC-only or blind-RAC.

Note that from our choice of conflict algorithm, out of the 1920 total states, there are potentially 960 states where a_k could induce a conflict with the in-progress action and potentially 1152 states where a_k could induce a conflict with the future-predicted action. Thus, when comparing against the percentages in Table 6-11 and Table 6-12, the worst-case numbers we could expect to see would be conflicts of {50%,60%} likelihood, not out of 100%.

Assuming the observer rapidly identifies the in-progress action, and assuming that the time needed to make this identification is less than time between policy execution iterations, the above tables accurately represent collision potential with a policy that balances potential for collision with reward for goal achievement. Next, consider two extreme reward weight cases: (1) robot greedily accomplishes goals without regard for human safety, and (2) robot is ultimately conservative, always retreating to (unstowed) safety if collision potential exists.

For this study, we use the same internal reward weights as in Equation (6-60) before:

$$\alpha_1 = 0, \beta_1 = 2, \beta_2 = 1, k_1 = 3, k_2 = 2, w_d = 1, w_{22} = 1,$$

$$w_{231} = 10, w_{232} = 5, w_{31} = 0.1, w_{32} = [0,0.2,0.2,0.2] \quad (6-61)$$

But now consider normalized terms for the reward function:

$$R(s^i, a_k) = \frac{w_1}{\max |R_1|} * R_1(G^i, F^i) + \frac{w_2}{\max |R_2|} * R_2(d^i, a^i, b^i, a_k, H^i) + \frac{w_3}{\max |R_3|} * R_3(a^i, b^i, a_k) \quad (6-62)$$

For the internal reward weightings given above in Equation (6-61) for our HIP+RAC case, the range of values for R_1 is [-8 3], for R_2 is [-6 0], and for R_3 is [-0.3 0]. Thus, our normalized equation is:

$$R(s^i, a_k) = \frac{w_1}{8} * R_1(G^i, F^i) + \frac{w_2}{6} * R_2(d^i, a^i, b^i, a_k, H^i) + \frac{w_3}{0.3} * R_3(a^i, b^i, a_k) \quad (6-63)$$

We look at the policy output for these cases and compare the value of each policy against the other (as computed within the Bellman equation). For these studies, we assume the HIP+RAC case (i.e., HIP and observer inputs are both available).

Table 6-13: Collision spread versus value, robot action a_k , in-progress action; in-progress%, future-predicted% (out of 1920 total states)

Policy # and Reward weights { w_1, w_2, w_3 }	# physical conflict states / total states (<i>press_b2</i>)	# mental conflict states / total states (<i>press_b1</i>)	# no conflict states / total states (<i>no-op</i>) (<i>return_to_unstow</i>)	Policy total value
P1 {1,0,0}	0%, 10%	0%, 15%	100%, 75%	11452
P2 {0,1,0}	0%, 0%	0%, 0%	100%, 100%	-160
P3 {1,0,1}	5.2%, 14.2%	5.7%, 19.6%	89.1%, 66.2%	9999
P4 {1,1,0}	0%, 10%	0%, 15%	100%, 75%	10978
P5 {1,1,1}	0%, 10%	0%, 15%	100%, 75%	8216

Table 6-13 shows a summary of results from this test set. Five policies were generated. The first two represent policies optimized over only 1 reward term (R_1 versus R_2). P3 ignores safety, P4 ignores action cost, and P5 weights all terms equally. For safety purposes, *return_to_unstow* was the default action selected by the policy in the event of a tie, which would remove the robot from the possibility of conflict.

Policy P1 rates the current state without regard for any upcoming conflict. It is interesting that it still produces less conflicts than the RAC-only or blind-RAC cases.

This is potentially because we chose a discount factor close to 1, so future reward impacts the present state with respect to long action sequences (i.e., actions that would cause conflict transition to a state with $d^i=1$, and while $d^i=1$ the goal will never transition to completed state, delaying future reward). Thus, because of the discount factor reducing future reward when subject to wait times, the policy chose other tasks to complete in the meantime. P2 shows that our RAC MDP policy can provide us with an absolutely-safe policy, if safety is giving overriding weight in the reward function. P3 rates state reward versus cost of motion, and in some cases the tradeoff between waiting to effect goal completion (continuing to choose the same action for no additional movement cost and pausing until the human performs a different action) and moving on to another goal (action-switching for some small additional cost) sees the robot sometimes choosing to wait in a conflict situation ($d^i=1$), and thus more conflicts occur. This verifies that the R_3 cost term does have some impact. However, the R_2 term seems to have far more impact than the R_3 term when paired with R_1 , as P5's policy is the same as the P4 policy. It is also apparent that the restrictions within the transition probability functions do favor conflict avoidance, as P1 and P4 generate similar policy output. The P5 policy matches more closely to policy P1 (chose same action 98.4% of the time) than policy P3 (chose same action 82.5% percent of the time).

It is notable that policy P1 chooses the *no-op* action 1.5625% of the time; and that when *no-op* was chosen, it never caused a conflict for either the in-progress or future-predicted cases. Policy P2, which weights safety above all else, always chose the *return_to_unstow* action, which does keep the robot from causing any conflict, but at the obvious detriment of never fulfilling any of its own goals. Policy P4 seems better balanced than P3, considering that the robot should avoid conflict with the in-progress action more carefully than the future-predicted action; P4 has higher rates of button-pushing choice with fewer expected impending conflicts. Policy P5 seems to converge to the same policy as P4. The tradeoff trend we see here is as-expected – lower value for less conflicts.

6.5 Conclusions and Discussion

We have presented a Markov Decision Process (MDP) formulation for using predicted human (astronaut) intent to inform robot action choice. The robotic manipulator arm

optimizes task completion over safety (conflict avoidance) and its own performance (robot goal completion). We have presented a case study to illustrate RAC MDP formulation. We show the relative improvement in performance when HIP+RAC is compared with models that use less or no human state information. When assuming that the observer and HIP state input are correct, the policies that leverage the most data choose safer policies. Reward weight tradeoffs verify that indeed goal achievement does come at a cost of heightened risk of conflict.

6.5.1 Feedback of RAC into HIP

As we have seen above, there are tradeoffs that can be made to give us multiple policies, some of which have a nonzero likelihood of conflict with the human in the future. We showed the worst-case estimates: if an action was attempted to completion but might cause a conflict, it was assumed to cause that conflict in the data we displayed. However, these worst-case estimates are absent of the details of timing issues, reduced manipulator speed by the reactive controller, and other mitigating factors. Thus, some flexibility on the human's part could help mitigate these circumstances, though this breaks our assumption of the human lack of situational awareness of the robot. Overall, if we wish to be able to use these policies that have higher value for the robot's goal-completion (R_I), then we need to determine when we might be able to use these policies without actually causing conflict.

The next step from this point is to realize that, while we wish the robot to be able to work nearby the human without causing conflict, if we also wish to improve robot performance beyond a certain level, there likely needs to be a tradeoff with the human's performance where the risk of physical conflict is higher than nonzero and may in fact require human response. This would require some form of communication to the human – such as the human noticing and responding to the robot once it may begin to impinge upon their nearby workspace or trajectory path. To handle this effectively within RAC, we would need to relax our assumptions that the human does not need to communicate with or will not be impacted by the robot. From this, it follows that the human now might need to internally model the robot under certain circumstances. Our HIP would need to be expanded to include a minimal representation of robot state for human reaction, similar to

the way RAC includes a minimal representation of HIP for conflict avoidance. This would likely need to only occur under conditions when there is a probability above a given threshold that there may be a conflict between the human and the robot (e.g., a higher danger index value). We would then need to investigate the consequences of doing so.

In future work, it would be best to demonstrate the utility of this feedback by demonstrating how and when HIP is wrong, in cases where no conflict with the robot is modeled in HIP but the human does react to the robot. The main characteristics of this set of circumstances could be determined through further human subject experiments using the safe robotic manipulator arm as in Chapter 3. Once these previously-unmodeled cases have been categorized, we could then update our HIP model and show through demonstration how the new HIP-with-feedback improves the robot's actions and better matches the human's actual reactions. The goal of these experiments would be to identify (1) the minimal set of new attributes that would need to be fed back into HIP from RAC and (2) the form of the functions that would use this information within the HIP model. The relative utility of several different strategies to "minimally inform" RAC could then be evaluated.

6.5.2 Comparison of primarily-scripted HIP+RAC to A*, POMDP, or other methods

When there is very little uncertainty in the human's actions, or the entire scenario is highly-scripted as on EVA with few surprises (one or no high-priority goals that might activate with no warning / cannot be predicted in-advance), the performance and safety increase that using HIP+RAC might provide may not be highly advantageous over using other methods. Also, goal objectives in our MDP are similar to those in a Hierarchical Task Network (HTN) and satisfaction of these goals is through execution of primitive tasks. The constraints among the tasks could be automatically generated and specified using a HTN planning approach. Determining the task breakdowns from goal tasks to the compound and primitive tasks that will complete them is a difficult but solvable problem. We can then determine how to encode each of these tasks in the model, whether as binary goals or actions, and the necessary action-history. Determining how to automatically

encode this information within our MDP's, and determining when the complexity of including such information, suggests use of simpler deterministic search can be explored in future work.

Alternately, the more uncertainty there is in the outcomes, the more useful we would expect HIP+RAC to be when compared to deterministic methods. However, as uncertainty increases further and HIP modeling assumptions break down, use of a POMDP for HIP or a combined POMDP might give better results (assuming a small state space size). It would be prudent to compare the HIP+RAC split setup to an equivalent POMDP setup, to determine how and when HIP+RAC breaks down (optimality of policy solution sharply decreases) because the separated models are reduced to sharing only minimal information between them. To explore the possibility of using POMDP formulations instead of MDP formulations, the HIP input to RAC would then be linked to a belief state. Future work should explore these possibilities.

However, it should be noted that we would not wish to replace HIP+RAC with either a combined MDP or a combined POMDP if it proves unnecessary. For example, for the HIP+RAC case study explored here, a combined MDP would necessarily need to include at least $(2^3 * 5 * n_h) * (2^1 * 2^2 * 4^1 * 2^1 * 2^1)$ states. For $n_h=4$, that's 640,000 states. If we stopped making the simplifying assumption that we don't have to note the starting point for our robot motions, then the number of robot actions we need to track jumps from 4 up to 16. For HIP+RAC, our HIP model would remain the same size, and our RAC MDP would have 12,800 states (instead of 1,920), which would still be feasible to calculate. For the combined MDP, that would be 2,560,000 states, which would already begin to run up against memory and other computation issues for things like the full transition probability tensor (2,560,000 x 2,560,000 x 5 possible robot actions $a_k = 3.2768 * 10^{13}$).

It is likely that, when our input noise is on the order of a random spread, it would no longer be advantageous to use HIP; in this case, we should instead only use the observer's in-progress action rather than trying to look that far ahead. It is the knowledge of the structured environment – and the human's predictability acting within it – that allows us to model the human and make the HIP output useful in some way. We could look at the results of the injection of 'noise' into the system, using Monte Carlo

simulations to test the robustness of the current RAC MDP formulation. Pushing RAC to fail would likely give us a good indication of the states in which it would most benefit the robot to request additional state information from the human, allowing us to refine the RAC model further. This would allow us to help characterize how robust the RAC MDP is to an inaccurate HIP model, given that we currently assume that HIP is 100% correct in its predictions. RAC MDP’s robustness constraints also determine the modeling accuracy to which HIP must conform. This accuracy is the most-important metric for a realistic HIP implementation, informing us whether our (or other) methods for HIP are viable to use with RAC. In fact, because of the modular architecture used (see Chapter 4), any viable method for HIP could theoretically be used to supply human predicted intent, provided that it meets the constraints that RAC requires.

6.5.3 Markov chains for progression of robot action choice

While we do compute policies for the entire state space, some states are more likely to occur than others. It may be interesting to look at a few of the most likely starting states and look at the progression of change, following the tree structure of all of the possible $s^i \rightarrow s^j$ transitions for the optimal policy actions, to see how the states progress forward in time for up to four levels of human state input H^i . We may also want to relax some of our current assumptions regarding the button activations, allowing a small (10% or less) likelihood for a button transitioning back to an active state.

6.5.4 Differing choice of R_2 algorithm

Currently, we use a function which gives varying cost to physical and mental collisions according to whether they occur as an in-progress or future-predicted action. If we wished to further suppress conflict, we could replace the current r_{23} algorithm being used with one that will give a constant negative weight if any future-conflict is seen for either upcoming human action. Alternately, we could use a version of r_{23} that does not differentiate between the type of conflict (e.g., physical and mental conflicts are treated as garnering equal cost).

Physical and mental conflicts could also be treated differently in both the r_{22} and r_{23} terms, with relative weight assigned to the severity of their impact on d^i . This might be useful for cases where we might consider mental collisions to be ‘safe but annoying’ due

to gross impingement into the human’s field-of-view necessary for task completion; mental conflict does not always preclude a physical collision. Mental collisions could be given a lower cost than physical collisions.

6.5.5 Impact of allowing reactive controller to handle conflict resolution ‘intelligently’

In future work, it would be interesting to explore the benefits and drawbacks of using either further-refined action-zones or a trinary danger index. We could subdivide the action-zones into more parts and extend the number of actions accordingly. This could be done by using the danger index to calculate the breakdown of a goal-driven trajectory into multiple trajectory pieces, each new action with their own action-zone and separate mapping of danger index value DI to $d^i=1$. In effect, every time the RAC MDP model completes a piece of the original trajectory, the robot would pause and reconsider continuing the same action. A trinary d^i would allow the robot to choose actions that might cause conflict with the human. We would expect high weighting on d^i to result in recoil when close to the human; low weighting on d^i with high reward on goal achievement could cause the robot to “stay and wait” for non-binary d^i .

While subdividing the action-zones does increase the size of the state space more quickly than using an integer-valued d^i , it is also very straightforward and does not lead to timing issues that might otherwise arise with an integer-valued d^i . For example, it is difficult to determine how likely is it that the human will complete their action with respect to the value of d^i given when we do not necessarily know how far through their action a human has already progressed at any point in time. We can only infer this from the value of d^i , and $DI=0$ when there is no relative motion!

We also currently assume that transition probabilities for $p(d^j|H^i, R^i, d^i, a_k, H^j, R^j)$ are binary; this assumption could be relaxed and we could take advantage of the fact that the reactive controller will, at base level, make sure that the robot’s motions will not be unsafe. A similar assumption which could be relaxed is the use of binary percentages for the likelihood of interruption occurring in the transition probability function.

6.5.6 Similar state spaces, same or different transition probability and reward functions

The MDP state space might look the same in the general case for a scenario when they include the same goals and actions, and same basic environmental characteristics. However, because of the relative arrangement of the robot, human, and goal locations, the transition probabilities and reward function may end up being very different. One example of a different reward function would be as briefly discussed above: varying FIFO queue rewards associated with different buttons having different priority levels or activation times. Transition probabilities may change most readily if there are blockages within the workspace, or higher probabilities of conflict due to more overlap between some of the human's and robot's trajectories.

6.5.7 Explicit zone calculations and mappings

It may be noted that we have not gone into great detail in discussing how to calculate the 3-D envelopes in space for the zone attributes. This is a nontrivial task and is completely dependent upon the robot (type, size, speed, and maneuverability), the layout of the workspace, and the human's location relative to the robot within the workspace (and their range of motion). The simplest way to determine a good first-cut approximation of each motion-trajectory envelope would be to determine end poses to every goal for each agent, calculate trajectories between the combinatorial set of them, and expand a bubble outwards around each trajectory using the danger index calculation. Once this has been accomplished, overlapping areas of conflict can be determined, and zones could be further subdivided according to chosen ranges of increasing danger index. We consider this an exercise to be demonstrated in future work prior to implementation on a particular robotic platform for new human subject experiment studies.

6.5.8 Relaxation of assumption of perfect HIP information

Currently, in Chapter 6.2.2.1, we do assume that the observer and that HIP give perfect information to RAC in our case studies. Note, however, that we could relax our second assumption and not assume that HIP is perfect. If we did this, then transitions from Step 1 to Step 2 would need to be spread across all states $\{^H a_{obs}^i = 0, ^H a_{HIP}^i = *\}$ with

unequal probability, and transitions from Step 2 to Step 1 would need to spread across all states $\{^H a_{obs}^i = *, ^H a_{HIP}^i = *\}$ with unequal probability.

6.5.9 Relaxation of fixed-base assumption

A second scenario we might wish to investigate is a case simplified from the EVA example from Chapter 5 for HIP in a highly-scripted space – working on a repair panel. The robot in this case would be free-floating and performing similar tasks of its own – video recording duties and supplying differing views of the workspace. The astronaut would still be fixed-base. We would explore the differences in zone specification (if any) and the effects that this expanded space would have on the state space formulation and robustness in terms of the safety of the policy actions, and what changes in reward terms may be necessary to assure this. A free-floating platform should have higher restrictions on certain trajectories that have it moving into and out of view of a suited astronaut while nearby their head or shoulders, for instance.

Chapter 7

Conclusions and Future Research Directions

7.1 Summary and Conclusions

Astronauts in a space environment are exposed to risk on both EVA and IVA. Risk could be reduced and overall productivity increased with the introduction of autonomous human-aware robotic systems that can perform HRI without requiring either teleoperation or close supervision. We hypothesize that disallowing explicit communication to reduce the astronaut's mental workload and increase productivity will not introduce unacceptable levels of risk during shared workspace operations, if human intent prediction is used to determine the human's near-term goal-based actions from the astronaut's rational motions so that the robot can avoid potential conflict with the human.

Before determining whether human intent prediction would be useful, we first ran human subject experiments to determine whether a semi-autonomous manipulator arm, when sharing a workspace with a human, would impact human productivity negatively if it was able to react to the human's actions and avoid short-term conflict. We then created a framework that supports safe human-aware HRI through the use of two separate Markov Decision Processes (MDPs) for human intent prediction (HIP) and robot action choice (RAC), respectively, and designed the system to require only a minimal amount of information sharing. We determined a MDP formulation that includes what we believe to be the least number of elements necessary for HIP to be useful; we also determined a similar formulation for RAC, focusing more on the choice of reward function metrics, such as Kulic's danger index, to allow a direct safety-efficiency tradeoff.

Our conclusions are the following:

- Human subject testing supports the theory that the inclusion of a robot into a human's workspace will not degrade or otherwise impact human performance, so long as the robot does not create conflict within the shared workspace.

- There are viable frameworks for supporting autonomous human-robot interaction in close-quarters collaborative space-environment settings, where maintaining safety is key; we have proposed such a framework based on decomposition of the problem such that we can model a HRI scenario as two separate MDPs for HIP and RAC that together recognize, understand, and exploit knowledge of human intent for robot task planning.
- Our simulation results from the HIP MDP models give consistent and understandable policies that are not overly sensitive to small variations in the reward function weightings.
- Our RAC MDP formulation incorporates HIP and direct observations into a more-informed state on which RAC is based. Case studies demonstrated the possible tradeoffs in safety and efficiency, although baseline safety is consistently maintained through a reactive collision avoidance capability.

7.2 Future Work

The simple case studies presented in Chapter 5 and 6 only begin to demonstrate and evaluate the capabilities of the decoupled HIP+RAC architecture presented in Chapter 4. Chapters 5 and 6 present extensive discussion of future work to further mature and evaluate HIP and RAC, respectively. Here we focus on future work at the integrated architecture level.

To validate the proposed framework, the first step is to run a full dynamic simulation of the integrated system, creating a ‘simulated reality’ to test the RAC output responses and timing effects when integrated with the HIP MDP. Differences between policy expectations and actual output can also be compared for cases where the simulated behavior matches the ideal HIP+RAC system model versus cases where some of the conditions of HIP have been relaxed, resulting in discrepancies between predicated and actual human intent. This can be done initially with simulated noise, and later using actual human subject experiments.

Our most significant assumption is that human intent can be predicted independent of robot activity, i.e., that there is no feedback of RAC into HIP, only feedforward from HIP into RAC. In reality, overall productivity might be enhanced by allowing situations in

which the robot can choose actions with potential to distract its human companion, in which case these actions must be considered for accurate HIP.

It will also be useful to examine impact of using a finite-horizon MDP formulation, or reduction in discount factor, on the policy output. Realistic HIP and RAC models will be more complex than those presented in case studies, which also will require careful knowledge engineering to appropriately abstract and decompose tasks into multiple MDPs per the discussion in Chapter 4. It will be critical to ensure HIP policies are consistent with expected human behavior, a property that is only possible to validate in human subject experiments. Such experiments can also provide insight into how different models for HIP+RAC compare with each other.

Expansion to a more collaborative domain, potentially with shared goals as well as feedback from RAC to HIP, would allow us to evaluate tradeoffs between expressiveness of a fully-coupled model for HIP+RAC versus the greater simplicity enabled by assuming the human need not consider the robot in decision-making. Such a study might also yield insight on when it would be beneficial for the robot to directly communicate with a human companion or supervisor. Finally, it will be essential to benchmark the proposed HIP+RAC strategy against alternate robot decision-making strategies. Quantitative metrics for such an evaluation include mission goal achievement (and time to completion) for both human and robot, HIP accuracy, and safety (how many times did conflicts emerge, and how many times (if any) were they not automatically resolved). Qualitative metrics will also be important, particularly with respect to the human subject's perceived workload and attitude toward (e.g., trust of) its robot companion. For all benchmarks, computational overhead is also an important evaluation metric. While the MDP is more tractable than the POMDP, both have complexity related to the number of MDP/POMDP states which, as illustrated in presented case studies, can quickly grow to an unmanageable size.

Appendices

Appendix A

MichiganMan(ipulator) Arm Characteristics

Physical MM-arm Design (and differences upon construction)

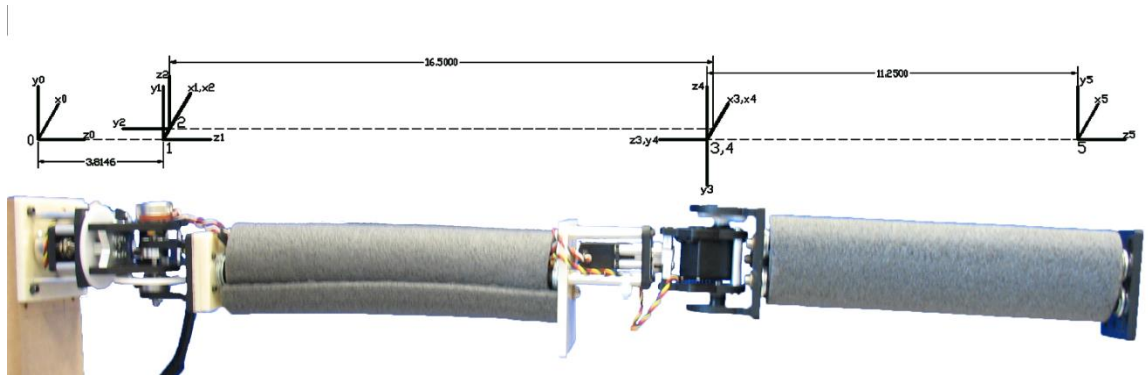


Figure A 1: MichiganMan(ipulator) arm illustrating Denavit-Hartenberg Kinematic Parameters.

The MM arm is a 4 degree-of-freedom (4-DOF) revolute R-P-R-P arm (roll-pitch shoulder, roll-pitch elbow), with an extension for possible future accommodation of a wrist. This arm was made with similar scale to a human arm and similar shoulder/elbow design to a Robonaut arm (see reference: Nickels, IEEE presentation), with differences enumerated in the below table. Note that the arm is designed using the Denavit-Hartenberg (D-H) notation convention; also note that the joint 1 and 3 axes are aligned when joint 2 is at its “nominal” 0 degree angle.

Table A 1: MM-arm versus Robonaut D-H Parameters

Distance...	Robonaut	MM-arm	explanation of differences
from base to joint 1	12"	~3.8"	keeps structural stability; negligible
from joint 2 to joint 3	14.5"	~16.5"	joint design changed; could be accommodated by change in link length; negligible
from joint 4 to tooltip	14.5"	~11.25"	accommodates future addition of wrist; negligible
between joint 1 and joint 2	2.5"	~0.345"	differences in joint sizing/alignment; negligible
between joint 3 and joint 4	2"	~0"	differences in joint sizing/alignment; negligible

The following table specifies the ranges of motion of each of the four MM-arm joints. Note that in Figure A 1, the arm is in a position where all joint angles are zeroed.

Table A 2: Joint Ranges for MM-arm

Joint number	Joint range (in degrees; counter-clockwise = positive)	
	minimum	maximum
1	-90	90
2	-90	45
3	-90	90
4	-45	90

Specific Measured D-H parameters of the Michigan Manipulator

The (approximate) D-H parameters for the MM-arm are given in the following table:

Table A 3: MM-arm D-H Parameters

i	α_{i-1} (degrees)	a_{i-1} (inches)	d_i (inches)	θ_i (degrees)
1	0	0	3.814	θ_1
2	-90	0.345	0	θ_2
3	-90	-0.345	-16.5	θ_3
4	90	0	0	θ_4

Note that for our implementation, we specify a transformation to frame 5, the tooltip frame, as:

$${}^4_5T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A-1})$$

where $d_5=11.25$ (inches).

Forward Kinematics Equations

Using the substitutions:

$$\begin{aligned} s_1 &= \sin(\theta_1) \\ c_2 &= \cos(\theta_2) \\ s_{(a+b)} &= \sin(a+b) \end{aligned} \quad (\text{A-2})$$

The forward kinematic transformation matrix (base to tooltip) for the MM-arm is given by:

$${}^0_5T = \begin{bmatrix} (c_1c_2c_3 + s_1s_3)c_4 - c_1s_2s_4 & -(c_1c_2c_3 + s_1s_3)s_4 - c_1s_2c_4 & c_1c_2s_3 - s_1c_3 & 11.25((c_1c_2c_3 + s_1s_3)s_4 + c_1s_2c_4) \\ (s_1c_2c_3 - c_1s_3)c_4 - s_1s_2s_4 & -(s_1c_2c_3 - c_1s_3)s_4 - s_1s_2c_4 & s_1c_2s_3 + c_1c_3 & -0.345(c_1c_2 - c_1) + 16.5c_1s_2 \\ s_2c_3c_4 - c_2s_4 & s_2c_3s_4 - c_2c_4 & -s_2s_3 & 11.25((s_1c_2c_3 - c_1s_3)s_4 + s_1s_2c_4) \\ 0 & 0 & 0 & -0.345(s_1c_2 - s_1) + 16.5s_1s_2 \\ & & & 3.814 - 11.25(s_2c_3s_4 - c_2c_4) \\ & & & + 0.345s_2 + 16.5c_2 \\ & & & 1 \end{bmatrix} \quad (\text{A-3})$$

Inverse Kinematics Equations (numerical solution methods for generalized solutions / position-only waypoint)

We give a derivation for finding an algebraic solution for the MM-arm below. The following matrix math equations were computed using the transformation matrix math above and the D-H parameters given earlier in the appendix. Nonzero a_{i-1} , d_i , and θ_i parameters are treated as variables; zeroed parameters and the α_{i-1} terms are substituted to simplify the equations. Tractable closed-form solutions are identified for joint angles

θ_i from the given waypoint, when all parameters $r_{11}, r_{12}, r_{13}, r_{21}, r_{22}, r_{23}, r_{31}, r_{32}, r_{33}, p_x, p_y, p_z$ in the 0_5T transformation matrix are assumed known (in other words, when we know the position and rotation of the tooltip frame in the origin frame in 3D space).

Using this convention for shorthand substitutions:

$$\begin{aligned} s_1 &= \sin(\theta_1) \\ c_2 &= \cos(\theta_2) \\ s_{(a+b)} &= \sin(a+b) \end{aligned} \quad (\text{A-4})$$

Consider the following informal derivation, starting with:

$$[{}^0_1T]^{-1}[{}^0_5T] = [{}^1_5T] \quad (\text{A-5})$$

Using the general transformation matrix for $[{}^0_1T]^{-1}$ and computing $[{}^1_5T]$ using the kinematic equations, we get:

$$\begin{aligned} & \begin{bmatrix} c_1 & s_1 & 0 & 0 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_2c_3c_4 - s_2s_4 & -c_2c_3s_4 - s_2c_4 & c_2c_3 & (-c_2c_3s_4 - s_2c_4)d_5 + c_2a_2 - s_2d_3 + a_1 \\ -s_3c_4 & s_3s_4 & c_3 & s_3s_4d_5 \\ -s_2c_3c_4 - c_2s_4 & s_2c_3s_4 - c_2c_4 & -s_2s_3 & (s_2c_3s_4 - c_2c_4)d_5 - s_2a_2 - c_2d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A-6}) \end{aligned}$$

Multiplying the right hand side, we get:

$$\begin{aligned}
& \begin{bmatrix} c_1 r_{11} + s_1 r_{21} & c_1 r_{12} + s_1 r_{22} & c_1 r_{13} + s_1 r_{23} & c_1 p_x + s_1 p_y \\ -s_1 r_{11} + c_1 r_{21} & -s_1 r_{12} + c_1 r_{22} & -s_1 r_{13} + c_1 r_{23} & -s_1 p_x + c_1 p_y \\ r_{31} & r_{32} & r_{33} & p_z - d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
= & \begin{bmatrix} c_2 c_3 c_4 - s_2 s_4 & -c_2 c_3 s_4 - s_2 c_4 & c_2 c_3 & (-c_2 c_3 s_4 - s_2 c_4) d_5 \\ & -s_3 c_4 & s_3 s_4 & c_3 & s_3 s_4 d_5 \\ -s_2 c_3 c_4 - c_2 s_4 & s_2 c_3 s_4 - c_2 c_4 & -s_2 s_3 & (s_2 c_3 s_4 - c_2 c_4) d_5 \\ & 0 & 0 & 0 & -s_2 a_2 - c_2 d_3 \\ & 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A-7})
\end{aligned}$$

Note that:

$$\begin{aligned}
r_{32} &= s_2 c_3 s_4 - c_2 c_4 \\
p_z - d_1 &= (s_2 c_3 s_4 - c_2 c_4) d_5 - s_2 a_2 - c_2 d_3
\end{aligned} \quad (\text{A-8})$$

Substituting and rearranging terms, we get

$$\begin{aligned}
p_z - d_1 &= r_{32} d_5 - s_2 a_2 - c_2 d_3 \\
s_2 a_2 + c_2 d_3 &= r_{32} d_5 - p_z + d_1
\end{aligned} \quad (\text{A-9})$$

Rewrite as:

$$s_2 "P_x" + c_2 "P_y" = K \quad (\text{A-10})$$

where

$$\begin{aligned}
"P_x" &= a_2 \\
"P_y" &= d_3 \\
K &= r_{32} d_5 - p_z + d_1
\end{aligned} \quad (\text{A-11})$$

Then, employ a strategic change of variables

$$\begin{aligned} P_x &= \rho \cos(\varphi) = a_2 \\ P_y &= \rho \sin(\varphi) = d_3 \end{aligned} \quad (\text{A-12})$$

where

$$\begin{aligned} \rho &= \sqrt{P_x^2 + P_y^2} = \sqrt{a_2^2 + d_3^2} \\ \varphi &= \tan^{-1}(P_y, P_x) = \tan^{-1}(d_3, a_2) \end{aligned} \quad (\text{A-13})$$

The equation

$$s_2 P_x + c_2 P_y = K \quad (\text{A-14})$$

then becomes

$$s_2 c_\varphi + c_2 s_\varphi = \frac{K}{\rho} \quad (\text{A-15})$$

Using the trigonometric identity

$$s_a c_b + c_a s_b = s_{(a+b)} = \sin(a+b) \quad (\text{A-16})$$

the equation

$$s_2 c_\varphi + c_2 s_\varphi = \frac{K}{\rho} \quad (\text{A-17})$$

becomes

$$s_{(\theta_2 + \varphi)} = \frac{K}{\rho} \quad (\text{A-18})$$

Using the trigonometric identity

$$(s_a)^2 + (c_a)^2 = 1 \quad (\text{A-19})$$

the equation

$$(s_{(\theta_2+\varphi)})^2 + (c_{(\theta_2+\varphi)})^2 = 1 \quad (\text{A-20})$$

becomes

$$\left(\frac{K}{\rho}\right)^2 + (c_{(\theta_2+\varphi)})^2 = 1 \quad (\text{A-21})$$

$$c_{(\theta_2+\varphi)} = \pm \sqrt{1 - \frac{K^2}{\rho^2}}$$

Since

$$\tan(\theta_2 + \varphi) = \frac{s_{(\theta_2+\varphi)}}{c_{(\theta_2+\varphi)}} \quad (\text{A-22})$$

we find

$$\theta_2 + \varphi = \tan^{-1}\left(\frac{K}{\rho}, \pm \sqrt{1 - \frac{K^2}{\rho^2}}\right)$$

$$\theta_2 + \tan^{-1}(d_3, a_2) = \tan^{-1}\left(\frac{K}{\rho}, \pm \sqrt{1 - \frac{K^2}{\rho^2}}\right) \quad (\text{A-23})$$

$$\theta_2 = \tan^{-1}\left(\frac{K}{\rho}, \pm \sqrt{1 - \frac{K^2}{\rho^2}}\right) - \tan^{-1}(d_3, a_2)$$

which is a closed-form solution for θ_2 , since K, ρ, d_3, a_2 are all known (see above).

From Equation (A-7), also note that:

$$c_1 r_{12} + s_1 r_{22} = -c_2 c_3 s_4 - s_2 c_4 \quad (\text{A-24})$$

$$c_1 p_x + s_1 p_y = (-c_2 c_3 s_4 - s_2 c_4) d_5 + c_2 a_2 - s_2 d_3 + a_1$$

Substituting and rearranging terms, we get

$$\begin{aligned} c_1 p_x + s_1 p_y &= (c_1 r_{12} + s_1 r_{22}) d_5 + c_2 a_2 - s_2 d_3 + a_1 \\ c_1 (p_x - r_{12} d_5) + s_1 (p_y - r_{22} d_5) &= c_2 a_2 - s_2 d_3 + a_1 \end{aligned} \quad (\text{A-25})$$

Using the same method as before, rewrite as:

$$s_1 "P_{x2}" + c_1 "P_{y2}" = K_2 \quad (\text{A-26})$$

where

$$\begin{aligned} "P_{x2}" &= p_y - r_{22} d_5 \\ "P_{y2}" &= p_x - r_{12} d_5 \\ K_2 &= c_2 a_2 - s_2 d_3 + a_1 \end{aligned} \quad (\text{A-27})$$

Then, using a strategic change of variables

$$\begin{aligned} "P_{x2}" &= \rho_2 \cos(\varphi_2) = p_y - r_{22} d_5 \\ "P_{y2}" &= \rho_2 \sin(\varphi_2) = p_x - r_{12} d_5 \end{aligned} \quad (\text{A-28})$$

where

$$\begin{aligned} \rho_2 &= \sqrt{"P_{x2}"^2 + "P_{y2}"^2} = \sqrt{(p_y - r_{22} d_5)^2 + (p_x - r_{12} d_5)^2} \\ \varphi_2 &= \tan^{-1}("P_{y2}", "P_{x2}") = \tan^{-1}(p_x - r_{12} d_5, p_y - r_{22} d_5) \end{aligned} \quad (\text{A-29})$$

the equation

$$s_1 "P_{x2}" + c_1 "P_{y2}" = K_2 \quad (\text{A-30})$$

becomes

$$s_1 c_{\varphi_2} + c_1 s_{\varphi_2} = \frac{K_2}{\rho_2} \quad (\text{A-31})$$

Using the trigonometric identity

$$s_a c_b + c_a s_b = s_{(a+b)} = \sin(a + b) \quad (\text{A-32})$$

the equation

$$s_1 c_{\varphi_2} + c_1 s_{\varphi_2} = \frac{K_2}{\rho_2} \quad (\text{A-33})$$

becomes

$$s_{(\theta_1+\varphi_2)} = \frac{K_2}{\rho_2} \quad (\text{A-34})$$

and, using the trigonometric identity

$$(s_a)^2 + (c_a)^2 = 1 \quad (\text{A-35})$$

the equation

$$(s_{(\theta_1+\varphi_2)})^2 + (c_{(\theta_1+\varphi_2)})^2 = 1 \quad (\text{A-36})$$

becomes

$$\left(\frac{K_2}{\rho_2}\right)^2 + (c_{(\theta_1+\varphi_2)})^2 = 1 \quad (\text{A-37})$$
$$c_{(\theta_1+\varphi_2)} = \pm \sqrt{1 - \frac{K_2^2}{\rho_2^2}}$$

Since

$$\tan(\theta_1 + \varphi_2) = \frac{s_{(\theta_1+\varphi_2)}}{c_{(\theta_1+\varphi_2)}} \quad (\text{A-38})$$

then

$$\begin{aligned}
\theta_1 + \varphi_2 &= \tan^{-1} \left(\frac{K_2}{\rho_2}, \pm \sqrt{1 - \frac{K_2^2}{\rho_2^2}} \right) \\
\theta_1 + \tan^{-1}(p_x - r_{12}d_5, p_y - r_{22}d_5) &= \tan^{-1} \left(\frac{K_2}{\rho_2}, \pm \sqrt{1 - \frac{K_2^2}{\rho_2^2}} \right) \quad (\text{A-39}) \\
\theta_1 &= \tan^{-1} \left(\frac{K_2}{\rho_2}, \pm \sqrt{1 - \frac{K_2^2}{\rho_2^2}} \right) - \tan^{-1}(p_x - r_{12}d_5, p_y - r_{22}d_5)
\end{aligned}$$

which is a closed-form solution for θ_1 , since $K_2, \rho_2, p_x, r_{12}, d_5, p_y, r_{22}, d_5$ are all known (see above).

Now that we have found these two angles, we can straightforwardly find the remaining angles:

$$\begin{aligned}
\begin{bmatrix} 0_T \\ 2_T \end{bmatrix}^{-1} \begin{bmatrix} 0_T \\ 5_T \end{bmatrix} &= \begin{bmatrix} 2_T \\ 5_T \end{bmatrix} \\
\begin{bmatrix} c_1c_2 & s_1c_2 & -s_2 & -c_2a_1 + s_2d_1 \\ -c_1s_2 & -s_1s_2 & -c_2 & s_2a_1 + c_2d_1 \\ -s_1 & c_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} &= \begin{bmatrix} c_3c_4 & -c_3c_4 & s_3 & -c_3d_5 + a_2 \\ s_4 & c_4 & 0 & c_4d_5 + d_3 \\ -s_3c_4 & s_3s_4 & c_3 & s_3s_4d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (\text{A-40})
\end{aligned}$$

$$\begin{aligned}
\begin{bmatrix} c_1c_2r_{11} + s_1c_2r_{21} - s_2r_{31} & c_1c_2r_{12} + s_1c_2r_{22} - s_2r_{32} & c_1c_2r_{13} + s_1c_2r_{23} - s_2r_{33} & c_1c_2p_x + s_1c_2p_y - s_2p_z - c_2a_1 + s_2d_1 \\ -c_1s_2r_{11} - s_1s_2r_{21} - c_2r_{31} & -c_1s_2r_{12} - s_1s_2r_{22} - c_2r_{32} & -c_1s_2r_{13} - s_1s_2r_{23} - c_2r_{33} & -c_1s_2p_x - s_1s_2p_y - c_2p_z + s_2a_1 + c_2d_1 \\ -s_1r_{11} + c_1r_{21} & -s_1r_{12} + c_1r_{22} & -s_1r_{13} + c_1r_{23} & -s_1p_x + c_1p_y \\ 0 & 0 & 0 & 1 \end{bmatrix} &= \quad (\text{A-41}) \\
\begin{bmatrix} c_3c_4 & -c_3c_4 & s_3 & -c_3d_5 + a_2 \\ s_4 & c_4 & 0 & c_4d_5 + d_3 \\ -s_3c_4 & s_3s_4 & c_3 & s_3s_4d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

Note that:

$$\begin{aligned}
s_3 &= c_1c_2r_{13} + s_1c_2r_{23} - s_2r_{33} \\
c_3 &= -s_1r_{13} + c_1r_{23}
\end{aligned} \quad (\text{A-42})$$

By definition:

$$\theta_3 = \tan^{-1}(s_3, c_3) \quad (\text{A-43})$$

which is a closed-form solution for θ_3 , since $\theta_1, \theta_2, r_{13}, r_{23}, r_{33}$ are all known (see above).

Note that:

$$\begin{aligned} s_4 &= -c_1 s_2 r_{11} - s_1 s_2 r_{21} - c_2 r_{31} \\ c_4 &= -c_1 s_2 r_{12} - s_1 s_2 r_{22} - c_2 r_{32} \end{aligned} \quad (\text{A-44})$$

By definition:

$$\theta_4 = \tan^{-1}(s_4, c_4) \quad (\text{A-45})$$

which is a closed-form solution for θ_4 , since $\theta_1, \theta_2, r_{11}, r_{21}, r_{31}, r_{12}, r_{22}, r_{32}$ are all known (see above).

Method 1: Numerical Evaluation with an Unfixed End Effector Rotation Matrix

When we have a desired end effector position with relaxed orientation constraints, we need to develop numerical methods for solving this problem. We cannot directly use the closed form solution since it depends on elements of the end effector rotation matrix. When solving numerically, we can implement constraints to try and force the computed solution to fit our particular needs. The current algorithm being implemented solves the equations

$$\begin{aligned} &(p_x - p_{x_i})^2 \\ &(p_y - p_{y_i})^2 \\ &(p_z - p_{z_i})^2 \end{aligned} \quad (\text{A-46})$$

by setting them equal to zero; where $p_x, p_y,$ and p_z are the desired position components and $p_{x_i}, p_{y_i},$ and p_{z_i} are described in terms of the unknown joint angles $\theta_1, \theta_2, \theta_3, \theta_4$ and the known D-H parameters. With three equations and four unknowns, to minimize

we set one of the angles to be constant. To keep angles in a specified range we can add additional equations that the algorithm minimizes. For example, equations that equate to large values for angles outside of our desired range and are zero inside the correct range would serve this purpose.

Method 2: Brute Force Search over Method 1

We call the Method 1 algorithm at incremental value steps (given) over a range of angles (given) for the “fixed” angle in a brute force search.

Method 3: Addition of Fourth Cost Metric

Use a cost function of some kind (e.g. least squares “distance” in per-angle change) to replace the fourth “equation”, rather than fixing one of the angles.

MM-Arm Dynamics & Singularity Identification

Note that since our arm is a 4-DOF manipulator system, our Jacobian is nonsquare. Using the formulation above and substituting in all measured D-H parameters, we find that the tooltip position with respect to the base frame, 0p_T , is given by:

$${}^0p_T = {}^0T_5 p_T$$

$${}^0p_T = \begin{bmatrix} 11.25((c_1c_2c_3 + s_1s_3)s_4 + c_1s_2c_4) - 0.345(c_1c_2 - c_1) + 16.5c_1s_2 \\ 11.25((s_1c_2c_3 - c_1s_3)s_4 + s_1s_2c_4) - 0.345(s_1c_2 - s_1) + 16.5s_1s_2 \\ 3.814 - 11.25(s_2c_3s_4 - c_2c_4) + 0.345s_2 + 16.5c_2 \end{bmatrix} \quad (\text{A-47})$$

Differentiating, we get:

$${}^0J_{trans} = \begin{bmatrix} [11.25(c_2c_3s_4 - s_2c_4) + 0.345(c_2 - 1) - 16.5s_2]s_1 + 11.25(s_3s_4)c_1 \\ [11.25(c_2c_3s_4 + s_2c_4) - 0.345(c_2 - 1) + 16.5s_2]c_1 + 11.25(s_3s_4)s_1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} [11.25(-s_2c_3s_4 + c_2c_4) + 0.345s_2 + 16.5c_2]c_1 & -11.25(c_1c_2s_3 - s_1c_3)s_4 \\ [11.25(-s_2c_3s_4 + c_2c_4) + 0.345s_2 + 16.5c_2]s_1 & -11.25(s_1c_2s_3 + c_1c_3)s_4 \\ (-11.25c_3s_4 + 0.345)c_2 + (-11.25c_4 - 16.5)s_2 & 11.25s_2s_3s_4 \end{bmatrix} \quad (\text{A-48})$$

$$\begin{bmatrix} 11.25(c_1c_2c_3 + s_1s_3)c_4 - (c_1s_2)s_4 \\ 11.25(s_1c_2c_3 - c_1s_3)c_4 - (s_1s_2)s_4 \\ -11.25(s_2c_3c_4 + c_2s_4) \end{bmatrix}$$

For ${}^0J_{rot}$ we find that

$${}^0J_{rot} = \begin{bmatrix} {}^0R(z) & {}^0R(z) & {}^0R(z) & {}^0R(z) & {}^0R(z) \\ 0 & -s_1 & -c_1s_2 & c_1c_2s_3 - s_1c_3 & c_1c_2s_3 - s_1c_3 \\ 0 & c_1 & -s_1s_2 & s_1c_2s_3 + c_1c_3 & s_1c_2s_3 + c_1c_3 \\ 1 & 0 & -c_2 & -s_2s_3 & -s_2s_3 \end{bmatrix} \quad (\text{A-49})$$

However, since frame 5 has the same orientation as frame 4, we remove column 5 to simplify:

$${}^0J_{rot} = \begin{bmatrix} 0 & -s_1 & -c_1s_2 & c_1c_2s_3 - s_1c_3 \\ 0 & c_1 & -s_1s_2 & s_1c_2s_3 + c_1c_3 \\ 1 & 0 & -c_2 & -s_2s_3 \end{bmatrix} \quad (\text{A-50})$$

Since the servos used for joint angle control have internal (control law) programming to force them to move at approximately constant speed during a command, the dynamics of the arm are simplified greatly... in some respects, the problem becomes more one of determining the dynamics from a command rather than commanding the linear and angular velocities and accelerations we want.

In some cases, however, we would like to try to pick the linear velocities we want for the end effector (even if we cannot command the joint angular velocities we desire). We also

still must try to avoid poses of the arm that may present problems in the magnitude of the forces brought to bear on and by the arm. For this we want to determine where the singularities of the arm may occur and concentrate more on the ${}^0J_{trans}$ than ${}^0J_{rot}$.

Looking at ${}^0J_{trans}$, we can see immediately that when $\theta_4=0$ or 180 degrees, the third column zeros out and the Jacobian loses rank. Taking into consideration the valid range of motion for the arm, whenever $\theta_4=0$ degrees, the arm is at a singularity – which makes sense, because not pitching the elbow joint leaves the shoulder and elbow roll joints z -axes aligned.

When we have a square matrix, the singularities can be found easily as roots of the determinant of the matrix. Since ${}^0J_{trans}$ is known to lose rank in the third column in instances when $\theta_4=0$ degrees, we are more interested in cases where this is not true. If we remove the third column from the Jacobian (thus making it square) and take the determinant, we find that singularities for this modified matrix exist when

$$\begin{aligned} & \theta_4 s_2 d_3 a_2 c_3 c_4 - d_5 c_2 c_3 d_3 + a_1 a_2 c_3 c_4 + a_1 s_4 d_5 c_4 + c_2 a_2^2 c_3 c_4 \\ & + c_4 c_2 a_2 s_4 d_3 - d_5 s_4 c_2 c_3^2 a_2 c_4 + d_5 c_2 c_3 d_3 c_4^2 + a_1 s_4 d_3 - a_1 d_5 c_3^2 s_4 c_4 \\ & - s_2 d_3^2 s_4 - d_5 s_4 c_4 s_2 d_3 - d_5 s_2 c_4^2 a_2 c_3 = 0 \end{aligned} \quad (A-51)$$

Note that solutions to this determinant are not necessarily singularities of the arm. When solutions to this equation cause loss of rank in the columns, they are singularities of the Jacobian; however, when they cause loss of rank row-wise, one must still check the associated row in the removed column 3 to determine if loss of rank in the Jacobian truly occurs.

So, a simple procedure for checking whether a particular arm state (in joint space) will cause a singularity is as follows:

- 1) Check θ_4 . If it is 0 degrees, it is a singular state. Stop. Else, step 2.
- 2) Substitute the state into the equation above. If it resolves =0, go to step 3, else step 4.

- 3) Substitute the state into ${}^0J_{trans}$ and check rank. If it loses rank, it is a singular state. Stop. Else, step 4.
- 4) Substitute the state into $-s_2s_3 = s_1c_2s_3 + c_1c_3 = c_1c_2s_3 - s_1c_3$ to check the rank of column 3. If the equalities resolve as true, it is a singular state. Stop. Else (assume) it is not a singular state.

Appendix B

Task Timelines for Test Sets

Below are tables detailing the number of interactions and conflicts and when they occurred in each test given.

Table B 1: Task timelines for human and robotic agents during tests in test set 1

Test #	Scenario type	Tasks	Task Timeline											
			(b1, b2, b3, ba = buttons 1, 2, 3, all; ch = chips, dr = drink ma(th) all other times; nc, pc, mc = no, physical, mental conflict; -- = no-op)											
1	Z	Human	--	--	--	--	--	--	--	--	--	--	--	
1	Z	Robot	b1	b2	b3	b1	b3	b2	b1					
1	Z	Conflict												
2	A	Human												
2	A	Robot	--											
2	A	Conflict												
3	B	Human	ch	dr	ch	dr	ch	dr	ch	dr				
3	B	Robot	--											
3	B	Conflict												
4	C	Human	b1	b2	b3	b2	b3	b1	b3	b1	b2			
4	C	Robot	--											
4	C	Conflict												
5	D	Human	ch	b3	b1	dr	b2	b1	ch	b2	b3	dr		
5	D	Robot	--											
5	D	Conflict												
6	F	Human			ch	dr	dr	dr	ch					
6	F	Robot	b2	b3	b3	b3	b2	b3	b2	b2	b2	b3	b3	
6	F	Conflict			nc	nc	nc	nc	nc					
7	G	Human	ch	ch	ch	ch	ch	ch	ch					
7	G	Robot	b3	b2	b3	b3	b2	b2	b2	b2	b3	b3	b3	
7	G	Conflict	nc	nc	pc	pc	nc	nc						
8	H	Human	ch	ch	dr		ch		ch		ch			
8	H	Robot	b3	b2	b1	b3	b2	b1	b1	b2	b3	b3	b3	
8	H	Conflict	nc	nc	nc		nc	mc	mc	nc		nc		
9	I	Human	ch	ch	ch	ch	ch	ch	ch					
9	I	Robot	b2	b1	b2	b1	b2	b1	b2	b1	b2	b1	b2	
9	I	Conflict	pc	mc	pc	mc	pc	mc	pc	mc	pc	mc	pc	
10	J	Human	ch	dr	ch	dr	ch	dr						
10	J	Robot	[b1,b2,b3 reactivate every 5 seconds]											
10	J	Conflict	pc	mc	nc	mc	pc	mc	nc	mc	pc	mc	nc	mc

Table B 2: Task timelines for human and robotic agents during tests in test set 2

Test #	Scenario type	Tasks	Task Timeline
			(b1, b2, b3, ba = buttons 1, 2, 3, all; ch = chips, dr = drink ma(th) all other times; nc, pc, mc = no, physical, mental conflict; -- = no-op)
1	A	Human	
1	A	Robot	--
1	A	Conflict	
2	E	Human	
2	E	Robot	b2 b3 b2 b3 b2 b3 b2 b3 b2 b3 b2 b3 b2 b3 b2 b3 b2
2	E	Conflict	
3	H	Human	
3	H	Robot	b1 b2 b3 b2 b3 b1 b3 b1 b2
3	H	Conflict	mc mc
4	H	Human	ch dr ch dr
4	H	Robot	b3 b1 b2 b1 b2 b3
4	H	Conflict	mc mc
5	D	Human	b2 b3 ch b3 dr b3 dr b2 dr b3 ch
5	D	Robot	--
5	D	Conflict	
6	D	Human	ch b2 ch b3 ch b2 ch b2 ch b3 ch
6	D	Robot	--
6	D	Conflict	
7	D	Human	ch b2 ch dr b3 b2 ch b1 b1 ch b2 b3 ch b3 b3
7	D	Robot	
7	D	Conflict	
8	D	Human	ch b1 ch b1 ch b1 ch b1 ch b1 ch
8	D	Robot	
8	D	Conflict	
9	D	Human	ch ba dr ba ch ba dr ba ch ba dr
9	D	Robot	--
9	D	Conflict	

Table B 3: Task timelines for human and robotic agents during tests in test set 3

Test #	Scenario type	Tasks	Task Timeline														
			(b1, b2, b3, ba = buttons 1, 2, 3, all; ch = chips, dr = drink ma(th) all other times; nc, pc, mc = no, physical, mental conflict; -- = no-op)														
1	A	Human															
1	A	Robot	--														
1	A	Conflict															
2	G	Human	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch
2	G	Robot	b3	b2	b3	b3	b2	b2	b2	b2	b3	b3	b3	b3	b3	b3	b3
2	G	Conflict	nc	nc	pc	pc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc	nc
3	I	Human	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch	ch
3	I	Robot	b2	b1	b2	b1	b2	b1	b2	b1	b2	b1	b2	b1	b2	b1	b2
3	I	Conflict	pc	mc	pc	mc	pc	mc	pc	mc	pc	mc	pc	mc	pc	mc	pc
4	J	Human	ch	dr	ch	dr	ch	dr	ch	dr	ch	dr	ch	dr	ch	dr	ch
4	J	Robot	[b1,b2,b3 reactivate every 5 seconds]														
4	J	Conflict	pc	mc	nc	mc	pc	mc	nc	mc	pc	mc	nc	mc	pc	mc	nc
5	G / I	Human															
5	G / I	Robot	b2	b3	b2	b3	b2	b3	b2	b3	b2	b3	b2	b3	b2	b3	b2
5	G / I	Conflict															
6	C / D	Human	b2	b3	b2	b2	b3										
6	C / D	Robot	--														
6	C / D	Conflict															
7	C / D	Human	b1	b1	b1	b1	b1										
7	C / D	Robot	--														
7	C / D	Conflict															
8	H / I	Human															
8	H / I	Robot	b3	b1	b2	b1	b2	b3									
8	H / I	Conflict		mc	mc												
9	C / D	Human	b3	b1	b2	b1	b2	b3									
9	C / D	Robot	--														
9	C / D	Conflict															

Note that for test set 3, in tests 5 through 9 no consumption messages were displayed but the human was told to eat or drink whenever they wished (thus, conflicts varied per person). These tests were not included in our data processing for this paper due to the high variability between subjects of when conflict cases could, and did, occur. It was collected towards later determination of variability and frequency of consumption task occurrence per subject in future work.

Bibliography

- [1] Craig, J. J., *Introduction to Robotics: Mechanics and Control*, 3rd ed., Pearson Education, Inc., Upper Saddle River, NJ, 2005, pp. 28, 35-36, 68-69.
- [2] Oman, C., "Spatial Orientation and Navigation in Microgravity," in *Spatial Processing in Navigation, Imagery and Perception*, Mast, F. & Jäncke, L. eds., Springer US, 2007, pp. 209-247.
- [3] NASA, *Flight Data Files (STS-135), EVA Checklist*, URL: http://www.nasa.gov/centers/johnson/pdf/567070main_EVA_135_FIN_1.pdf [cited 16 January 2014].
- [4] Jha, A., "Meet Robonaut 2, astronaut assistant | Science | The Guardian," URL: <http://www.guardian.co.uk/science/2010/nov/02/robonaut-2-international-space-station> [cited 31 March 2011].
- [5] Stoll, E., Jaekel, S., Katz, J., Saenz-Otero, A., and Varatharajoo, R., "SPHERES Interact, Human-Machine Interaction aboard the International Space Station," *Journal of Field Robotics*, Vol. 29, No. 4, 2012, pp. 554-575.
- [6] Pedersen, L., Kortenkamp, D., Wettergreen, D., and Nourbakhsh, I., "A survey of space robotics," *Proceedings of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2003, pp. 19-23.
- [7] Ahmed, M. R., *Compliance Control of Robot Manipulator for Safe Physical Human Robot Interaction*, university, Ö. ed., Doctoral thesis, Örebro, Sweden, 2011.
- [8] Roderick, S., Roberts, B., Atkins, E., and Akin, D., "The Ranger robotic satellite servicer and its autonomous software-based safety system," *IEEE Intelligent Systems*, Vol. 19, No. 5, September-October 2004, pp. 12-19.
- [9] Haidegger, T., "Advanced Robotic Arms In Space," *55th International Astronautical Congress*, Vancouver, Canada, 2004, pp. 1-10.
- [10] Liu, Y., and Najat, G., "Robotic Urban Search and Rescue: A Survey from the Control Perspective," *Journal of Intelligent & Robotic Systems*, March 2013, pp. 1-19.
- [11] Lungarella, M., Metta, G., Pfeifer, R., and Sandini, G., "Developmental robotics: a survey," *Connection Science*, Vol. 15, No. 4, 2003, pp. 151-190.
- [12] Gerkey, B. P., and Matarić, M. J., "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *The International Journal of Robotics Research*, Vol. 23, No. 9, September 2004, pp. 939-954.
- [13] Van Der Krogt, R., and De Weerd, M., "Plan Repair as an Extension of Planning," *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, 2005, pp. 161-170.
- [14] Kulic, D., and Croft, E., "Pre-collision safety strategies for human-robot interaction," *Autonomous Robots*, Vol. 22, No. 2, 2007, pp. 149-164.
- [15] Karami, A.-B., Jeanpierre, L., and Mouaddib, A.-I., "Human-robot collaboration for a shared mission," *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction (HRI'10)*, 2010, pp. 155-156.
- [16] De Santis, A., Siciliano, B., De Luca, A., and Bicchi, A., "An atlas of physical human-robot interaction," *Mechanism and Machine Theory*, Vol. 43, No. 3, March 2008, pp. 253-270.
- [17] Wikipedia contributors, "Safety," URL:

<http://en.wikipedia.org/w/index.php?title=Safety&oldid=507198516> [cited 17 August 2012].

- [18] Haddadin, S., Albu-Schaffer, A., Frommberger, M., Rossmann, J., and Hirzinger, G., "The "DLR Crash Report": Towards a standard crash-testing protocol for robot safety - Part I: Results," *ICRA '09, IEEE International Conference on Robotics and Automation*, Kobe, 2009, pp. 272-279.
- [19] Haddadin, S., Albu-Schaffer, A., Frommberger, M., Rossmann, J., and Hirzinger, G., "The "DLR crash report": Towards a standard crash-testing protocol for robot safety - Part II: Discussions," *ICRA '09, IEEE International Conference on Robotics and Automation*, Kobe, 2009, pp. 280-287.
- [20] Albu-Schäffer, A., Ott, C., and Hirzinger, G., "A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots," *The International Journal of Robotics Research*, Vol. 26, No. 1, January 2007, pp. 23-39.
- [21] Kulic, D., and Croft, E. A., "Real-time safety for human-robot interaction," *Robotics and Autonomous Systems*, Vol. 54, No. 1, 2006, pp. 1-12.
- [22] Spong, M. W., and Vidyasagar, M., *Robot Dynamics and Control*, John Wiley & Sons, New York, 1989.
- [23] Betts, and T., J., "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193-207.
- [24] Pfeiffer, F., and Johanni, R., "A concept for manipulator trajectory planning," *IEEE Journal of Robotics and Automation*, Vol. 3, No. 2, April 1987, pp. 115-123.
- [25] Hwang, Y. K., and Ahuja, N., "Gross motion planning—a survey," *ACM Computing Surveys (CSUR)*, Vol. 24, No. 3, 1992, pp. 219-291.
- [26] Lavalle, S. M., and Kuffner Jr., J. J., "Rapidly-Exploring Random Trees: Progress and Prospects," *Algorithmic and Computational Robotics: New Directions: Fourth Workshop on the Algorithmic Foundations of Robotics*, Dartmouth College, 2000, p. 293.
- [27] Macfarlane, S., and Croft, E. A., "Jerk-bounded manipulator trajectory planning: design for real-time applications," *IEEE Transactions on Robotics and Automation*, February 2003, pp. 42-52.
- [28] Yang, S. X., and Meng, M., "Neural network approaches to dynamic collision-free trajectory generation," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 31, No. 3, June 2001, pp. 302-318.
- [29] Vadakkepat, P., Tan, K. C., and Ming-Liang, W., "Evolutionary artificial potential fields and their application in real time robot path planning," *Proceedings of the 2000 Congress on Evolutionary Computation*, Vol. 1, La Jolla, CA, 2000, pp. 256-263.
- [30] Kulic, D., *Safety for Human-Robot Interaction*, Columbia, U. o. B. ed., Doctoral thesis, British Columbia, Canada, 2005.
- [31] Zweben, M., Davis, E., Daun, B., and Deale, M. J., "Scheduling and rescheduling with iterative repair," *IEEE Transactions on Man and Cybernetics Systems*, Vol. 23, No. 6, 1993, pp. 1588-1596.
- [32] Chien, S., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G., "Using iterative repair to improve the responsiveness of planning and scheduling," *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling*, Brackenridge, CO, 2000.
- [33] Rabideau, G., Knight, R., Chien, S., Fukunaga, A., and Govindjee, A., "Iterative repair planning for spacecraft operations using the aspen system," *Artificial Intelligence, Robotics and Automation in Space*, Vol. 440, August 1999, pp. 99-106.
- [34] Alpaydin, E., *Introduction to Machine Learning*, 2nd ed., MIT Press, Boston, MA, 2010.
- [35] Russell, S. J., and Norvig, P., *Artificial intelligence: A modern approach*, 2nd ed., Prentice Hall/Pearson Education, Upper Saddle River, N.J, 2003.
- [36] Malik, G., Nau, D. S., and Traverso, P., *Automated planning: theory and practice*, Elsevier, 2004.
- [37] Musliner, D. J., Durfee, E. H., and Shin, K. G., "World Modeling for the Dynamic Construction of Real-Time Control Plans," *AI Journal*, Vol. 74, No. 1, March 1995, pp. 83-127.
- [38] Boutilier, C., Dean, T., and Hanks, S., "Decision-Theoretic Planning: Structural Assumptions and Computational Leverage," *Journal of Artificial Intelligence Research*, Vol. 11, No. 1, 1999, pp. 1-94.
- [39] Puterman, M. L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed.,

John Wiley & Sons, New York, USA, 1994.

- [40] Cassandra, A. R., Kaelbling, L. P., and Littman, M. L., "Acting optimally in partially observable stochastic domains," *Proceedings of the National Conference on Artificial Intelligence*, 1995, pp. 1023-1023?
- [41] Williams, T., and Tanygin, S., "On-orbit engineering tests of the AERCam Sprint robotic camera vehicle," *Proceedings of the AAS/AIAA Space Flight Mechanics Meeting*, 1998, pp. 1001-1020.
- [42] Wagenknecht, J., Fredrickson, S., Manning, T., and Jones, B., "Design, Development and Testing of the Miniature Autonomous Extravehicular Robotic Camera (Mini AERCam) Guidance, Navigation, and Control System," *26th Annual American Astronautical Society Guidance and Control Conference*, February 5-9, 2003.
- [43] Dorais, G. A., and Gawdiak, Y., "The personal satellite assistant: an internal spacecraft autonomous mobile monitor," *Proceedings of IEEE Aerospace Conference*, Vol. 1, 2003.
- [44] Hexmoor, H., and Vaughn, J., "Computational adjustable autonomy for NASA Personal Satellite Assistants," *Proceedings of the 2002 ACM Symposium on Applied Computing*, 2002, pp. 21-26.
- [45] Mohan, S., Saenz-Otero, A., Nolet, S., Miller, D., and Sell, S., "SPHERES Flight Operations Testing and Execution," *Proceedings of the 58th International Astronautical Congress*, Hyderabad, India, 2007.
- [46] Katz, J., Saenz-Otero, A., and Miller, D., "Development and Demonstration of an Autonomous Collision Avoidance Algorithm aboard the ISS," *Proceedings of the 2011 IEEE Aerospace Conference*, Big Sky, MT, 2011, pp. 1-6.
- [47] Biesiadecki, J. J., Leger, P. C., and Maimone, M. W., "Tradeoffs Between Directed and Autonomous Driving on the Mars Exploration Rovers," *The International Journal of Robotics Research*, Vol. 26, January 2007, pp. 91-104.
- [48] Coleshilla, E. *et al.*, "Dextre: Improving maintenance operations on the International Space Station," *Acta Astronautica*, Vol. 64, No. 9-10, May-June 2009, pp. 869-874.
- [49] NASA, "NASA - Dextre's Final Exam Scheduled for December 22-23, 2010," URL: http://www.nasa.gov/mission_pages/station/structure/dextre_final_exam.html [cited 31 March 2011].
- [50] NASA, "NASA - Dextre Successfully Completes Its First Official Job," URL: http://www.nasa.gov/mission_pages/station/expeditions/expedition26/dextre_firstjob.html [cited 31 March 2011].
- [51] Harding, P., "Dextre and RRM complete record breaking week of robotics on ISS," URL: <http://www.nasaspaceflight.com/2012/03/dextre-rrm-complete-record-breaking-week-robotics-iss/> [cited 26 June 2012].
- [52] Harding, P., URL: <http://www.nasaspaceflight.com/2012/06/iss-dextre-rrm-complete-second-round-joint-ops-cdra-recovered/> [cited 26 June 2012].
- [53] Diftler, M. A., Culbert, C. J., Ambrose, R. O., Platt, R. . J., and Bluethmann, W. J., "Evolution of the NASA/DARPA Robonaut control system," *ICRA '03, Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2, 2003, pp. 2543 - 2548.
- [54] Diftler, M. A. *et al.*, "Robonaut 2 – The First Humanoid Robot in Space," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, CN, May 2011, pp. 2178-2183.
- [55] Pappas, S., "Humanoid Robot Hitching Space Ride on Shuttle Discovery | Space.com," URL: <http://www.space.com/9384-humanoid-robot-hitching-space-ride-shuttle-discovery.html> [cited 31 March 2011].
- [56] Choi, C. Q., "New Robot Could Aid Astronauts in Space | Space.com," URL: <http://www.space.com/7871-robot-aid-astronauts-space.html> [cited 31 March 2011].
- [57] Sheridan, T. B., *Telerobotics, Automation, and Human Supervisory Control*, MIT Press, Cambridge, MA, 1992.
- [58] Hung, J. Y., Gao, W., and Hung, J. C., "Variable structure control: a survey," *IEEE Transactions on Industrial Electronics*, Vol. 40, No. 1, 1993, pp. 2-22.
- [59] Ikeura, R., Inooka, H., and Mizutani, K., "Subjective evaluation for maneuverability of a robot

- cooperating with human," *Proceedings of the 1999 IEEE International Workshop on Robot and Human Interaction*, 1999, pp. 201-205.
- [60] Kato, R., and Arai, T., "Assessment of Mental Stress on Human Operators Induced by the Assembly Support in a Robot-Assisted "Cellular Manufacturing" Assembly System," *International Journal of Automation Technology*, Vol. 3, No. 5, 2009, pp. 569-579.
- [61] Steinfeld, A. *et al.*, "Common metrics for human-robot interaction," *HRI '06, Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, Salt Lake City, 2006, pp. 33-40.
- [62] Sisbot, E. , Clodic, A., Alami, R., and Ransan, M., "Supervision and motion planning for a mobile manipulator interacting with humans," *HRI '08 Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, Amsterdam, The Netherlands, 2008, pp. 327-334.
- [63] "American National Standard for Industrial Robots and Robot Systems – Safety Requirements," Robotic Industries Association, American National Standards Institute, ANSI/RIA R15.06-1999, 1999.
- [64] "Robots for Industrial Environment - Safety Requirements - Part 1 - Robot," American National Standards Institute, International Standard Organization, ANSI/RIA/ISO 10218-1:2007, 2007.
- [65] Schuster, G., and Winrich, M., "Robotics Safety," Rockwell Automation, White Paper SAFETY-WP009A-EN-P, December 2009.
- [66] Carpin, S., and Parker, L. E., "Cooperative leader following in a distributed multi-robot system," *ICRA '02, Proceedings of IEEE International Conference on Robotics and Automation*, Washington, D.C., 2002, pp. 2994 - 3001.
- [67] Greenstein, J. S., and Revesman, M. E., "Two Simulation Studies Investigating Means of Human-Computer Communication for Dynamic Task Allocation," *IEEE Transactions on Man and Cybernetics Systems*, Vol. 16, No. 5, September 1986, pp. 726-730.
- [68] Breazeal, C., Kidd, C. D., Thomaz, A. L., Hoffman, G., and Berlin, M., "Effects of nonverbal communication on efficiency and robustness in human-robot teamwork," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, 2005, pp. 708-713.
- [69] Kaupp, T., Makarenko, A., and Durrant-Whyte, H., "Human-robot communication for collaborative decision making — A probabilistic approach," *Robotics and Autonomous Systems*, Vol. 58, No. 5, May 2010, pp. 444-456.
- [70] Yang, J., Xu, Y., and Chen, C. S., "Human action learning via hidden Markov model," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 27, No. 1, 1997, pp. 34-44.
- [71] Aggarwal, J. K., and Cai, Q., "Human motion analysis: A review," *Computer Vision and Image Understanding*, Vol. 73, No. 3, March 1999, pp. 428-440.
- [72] Yamato, J., Ohya, J., and Ishii, K., "Recognizing human action in time-sequential images using hidden Markov model," *Proceedings of the CVPR'92, IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1992, pp. 379-385.
- [73] Bregler, C., "Learning and recognizing human dynamics in video sequences," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 568-574.
- [74] Brand, M., Oliver, N., and Pentland, A., "Coupled hidden Markov models for complex action recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 994-999.
- [75] Lee, C., and Xu, Y., "Online, interactive learning of gestures for human/robot interfaces," *Proceedings of IEEE International Conference on Robotics and Automation*, 1996, pp. 2982-2987.
- [76] Bobick, A. F., "Movement, activity and action: The role of knowledge in the perception of motion," *Philosophical Transactions of the Royal Society B: Biological Sciences*, Vol. 352, No. 1358, August 29 1997, pp. 1257-1265.
- [77] Gavrilu, D. M., "The Visual Analysis of Human Movement: A Survey," *Computer Vision and Image Understanding, Volume 73, No. 1*, January 1999, pp. 82-98.
- [78] Poppe, R., "Vision-based human motion analysis: an overview," *Computer Vision and Image Understanding (CVIU)*, Vol. 108, No. 1-2, 2007, pp. 4-18.

- [79] Poppe, R., "A survey on vision-based human action recognition," *Image and Vision Computing*, Vol. 28, No. 6, 2010, pp. 976-990.
- [80] Takeda, T., Hirata, Y., and Kosuge, K., "Dance Step Estimation Method Based on HMM for Dance Partner Robot," *IEEE Transaction on Industrial Electronics*, Vol. 54, No. 2, 2007, pp. 699-706.
- [81] Anh, M., Ho, T., Yamada, Y., and Umetani, Y., "An Adaptive Visual Attentive Tracker for Human Communicational Behaviors Using HMM-Based TD Learning With New State Distinction Capability," *IEEE Transactions on Robotics*, Vol. 21, No. 3, 2005, pp. 497-504.
- [82] Vondrak, M., Sigal, L., and Jenkins, O. C., "Physical simulation for probabilistic motion tracking," *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1-8.
- [83] Karami, A.-B., Jeanpierre, L., and Mouaddib, A.-I., "Partially Observable Markov Decision Process for Managing Robot," *21st International Conference on Tools with Artificial Intelligence (ICTAI '09)*, 2009, pp. 518-521.
- [84] Matignon, L., Karami, A. B., and Mouaddib, A. I., "A Model for Verbal and Non-Verbal Human-Robot Collaboration," *2010 AAAI Fall Symposium Series*, 2010.
- [85] Schmidt-Rohr, S. R., Knoop, S., Losch, M., and Dillmann, R., "Bridging the gap of abstraction for probabilistic decision making," *Robotics: Science and Systems, Zurich*, 2008.
- [86] Nikolaidis, S., and Shah, J., "Human-Robot Interactive Planning using Cross-Training: A Human Team Training Approach," *Proc. of Infotech@Aerospace*, Garden Grove, CA, June 2012.
- [87] Keizer, S., Foster, M. E., Lemon, O., Gaschler, A., and Giuliani, M., "Training and evaluation of an MDP model for social multi-user human-robot interaction," *Proceedings of the 14th Annual SIGdial Meeting on Discourse and Dialogue*, Metz, France, August 2013.
- [88] Fasola, J., and Matarić, M. J., "A Socially Assistive Robot Exercise Coach for the Elderly," *Journal of Human-Robot Interaction*, Vol. 2, No. 2, June 2013, pp. 3-32.
- [89] Box, G., and Hunter, W., *Statistics for Experimenters*, Wiley-Interscience, New York, 2005.
- [90] Montgomery, D., *Design and Analysis of Experiments*, John Wiley & Sons, Chichester, 2009.
- [91] "NASA TLX Paper and Pencil Version Instruction Manual," *NASA TLX Homepage [online archive]*, URL: <http://humansystems.arc.nasa.gov/groups/TLX/> [cited 2 January 2011].
- [92] McGhan, C. L. R., and Atkins, E. M., "Physically-Proximal Human-Robot Collaboration: Enhancing Safety and Efficiency Through Intent Prediction," *Proc. Infotech@Aerospace Conference*, Seattle, WA, Apr. 2009.
- [93] McGhan, C. L. R., and Atkins, E. M., "A Low-Cost Manipulator for Space Research and Undergraduate Engineering Education," *Proc. Infotech@Aerospace Conference*, Atlanta, GA, Apr. 2010.
- [94] Nickels, K., "Hand-Eye Calibration of Robonaut," *San Antonio Chapter IEEE Computer Society Past Meeting Presentations September 16, 2004 [online archive]*, URL: http://www.ieee-cs-cts.org/past_meetings.htm [cited 2 January 2011].
- [95] Rekimoto, J., "SmartSkin: an infrastructure for freehand manipulation on interactive surfaces," *Proceedings of the SIGCHI conference on Human factors in computing systems*, Minneapolis, MN, 2002, pp. 113-120.
- [96] Yarrow, K., Brown, P., and Krakauer, J. W., "Inside the brain of an elite athlete: the neural processes that support high achievement in sports," *Nature Reviews Neuroscience*, Vol. 10, No. 8, July 2009, pp. 585-596.
- [97] Bonasso, R. P. *et al.*, "Experiences with an architecture for intelligent, reactive agents," *Journal of Experimental & Theoretical Artificial Intelligence*, 9, Vol. 2, No. 3, 1997, pp. 237-256.
- [98] Gat, E., "On Three-Layer Architectures," in *Artificial Intelligence and Mobile Robots*, David Kortenkamp, R. P. B. a. R. M. ed., AAAI Press, 1997, pp. 195-210.
- [99] Montemerlo, M., Roy, N., and Thrun, S., "Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit," *Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003. (IROS 2003)*, Vol. 3, 2003, pp. 2436-2441.

- [100] Casner, S. A., "Understanding the determinants of problem-solving behavior in a complex environment," *Human Factors*, Vol. 36, No. 4, December 1994, pp. 580-596.
- [101] Dellnitz, M., and Junge, O., "An adaptive subdivision technique for the approximation of attractors and invariant measures," *Computing and Visualization in Science*, Vol. 1, No. 2, 1997, pp. 63-68.
- [102] Fonseca, C. M., and Fleming, P. J., "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evolutionary Computation*, Vol. 3, No. 1, Spring 1995, pp. 1-16.
- [103] Teich, J., "Pareto-Front Exploration with Uncertain Objectives," in *Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science, 2001*, Zitzler, E., Thiele, L., Deb, K., Coello Coello, C. & Corne, D. eds., Springer Berlin / Heidelberg, 2001, pp. 314-328.
- [104] McGhan, C. L. R., Nasir, A., and Atkins, E. M., "Human Intent Prediction Using Markov Decision Processes," *Proc. Infotech@Aerospace Conference*, Garden Grove, CA, June 2012.
- [105] McGhan, C. L. R., and Atkins, E. M., "Towards Guaranteeing Safe and Efficient Human-Robot Collaboration Using Human Intent Prediction," *Proc. of AIAA Space 2012 Conference and Exposition*, Pasadena, CA, Sept. 2012.
- [106] McGhan, C. L. R., and Atkins, E. M., "Human Productivity in a Workspace Shared with a Safe Robotic Manipulator," *Journal of Aerospace Information Systems*, accepted June 27, 2012 (to be published).
- [107] McGhan, C. L. R., Atkins, E. M., and Nasir, A., "Human Intent Prediction Using Markov Decision Processes," *Journal of Aerospace Information Systems*, accepted under major revision 7 June 2013 (under review 10 Sept. 2013).