MICHIGAN ENGINEERING

UNIVERSITY of MICHIGAN ■ COLLEGE of ENGINEERING

Object Detection for Hardware-Accelerated Task Planning
Honors Capstone Project

By: Anastasia Warner, Computer Science Engineering

Advisor: Odest Chadwicke Jenkins, Professor, Michigan Robotics Institute

Collaborators: Anthony Opipari, PhD Candidate, Computer Science and Engineering
Alphonsus Adu-Bredu, PhD Candidate, Michigan Robotics Institute

Date: April 25, 2022

Departments: Honors Engineering, Michigan Robotics Institute

OVERVIEW

Belief-space planning and probabilistic inference are two techniques used to inform the actions taken by autonomous robotic systems. These methods allow for the design of fully autonomous systems which are sensitive to uncertainty about the current state of the system, and much promising research has been done to design algorithms that employ these techniques. However, current methods require extensive computational resources and time, becoming prohibitive for robot systems that need to act in real time.

Our project was to design a perception-planning system for autonomous robots that requires less time and power by using parallel computing on FPGAs to speed up belief-space planning algorithms. In particular, my capstone work was to design the perception system for a real-world experiment necessary to confirm the project's effectiveness. I implemented the components necessary to detect and localize colored blocks used in the experiment and interface with the parallelized planning code, which uses the blocks' locations to plan a robot's movements towards a goal configuration by sweeping them into place.

This report will discuss the motivation for the team's belief-space planning project and will focus on the work done to develop and test the perception system for the final sweeping experiment. Work on the project is still in progress, so we hope to complete the experiment and get results by the end of the summer.

INTRODUCTION

Much work in accelerated belief-space planning is based on Ye et. al's DESPOT [1]. This algorithm adapted the *partially observable Markov decision process* (POMDP), commonly used for robotic planning under uncertainty, into a more computationally tractable problem by creating a "sparse approximation of the standard belief tree." The nodes of the belief tree contain randomly sampled "particles" or "scenarios" representing possible states of the world. Each scenario is "played out" and propagated to determine the most optimal policy, or set of actions, that can be taken to maximize the reward of the execution. Figure 1 illustrates an outline of how the tree represents the beliefs about the possible state of the world.

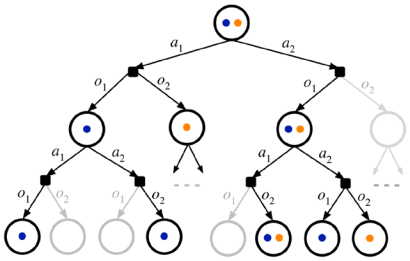

Figure 1: As an action a_1 is considered, the possible scenarios are updated based on possible next observations o_1 and o_2. A scenario is represented by a particle, and the belief about the world is represented by the collection of all the particles.

In HyP-DESPOT [2], Cai et. al made the DESPOT algorithm even more efficient by using parallelization on both the CPU and a GPU to accelerate its planning. HyP-DESPOT uses multi-core CPUs to traverse the belief tree simultaneously and GPUs to expand leaf nodes in parallel, propagating the beliefs forward in time. This advancement brought belief-space planning to near real-time capabilities.

However, GPUs are still not as energy efficient as possible. In order for a system with high dimensionality of state space, action space, and observation space to plan in real time and with low power, it may require even more accelerated approaches. To accomplish this, our project team is developing an algorithm inspired by DESPOT and HyP-DESPOT that uses FPGAs (field-programmable gate arrays) to implement some of the parallel processing needed for the belief tree search. FPGAs also generally have lower power consumption than GPUs, which will allow for less expensive robotic planning systems overall.

The goal of my capstone project was to design the perception system for the real-world sweeping experiment we will use to prove the accuracy and efficiency of the new algorithm. I implemented the components necessary to detect the blocks used in the experiment, calculate their locations, and interface with the parallelized planning code, which uses the blocks' locations to plan the robot's movements.

The rest of this report will discuss the process of designing the perception system used for the sweeping experiment and its final design and preliminary results. The FPGA-accelerated code is not yet complete, but it is anticipated to be finished by the end of the summer. Once the algorithm is ready, it will be tested on virtual toy experiments and this sweeping system.

SWEEPING EXPERIMENT

The team selected a sweeping experiment to test the belief-space planning baseline algorithms (DESPOT and HyP-DESPOT) and the FPGA-accelerated algorithm. In this experiment, the Digit robot from Agility Robotics will use a pushing or sweeping object such as cardboard to push blocks on a table into a goal configuration. This experiment provides a strong test case because it involves uncertainty about the observations of the state of the world (i.e. the blocks' locations and orientations), which will allow us to demonstrate the probabilistic uncertainty handling capabilities of the belief-space planning algorithm.

Each possible set of configurations for the blocks are represented as scenarios in the DESPOT-inspired algorithm (Figure 2).

Digit's manipulation system is made possible by work done by collaborator Alphonsus Adu-Bredu in [3].

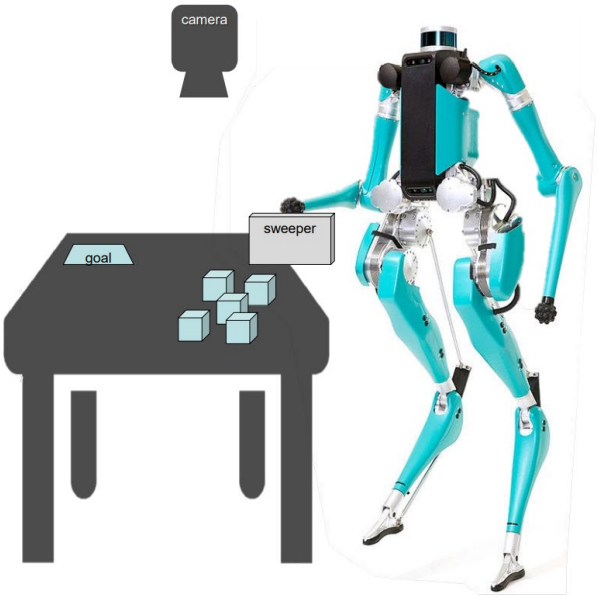

Figure 2: In the sweeping experiment, the Digit robot will manipulate a sweeper tool to push colored blocks on a table into a goal configuration. The optimal experimental setup was determined in an iterative testing process.

Notably, using blocks in the sweeping experiment to verify the results introduces an interesting constraint. The blocks are easier to detect, push, and manage than something like rice. However, we still want to maintain a natural source of uncertainty in the system, so this guided certain decisions within the design process which I will discuss.

DESIGN PROCESS

Constraints and overview
The most common way to facilitate object detection is to use white borders on a black object such as a box or QR code, which makes the object easy to detect using any type of edge

detection. However, as mentioned above, we want to have a natural source of uncertainty about the locations of the blocks used in the experiment. Informally, we also want the experiment to present a challenging problem for the belief-space planning algorithm to solve. For example, using single-colored blocks if possible would be more convincing than using multi-colored blocks which are easier to detect and segment.

In order to achieve this, I determined requirements for candidate detection methods and block color options, explored options for the above, and recorded other experimental setup requirements. Figure 3 illustrates a sample image that would be obtained in the experimental setup under our initial assumptions, that we would use the Digit's built-in camera mounted on the robot and use single-colored red or green blocks.

Figure 3: Sample cropped image capture from RGB camera at an angle, highlighting the difficult-to-detect edges of blocks that are swept into a pile.

Block detection

To begin my detection search, I explored both machine learning options and raw image-processing approaches in parallel before I could determine the most promising approach for our purposes. Several shape and cuboid detectors exist such as [4], [5], and [6], but none were suitable for the sweeper experiment setup or the time frame of the project. [4] and [6] would need to be trained on an image dataset, which I ruled out because of some project-specific constraints: we did not yet know the exact final experimental setup and didn't have access to the Digit robot to test the setup, and I didn't want to train the models on a dataset that was not necessarily transferable to the final experiment. For [5], the project had complex dependencies and I determined it would be too computationally expensive, since our priority was to measure computational time and energy of the planning algorithm, not the perception system.

After ruling out deep learning methods, I focused on testing options using OpenCV image processing methods. Using OpenCV, which is only implemented in Python and C++, introduced the need for an interface system between the block detection code and the planning code, which was being written in Julia.

Processing Methods

I started testing block detection with the most standard method: contour detection on a grayscale image (using single-colored blocks). Since the pixel colors were very similar, contour detection failed to segment the blocks (Figure 4). Next, I tried thresholding on pure HSV and contouring based on the binary thresholding output, with similar results. I also tried creating a grayscale image based on pixel difference from the pure block color. The results, also poor, can be seen below.

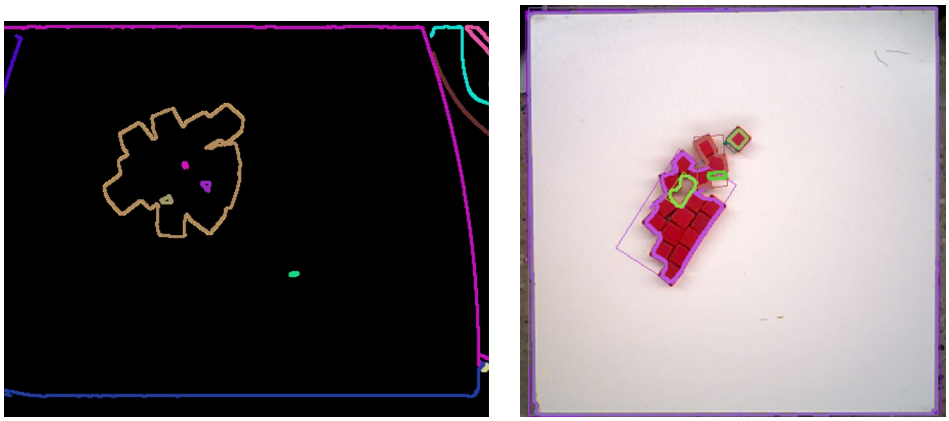

Figure 4: Grayscale contour detection (left) and color difference contour detection (right) fail to segment blocks from each other.

After these attempts, I tried testing with multiple colors of blocks. In these tests, I tuned thresholds for each color block present, specifying high and low pixel values for both RGB images and HSV images. RGB is the most common image representation format, but HSV is commonly used in image detection since the Hue value can be less variable while Saturation and Value may change due to shadows or light. For each color, I apply thresholding on just that color range in the image and get the contours of that color alone. Then I filter the contours by length to remove ones that are too small to be blocks, and consolidate the remaining contours for each color. This proved to be the best method I tested. The process and result of this method will be discussed in more detail in the results section of this report.

Camera Testing

I explored using three different cameras and selected the best one for the experiment. The first I tested was a high-resolution (1920x1080 pixel) webcam, NexiGo N60, on all-red blocks. This had good resolution but lacked depth detection capabilities which made it easier to segment the blocks. Next I tested the Asus Xtion Pro Live which had resolution too poor to detect the separation between single-colored blocks. Finally I tried the Intel RealSense D345, which had resolution up to 1280x720 but included stereo depth sensing. This was still not a high enough resolution to segment all the blocks, which was what inspired me to explore using multicolored blocks for the experiment.

In the course of the testing, I determined that it would be necessary to mount the camera directly above the table because depth filtering was complicated by having any significant angle of view of the blocks. Specifically, since the blocks are small, the height of the blocks on the far edge of the table in view would be further from the camera than the close end of the table. Mounting the

camera downwards allows the use of a simple depth filter and avoids using a more computationally expensive point-cloud ground layer extraction.

Verification and Data Collection
To determine which system would best meet the needs of the experiment, I prepared three test datasets to allow comparison of the blocks' colors and the color detection method used. I used the Intel RealSense D435 camera mounted above a table and collected datasets using three color choices: all red blocks, primary-colored blocks (red, blue, and yellow) for some contrast, and multi-colored blocks (red, orange, yellow, green, blue) for more contrast. For each dataset, I recorded two 30-second videos while pushing the blocks around on the table using a cardboard sweeper: one video had occlusion, where I held the sweeper straight up and allowed my arm to impede the camera view of the blocks like the robot arm would, and one video had no occlusion, where I pushed from the side to avoid obscuring the blocks. I also recorded 15-20 images of each, in various configurations of blocks separated and pushed together, and saved the depth and RGB data for all of the tests.

I evaluated the accuracy of each dataset so that the team could determine which to use, weighing accuracy with our informal need for uncertainty in the system.

RESULTS AND FINAL DESIGN

In the final perception system design, the code captures an image from a Robot Operating System (ROS) camera stream that interfaces with the robot, and uses OpenCV to dilate and blur the image by a few pixels, which reduces noise. Then any region outside the table is masked out, and color contours are extracted. To get the locations, the pixel values for the location and orientation of each contour's bounding box are collected. The depth data for that pixel is used to get the real-world position relative to the center of the camera view.

The location information of each block is stored and passed to the Julia planning code via an interface that makes use of PyCall, a Julia-Python interfacing package, to convert the Python vector of output to Julia.

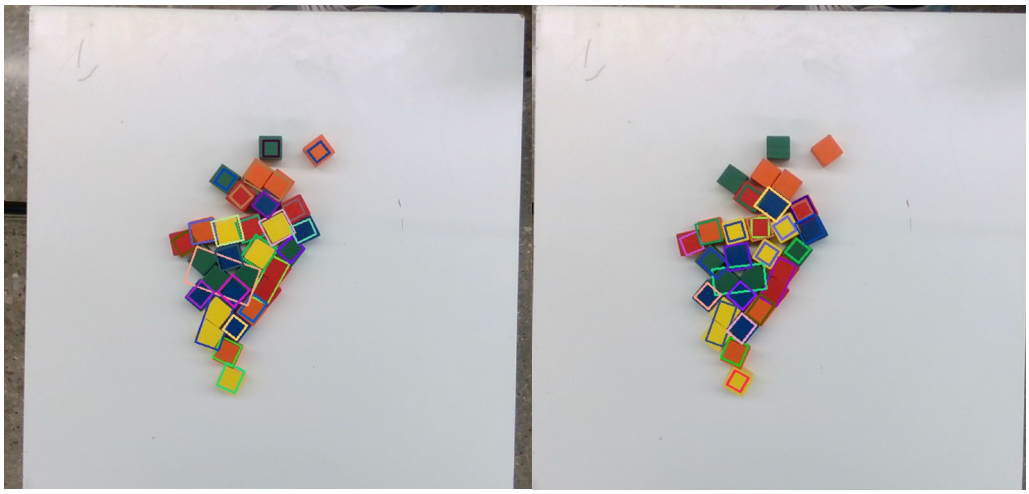

Figure 5: HSV filtering (left) and RGB filtering (right) applied to a sample image.

Figure 5 depicts the bounding boxes isolated from an image using HSV and RGB filtering.

System overview
Figure 6, below, summarizes the perception system's design.

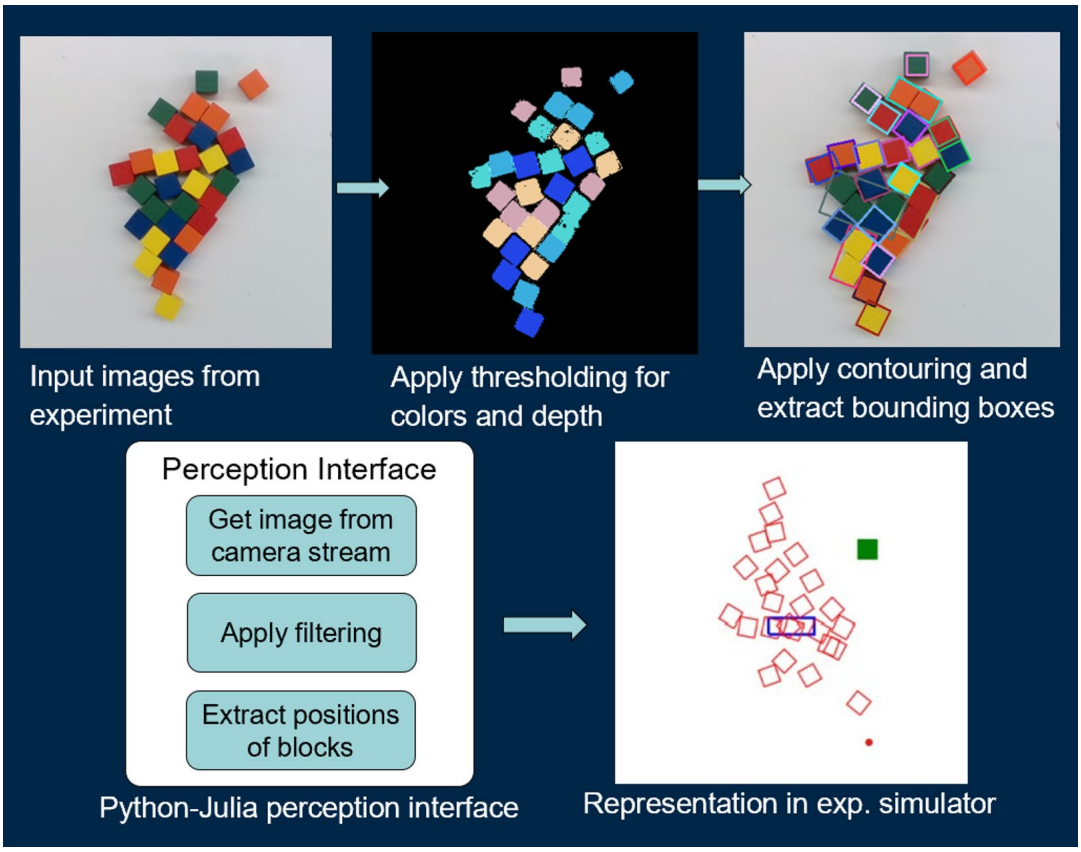

Figure 6: Perception system, with final results loaded into experiment simulator and parallelized planning code.

Dataset performance
The accuracy of each dataset, tested with per-color RGB and HSV thresholding and contouring, is given below.

Accuracy of HSV/RGB filtering on each dataset

Mode/Data	Multicolored	Primary	Red only
HSV	90.5%	79.2%	83.3%
RGB	91.8%	85.8%	49.8%

Accuracy was measured by recording the total number of blocks detected vs the actual number of blocks present in the image. False positives were extremely rare so I used an accuracy measurement that focused primarily on false negatives, or groupings of numerous blocks.

The best performing dataset was the RGB on the multicolored blocks, which is somewhat unexpected since HSV was assumed to be more useful for image processing. However, these results could be influenced by imperfect tuning of the color thresholds. The threshold values should be re-tuned with the final experimental setup.

The HSV detection of the red blocks has a high performance, so it could be a good choice for the final experiment if additional testing confirms these results.

Submitted Deliverables

1. Test datasets on NexiGo and Xtion cameras (shared Google Drive)
2. Test datasets with red, primary-colored, and multicolored blocks on Intel RealSense camera (shared Google Drive)
3. `findthresh.py`, a script that allows the user to tune RGB or HSV thresholds for a color in an image and see the contouring results in real time (GitHub repository)
4. `block_detector.py`, a script that captures an image from a ROS camera stream and applies the contouring steps, extracting the locations and orientations of the blocks in space
5. `block_detection.jl`, a Julia script that uses the package PyCall to call the `block_detector.py` script on command and convert the output to Julia format

FUTURE WORK

This project is still in progress. The research team has currently completed implementing some baseline algorithms on a CPU and GPU and run preliminary tests. To complete the project, the team will implement the parallelized belief-space planning algorithm on FPGA hardware, then run simulated to verify its effectiveness on virtual toy experiments. Then we will run the sweeper experiment on the Digit robot with the perception system, comparing the baseline implementations of the planning algorithm to the updated, accelerated approach with FPGAs. Before running the experiment, it might be necessary to tune the perception system's HSV filter settings using the final setup, which will have different lighting, table reflectiveness, and shadows, for more robust thresholding. Work on the project is expected to be completed by the fall and final results will be determined before the work is published.

CONCLUSION

I have presented my work completed this semester for my Honors Engineering capstone project and discussed my process and final design for a perception system for a block-sweeping experiment meant to aid hardware-accelerated belief-space planning research for autonomous robotics.

Working on this capstone allowed me to learn about the process of completing research in the field of robotics and to take ownership over a component of the experiment. In my independent work, I was able to practice researching a problem, determining requirements, weighing options for a solution, and integrating within a larger project.

Although my work is done, there is still lots to be accomplished within improving belief-space planning algorithms for robotics. I'd like to thank those in the Laboratory for Progress who helped enable me to complete this capstone, especially my advisor Professor Chad Jenkins, Anthony Opipari, Alphonsus Adu-Bredu, and Cameron Kisailus.

REFERENCES

[1] Ye, N., Somani, A., Hsu, D., Lee, W. S. “DESPOT: Online POMDP Planning with Regularization.” Journal of Artificial Intelligence Research, vol. 58, 2017, pp. 231-266. DOI: https://doi.org/10.48550/arXiv.1609.03250.

[2] Cai, P., Luo, Y., Hsu, D., Lee, W. “HyP-DESPOT: A Hybrid Parallel Algorithm for Online Planning under Uncertainty.” RSS, 2018. DOI: https://doi.org/10.1177/0278364920937074.

[3] Adu-Bredu, A., Zeng, Z., Pusalkar, N., Jenkins, O.C. Elephants Don’t Pack Groceries: Robot Task Planning for Low Entropy Belief . IEEE Robotics and Automation Letters, vol. 7, 2022. DOI: https://doi.org/10.1109/LRA.2021.3116327.

[4] Dwibetti, D., Malisiewicz, T., Badrinarayan, V., Rabinovich, A. “Deep Cuboid Detection: Beyond 2D Bounding Boxes.” 2016. DOI: https://doi.org/10.48550/arXiv.1611.10010.

[5] Wei, Q., Jiang, Z., Zhang, H. “Robust Spacecraft Component Detection in Point Clouds.” Sensors, vol. 18, 2018. DOI: https://doi.org/10.3390/s18040933.

[6] Xiao, J., Russell, B., Torralba, A. “Localizing 3D cuboids in single-view images.” NeurIPS Proceedings, vol. 25, 2012. DOI: https://doi.org/10.5555/2999134.2999218.