

Learning Deep Controllable and Structured Representations for Image Synthesis, Structure Prediction and Beyond

by

Xinchen Yan

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2019

Doctoral Committee:

Associate Professor Honglak Lee, Chair
Associate Professor Jason Corso
Assistant Professor Jia Deng
Professor Benjamin Kuipers

“Three passions, simple but overwhelmingly strong, have governed my life: the longing for love, the search for knowledge, and unbearable pity for the suffering of mankind.”

— Bertrand Russell

“You don’t become what you want, you become what you believe.”

— Oprah Winfrey

Xinchen Yan

`xcyan@umich.edu`

ORCID ID: 0000-0003-1019-5537

©Xinchen Yan 2019

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisor, Honglak Lee. It has been a privilege and truly an honor to have him as a mentor. Honglak has been an amazing mentor and advisor, not only in research, but also other aspects of academic life. I cannot thank you enough.

I would like to thank all my committee members, Benjamin Kuipers, Jason Corso, and Jia Deng. It has been truly a great privilege and honor to have them as mentors, and I received invaluable advice and constructive feedback for the research. Thank you so much.

I would like to thank all my collaborators in academia and industry labs: Zeynep Akata (UvA), Bernt Schiele (MPI), Karen Livescu (TTIC), Abhinav Gupta (CMU), Kihyuk Sohn (NEC), Jimei Yang (Adobe), Scott Reed (Deepmind), Ersin Yumer (Uber), Weiran Wang (Amazon), James Davidson (Third Wave Automation), Mohi Khansari (Google X), Yunfei Bai (Google X), Kalyan Sunkavalli (Adobe), Eli Shechtman (Adobe), Sunil Hadap (Amazon), Changhan Wang (Facebook), Lajanugen Logeswaran, Yijie Guo, Jasmine Hsu (Google), Arkanath Pathak (Google), Ruben Villegas, Akash Rastogi, Sören Pirk (Google), Louis Gong (Google X), Max Smith, Kanika Kochhar, Marcie Rubin (NYU), Stephen Warren (NYU), James Wrobel, Bo Li (UIUC), Chaowei Xiao, Lei Yang (CUHK), and Haonan Qiu (CUHK). Thank you for your support, which played a crucial role in finishing up my PhD thesis. Specifically, I would like to thank Jimei Yang and Kihyuk Sohn, who gave me invaluable support and guidance in the first two years of my PhD study. Many thanks to former

and current members from UM Machine Learning group, for their valuable discussions and warm friendship: Yuting Zhang, Seunghoon Hong, Roni Mittelman, Junhyuk Oh, Ye Liu, Yi Zhang, Brian Wang, Wenling Shang, Rui Zhang, Kibok Lee, Sungryull Sohn, Jongwook Choi, Yunseok Jang, Kimin Lee, and Wilka Carvalho.

I would like to thank Adobe Research, Google Brain, and X Lab for offering me great industry-lab experience. I really enjoyed my year-long internships in California and I feel so lucky to meet and chat with many smart and nice colleagues there, who indeed broadened my eyesight not only in the research I have been focused on for my PhD thesis, but also other research topics. I would like to also thank many other friends in the academia and industry around the globe.

Besides, I gratefully acknowledge the financial support from MCubed initiative, Office of Naval Research (ONR N00014-13-1-0762), National Science Founding (NSF CMMI-1266184 and NSF CAREER IIS-1453651), DARPA Explainable AI (XAI) program (313498), Adobe gift fundings, Google Faculty Research Award (to Honglak), Sloan Research Fellowship (to Honglak), Adobe Research Fellowship (to Xinchun), Google PhD Fellowship (to Xinchun), Rackham Predoctoral Fellowship (to Xinchun), and the donation of GPUs from NVIDIA.

I would like to thank my friends for their countless support, patience, and company. Specifically, many thanks to my close friends in US and China: Yiqian Gan, Kevin Mingyang Zhou, Chengwei Dai, Faye Fengyi Liu, Joyce Xuwei Zhang, Amanda Ying Sheng, Alan Subing Qu, and Sophia Feiya Chen. I would like to thank my friends and office mates in Ann Arbor. They have been offering warm friendships and companies even during the dark and cold winter nights in Michigan: Wenqiang Huang, Xueman Zhao, Xiao Li, Xingyu Li, Biyun Jiang, Haotian Chang, Yan Zhao, Yuhan Lin, Yundi Wang, Ke Yang, Meiyin Liu, Xianglong Wang, Lunyu Zhang, Wenqin Qiang, Shiwei Xu, Tianjia Jin, Zhuang Han, Chenxi Li, Sylvia Xu, Yu Wang, Xiuyuan Yang, Maggie Xu, Yi Wang, Beiming Liu, Jing Leng, Jiecao Yu, Xintong

Wang, Yike Liu, Konstantinos Pappas, Aparna Garimella, Janarthanan Rajendran, Huan Feng, Dongyao Chen, Zhefan Ye, Zeyu Zheng, Johnny Chao, Chenliang Xu, Daniel DeTone, Jonathan Chandler Stroud, Nate Harada, Alejandro Newell, Michael Chang, Mahmoud Azab, Crisian Paul Bara, Luowei Zhou, Abhishek Bafna, Oana Ignat, Mason Wright, Shun Zhang, Crisina Garbacea, Santiago Castro, Lynn Garrett, Weifeng Chen, Dawei Yang, Lanlan Liu, Haozhu Wang and many others.

I would also like to thank my friends in California, who turned my final year into a wonderful and enjoyable memory: Ioana Bica, Madalina Hurmuz, Alex Tampkin, Yuke Liao, Li Ding, Linchen Sun, Siqi Guo, Chengyuan Yan, Yu Zhang, Xi Yi, Sherry Yang, Bo Dai, Weiyue Wang, Yi Zhou, Xun Huang, Xin Zheng, Yao Qin, Qi Sun, Cynthia Jingwan Lv, Zhaolun Su, Zhiyuan Zuo, Betty Beidi Chen, Chen Luo, Xiaoran Xu, Yanfei Wu, Qing Ye, Jingchen Feng, Yang Li, Piaoyang Cui, Yi Chen, Yi Xie, Chen Sun, Xin Wang, Qian Yu, Chen Ma, Zangnan Yu, Xiran Bai, Yinghan Xu, Han Yan, Nero Li, Xiaochen Sun, Yuzhu Qin, Shujie Jiang and Jianan Zhan. I cannot thank you enough.

Finally, I would like to thank my parents and family for their countless love and support. They have always been a source of comfort and hope in difficult times.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	ii
LIST OF FIGURES	viii
LIST OF TABLES	xii
ABSTRACT	xiii
CHAPTER	
I. Introduction	1
1.1 Motivation	1
1.2 Organization of the Thesis	2
1.2.1 Learning Controllable Representations for Attribute-to-Image Generation (Chapter II)	3
1.2.2 Learning Controllable and Structured Representations for Semantic Image Manipulation (Chapter III)	3
1.2.3 Learning Structured Representations for Human Motion Generation (Chapter IV)	4
1.2.4 Learning Geometry Representations for Single-View 3D Object Reconstruction (Chapter V)	4
1.2.5 Learning Geometry-aware Deep Representation for 6-DOF Grasping (Chapter VI)	5
II. Learning Controllable Representations for Attribute-to-Image Generation	7
2.1 Introduction	7
2.2 Related Work	10
2.3 Attribute-conditioned Generative Modeling of Images	12
2.3.1 Base Model: Conditional Variational Auto-Encoder (CVAE)	12
2.3.2 Disentangling CVAE with a Layered Representation	14

2.4	Posterior Inference via Optimization	17
2.5	Experiments	18
2.5.1	Attribute-conditioned Image Generation	21
2.5.2	Attribute-conditioned Image Reconstruction and Completion	25
2.6	Discussions	27
III. Learning Controllable and Structured Representations for Semantic Image Manipulation		28
3.1	Introduction	28
3.2	Related Work	30
3.3	Hierarchical Image Manipulation	32
3.3.1	Structure generator	33
3.3.2	Image generator	35
3.4	Experiments	37
3.4.1	Implementation Details	37
3.4.2	Quantitative Evaluation	40
3.4.3	Qualitative Analysis	42
3.5	Discussions	45
IV. Learning Structured Representations for Human Motion Generation		47
4.1	Introduction	47
4.2	Related Work	50
4.3	Problem Formulation and Methods	52
4.3.1	Preliminaries	52
4.3.2	Motion-to-Motion Transformations in Latent Space	54
4.3.3	Additive Transformations in Latent Space	55
4.4	Experiments	58
4.4.1	Multimodal Motion Generation	59
4.4.2	Analogy-based Motion Transfer	64
4.4.3	Towards Multimodal Hierarchical Video Generation	65
4.5	Discussions	66
V. Learning Geometry Representations for Single-View 3D Object Reconstruction		68
5.1	Introduction	68
5.2	Related Work	70
5.3	Problem Formulation	71
5.3.1	Learning to Reconstruct Volumetric 3D Shape from Single-View	72
5.3.2	Perspective Transformer Networks	74

5.3.3	Training	75
5.4	Experiments	76
5.4.1	Training on a single category	79
5.4.2	Training on multiple categories	82
5.4.3	Out-of-Category Tests	82
5.5	Discussions	84
VI. Learning Geometry-aware Deep Representation for 6-DOF Grasping		85
6.1	Introduction	85
6.2	Related Work	88
6.3	Multi-objective framework with geometry-aware representation	90
6.3.1	Learning generative geometry-aware representation from RGBD input	90
6.3.2	Depth supervision with in-network projection layer	91
6.3.3	Viewpoint-invariant geometry-aware representation with multi-view supervision	93
6.3.4	Learning predictive grasping interaction with geometry-aware representation.	94
6.3.5	DGGN: Deep geometry-aware grasping network.	95
6.4	Experiments	96
6.4.1	Dataset collection	96
6.4.2	Implementation details	98
6.4.3	Visualization: 3D shape generation	99
6.4.4	Model evaluation: Grasping outcome prediction	101
6.4.5	Application: Analysis-by-synthesis grasping planning.	101
6.5	Discussions	103
VII. Future work		105
APPENDIX		107
A.1	Derivation of disCVAE Objective	108
A.2	disCVAE Network Architecture	109
A.3	Attribute-conditioned Image Progression	111
A.4	Datasets for Motion Generation.	112
A.5	MT-VAE Network Architecture.	113
A.6	MT-VAE Implementation Details	113
A.7	Details regarding Perspective Transformer Network	114
BIBLIOGRAPHY		118

LIST OF FIGURES

Figure

2.1	An example that demonstrates the problem of conditioned image generation from visual attributes. We assume a vector of visual attributes is extracted from a natural language description, and then this attribute vector is combined with learned latent factors to generate diverse image samples.	8
2.2	Graphical model representations of attribute-conditioned image generation models (a) without (CVAE) and (b) with (disCVAE) disentangled latent space.	14
2.3	Attribute-conditioned image generation.	21
2.4	Attribute-conditioned image progression. The visualization is organized into six attribute groups (e.g., “gender”, “age”, “facial expression”, “eyewear”, “hair color” and “primary color (blue vs. yellow)”). Within each group, the images are generated from $p_\theta(x y, z)$ with $z \sim \mathcal{N}(0, I)$ and $y = [y_\alpha, y_{\text{rest}}]$, where $y_\alpha = (1 - \alpha) \cdot y_{\text{min}} + \alpha \cdot y_{\text{max}}$. Here, y_{min} and y_{max} stands for the minimum and maximum attribute value respectively in the dataset along the corresponding dimension.	23
2.5	Analysis: Latent Space Disentangling.	24
2.6	Attribute-conditioned image reconstruction and completion.	26
3.1	Overall pipeline of the proposed semantic manipulation framework.	30
3.2	Architecture of the structure generator. Given a masked layout \bar{M} and a binary mask \bar{B} encoding the class and location of the object, respectively, the model produces the manipulated layout \hat{M} by the outputs from the two-stream decoder corresponding to the binary mask of object and semantic label map of entire region inside the box.	34
3.3	Architecture of the image generator. Given a masked image \bar{I} and the semantic layout \hat{M} , the model encodes visual style and semantic structure of the object using separate encoding pathways and produces the manipulated image.	37
3.4	Comparisons between variants of the proposed method. The last two rows (TwoStream and TwoStream-Pred) correspond to our full model using ground-truth and predicted layout, respectively.	39

3.5	Qualitative comparisons to the baselines in Table 3.4. The first two columns show the masked image and ground-truth layout used as input to the models (except <code>TwoStream-Pred</code>). The manipulated objects are indicated by blue arrows. Best viewed in color.	39
3.6	Generation results on various locations in an image.	41
3.7	Generation results in diverse contexts.	41
3.8	Controlling object color with style vector. Colors used for manipulation are presented at left-upper corners.	43
3.9	Examples of manipulation of multiple objects in images. The line style indicates manipulation operation (solid: addition, dotted: deletion) and the color indicates the object class.	44
3.10	Example of data-driven image manipulation. We manipulate the target image by transferring bounding boxes from source image (blue boxes).	45
3.11	Examples of image manipulation results on indoor images.	45
4.1	Top: Learning motion sequence generation using <i>Motion Transformation VAE</i> . Bottom: Generating multiple future motion sequences from the transformation space.	48
4.2	Illustrations of different models for motion sequence generation. $s(x_{1:T})$ indicates the hidden state of the Encoder LSTM at time T	54
4.3	Illustrations of cycle consistency in MT-VAE variations.	57
4.4	Multimodal Sequence Generation. Given an input sequence (green boundary), we generate future sequences (red boundary). We predict 32 frames given 8 frames for face motion, and 64 frames given 16 frames for human body motion. Given the initial frames as condition, we demonstrate (top to bottom) the ground truth sequence, Prediction LSTM, Vanilla VAE, and our MT-VAE model. Overall, our model produces (1) diverse and structured motion patterns and (2) more natural transitions from the last frame observed to the first frame generated (See the subtle mouth shape and scale change from the last observed frame to the first generated one).	62
4.5	Analogy-based motion transfer. Given three motion sequences A, B, and C from test set, the objective is to extract the motion mode transition from A to B and then apply it to animate the future starting from sequence C. For fair comparison, we set the encoder Gaussian distribution parameter σ to zero during evaluation.	65
4.6	Multimodal Hierarchical video generation. Top rows: Face video generation results from 8 observed frames. Bottom rows: Human video generation results from 16 observed frames.	66
5.1	(a) Understanding 3D object from learning agent’s perspective; (b) Single-view 3D volume reconstruction with perspective transformation. (c) Illustration of perspective projection. The minimum and maximum disparity in the screen coordinates are denoted as d_{min} and d_{max}	72
5.2	Illustration of network architecture.	76

5.3	Single-class results. GT: ground truth, PR: PTN-Proj, CO: PTN-Comb, VO: CNN-Vol (Best viewed in digital version. Zoom in for the 3D shape details). The angles are shown in the parenthesis. Please also see more examples and video animations on the project webpage.	78
5.4	View-dependent IU. For illustration, images of a sample chair with corresponding azimuth angles are shown below the curves. For example, 3D reconstruction from 0° is more difficult than from 30° due to self-occlusion.	80
5.5	Multiclass results. GT: ground truth, PR: PTN-Proj, CO: PTN-Comb, VO: CNN-Vol (Best viewed in digital version. Zoom in for the 3D shape details). The angles are shown in the parenthesis. Please also see more examples and video animations on the project webpage.	81
5.6	Out-of-category results. GT: ground truth, PR: PTN-Proj, CO: PTN-Comb, VO: CNN-Vol (Best viewed in digital version. Zoom in for the 3D shape details). The angles are shown in the parenthesis. Please also see more examples and video animations on the project webpage.	83
6.1	Learning grasping interactions from demonstrations with deep geometry-aware representations. First, we learn to build mental geometry-aware representation by reconstructing the 3D scene with 2.5D training data. Second, we learn to predict grasping outcome with its internal geometry-aware representation.	86
6.2	Illustration of DGGN (deep geometry-aware grasping network). Our DGGN has a shape generation network and an outcome prediction network. The shape generation network has a 2D CNN encoder, 3D CNN decoder, and a global sampling layer (detailed in Sec. 6.3.2). Our outcome prediction network has a 2D CNN encoder, a local sampling layer (detailed in Sec. 6.3.4), and a fully-connected prediction network.	90
6.3	Illustrations of our VR-Grasping-101 dataset.	97
6.4	Visualization: 3D shape generation from single-view RGBD. (a) The performance on training (seen) objects. (b) The performance on testing (novel) objects. (c) Local geometry inference from generated occupancy grid.	98
6.5	Visualization: grasping optimization with CEM based on the grasping prediction output. In each row, we selected three representative steps in grasping optimization (in sequential order from left to right). Red box represents a failure grasp while green box represents a successful grasp.	102
A.1	Network Architecture for disentangling CVAE	110

A.2	Attribute-conditioned Image Progression. The visualization is organized into eight attribute groups (e.g., “gender”, “age”, “race”, “eyewear”, “facial expression”, “hair color”, “primary color (blue vs. yellow)”, and “primary color (black vs. white)”). Within each group, the images are generated from $p_\theta(x y, z)$ with $z \sim \mathcal{N}(0, I)$ and $y = [y_\alpha, y_{rest}]$, where $y_\alpha = (1 - \alpha) \cdot y_{min} + \alpha \cdot y_{max}$. Here, y_{min} and y_{max} stands for the minimum and maximum attribute value respectively in the dataset along the corresponding dimension.	111
A.3	Illustration of perspective projection. The minimum and maximum disparity in the screen coordinates are denoted as d_{min} and d_{max} . . .	115

LIST OF TABLES

Table

2.1	Quantitative comparisons on face reconstruction and completion tasks.	27
3.1	Quantitative comparison to other methods. Context Encoder and Pix2PixHD refer to the previous work (<i>Pathak et al.</i> , 2016) and (<i>Wang et al.</i> , 2017), respectively.	41
4.1	Quantitative evaluations for multimodal motion generation. We compare against two simple data-driven baselines for quantitative comparison: <i>Last-step Motion</i> that recursively applies the motion (velocity only) from the last step observed; <i>Sequence Motion</i> that recursively adds the average sequence velocity from the observed frames. Top: Results on Aff-Wild with facial expression coefficients. Bottom: Results on Human3.6M with 2D joints.	59
4.2	Crowd-sourced Human Evaluations on Human3.6M. *We did not include Prediction LSTM for the diversity evaluation, as it makes deterministic prediction.	63
4.3	Ablation Study on Different variants of MT-VAE (add) model: We evaluate models trained without motion coherence objective, without cycle consistency objective, and the model with context-free latent decoder.	63
5.1	Prediction IU using the models trained on chair category. Below, chair corresponds to the setting where each object is observable with full azimuth angles, while chair-N corresponds to the setting where each object is only observable with a narrow range (subset) of azimuth angles.	79
5.2	Prediction IU using the models trained on large-scale datasets.	81
5.3	Prediction IU in out-of-category tests.	82
6.1	Grasping Outcome prediction accuracy from seen elevation angles.	100
6.2	Grasping Outcome prediction accuracy from novel elevation angles.	100
6.3	Grasping planning on novel objects: success rate by optimizing for up to 20 steps.	101

ABSTRACT

Generative modeling is a frontier research topic in machine learning and AI. Despite the recent success in image synthesis, developing a general form of deep generative models on complex data (e.g., multi-modal and structured data) also directly applicable to real-world tasks remains a challenging open problem. The major challenges include high-dimensional representation space, many entangled factors of variation, and lack of existing deep modules for this data generation process.

In the thesis, I introduce the concept of controllable representations, a set of factors as intermediate representations, for deep generative modeling. In the context of attribute-to-image synthesis, we consider controllable units as (1) semantic factors described by the visual attributes and (2) other related factors not included in the input attributes for image synthesis (e.g., such as pose and background color). To facilitate efficient learning of such controllable units for attribute-to-image synthesis, I explore novel deep structured modules that can be trained in auto-encoding style that synthesizes images from controllable units. In addition, I demonstrate the representation power of such design in conditional generation (e.g., control partial set of units while keeping the rest unchanged) as well as other related applications including image completion via analysis-by-synthesis optimization.

For the rest of the thesis, I investigate and propose several variations to learn controllable and structured representations in several related problems including (1)

image manipulation with semantic structures (e.g., object bounding boxes), (2) human motion prediction with transformation-based representations, and (3) 3D shape prediction from single-view with geometry-aware modules. The case studies in the thesis demonstrate not only the representation power but also a common aspect that the representations can be learned in an unsupervised or weakly-supervised manner. In the end, I discuss several future directions in learning deep controllable and structured modules for other multi-modal and structured data as well as the application to adversarial learning.

CHAPTER I

Introduction

1.1 Motivation

Generative models, describing the data generation process, is one of the key research topics in machine learning and AI. There has been a line of early research on unsupervised representation learning using deep generative models on simple and toy data. Efficient learning using deep generative models on real-world data with domain-specific structures is still an open problem in the research community. One possible solution is to learn the representation in a purely unsupervised way with black-box deep modules. Several recent work has demonstrated the success of this approach on large-scale image datasets (e.g., ImageNet, CelebA) using very powerful computation resources. One weakness is that the learned representation may not generalize in novel but related settings without sufficient training data or when the data is highly complicated. Approximating the real-world data distribution using deep generative models is essentially difficult due to the high-dimensional representation space and many entangled factors of variation involved in the data generation process. For example, imagery observations of the environment are entangled representations of intrinsic properties (e.g., texture, geometry and material), as well as its extrinsic environmental properties such as illumination.

Alternatively, one can apply certain regularization to constrain the representation

space so as to learn disentangled or *controllable* units in the representation. Specifically, one advantage of learning *controllable* representation is that the units can be directly applied for conditional generation and manipulation (e.g., control partial set of units while keeping the rest unchanged). In addition, the controllable representation allows for relatively easier generalization and task transfer (to other related tasks) compared to representations trained without certain regularization or conditioning.

Motivated by this, the thesis mainly addresses three aspects of the problem: (1) how to effectively learn a *controllable* representation using deep neural networks; (2) how to better incorporate domain knowledge into data generation process with *structured* deep in-network modules; and (3) how to achieve robust performance for other supervised or semi-supervised tasks with additional generative objectives. On the application side, the thesis can be categorized into the following related topics: controllable image generation (Chapter II and III) and structure prediction using deep generative models (Chapter IV, V, and VI).

1.2 Organization of the Thesis

This thesis is organized in 7 chapters including the introduction (Chapter I), and the conclusion and future work (Chapter VII). The main chapters (II – VI) are divided into two parts according to the application domain. As one of the most common digital data formats, images have been the key data modality that has advanced fundamental research in machine learning and computer vision for decades. The Chapter II presents the problem of attribute-conditional image synthesis and a layered module with variational auto-encoders. The Chapter III further investigates the high-resolution image manipulation conditioned on structured semantic representations. As image synthesis is a special form of structure prediction problem, we discuss possible extensions to learning controllable and structured representations for motion data (in Chapter IV), 3D shape data (Chapter V), and human-object interaction

(Chapter VI). For each chapter, we briefly state the problem, our approach, and further discussions.

1.2.1 Learning Controllable Representations for Attribute-to-Image Generation (Chapter II)

This chapter investigates a novel problem of generating images from visual attributes. We model the image as a composite of foreground and background and develop a layered generative model with disentangled latent variables that can be learned end-to-end using a variational auto-encoder. We experiment with natural images of faces and birds and demonstrate that the proposed models are capable of generating realistic and diverse samples with disentangled latent representations. We use a general energy minimization algorithm for posterior inference of latent variables given novel images. The learned generative models show excellent quantitative and visual results in the tasks of attribute-conditioned image reconstruction and completion.

1.2.2 Learning Controllable and Structured Representations for Semantic Image Manipulation (Chapter III)

Understanding, reasoning, and manipulating semantic concepts of images have been a fundamental research problem for decades. Previous work mainly focused on direct manipulation on natural image manifold through color strokes, key-points, textures, and holes-to-fill. In this work, we present a novel hierarchical framework for semantic image manipulation. Key to our hierarchical framework is that we employ structured semantic layout as our intermediate representation for manipulation. Initialized with coarse-level bounding boxes, our structure generator first creates pixel-wise semantic layout capturing the object shape, object-object interactions, and object-scene relations. Then our image generator fills in the pixel-level

textures guided by the semantic layout. Such framework allows a user to manipulate images at object-level by adding, removing, and moving one bounding box at a time. Experimental evaluations demonstrate the advantages of the hierarchical manipulation framework over existing image generation and context hole-filling models, both qualitatively and quantitatively. Benefits of the hierarchical framework are further demonstrated in applications such as semantic object manipulation, interactive image editing, and data-driven image manipulation.

1.2.3 Learning Structured Representations for Human Motion Generation (Chapter IV)

Long-term human motion can be represented as a series of *motion modes*—motion sequences that capture short-term temporal dynamics—with transitions between them. We leverage this structure and present a novel *Motion Transformation Variational Auto-Encoders (MT-VAE)* for learning motion sequence generation. Our model jointly learns a feature embedding for motion modes (that the motion sequence can be reconstructed from) and a feature transformation that represents the transition of one motion mode to the next motion mode. Our model is able to generate multiple diverse and plausible motion sequences in the future from the same input. We apply our approach to both facial and full body motion, and demonstrate applications like analogy-based motion transfer and video synthesis.

1.2.4 Learning Geometry Representations for Single-View 3D Object Reconstruction (Chapter V)

Understanding the 3D world is a fundamental problem in computer vision. However, learning a good representation of 3D objects is still an open problem due to the high dimensionality of the data and many factors of variation involved. In this chapter, we investigate the task of single-view 3D object reconstruction from a learning

agent’s perspective. We formulate the learning process as an interaction between 3D and 2D representations and propose an encoder-decoder network with a novel projection loss defined by the perspective transformation. More importantly, the projection loss enables the unsupervised learning using 2D observation without explicit 3D supervision. We demonstrate the ability of the model in generating 3D volume from a single 2D image with three sets of experiments: (1) learning from single-class objects; (2) learning from multi-class objects and (3) testing on novel object classes. Results show superior performance and better generalization ability for 3D object reconstruction when the projection loss is involved.

1.2.5 Learning Geometry-aware Deep Representation for 6-DOF Grasping (Chapter VI)

This chapter focuses on the problem of learning 6-DOF grasping with a parallel jaw gripper in simulation. Our key idea is constraining and regularizing grasping interaction learning through 3D geometry prediction. We introduce a deep geometry-aware grasping network (DGGN) that decomposes the learning into two steps. First, we learn to build mental geometry-aware representation by reconstructing the scene (i.e., 3D occupancy grid) from RGBD input via generative 3D shape modeling. Second, we learn to predict grasping outcome with its internal geometry-aware representation. The learned outcome prediction model is used to sequentially propose grasping solutions via analysis-by-synthesis optimization. Our contributions are fourfold: (1) To best of our knowledge, we are presenting for the first time a method to learn a 6-DOF grasping net from RGBD input; (2) We build a grasping dataset from demonstrations in virtual reality with rich sensory and interaction annotations. This dataset includes 101 everyday objects spread across 7 categories, additionally, we propose a data augmentation strategy for effective learning; (3) We demonstrate that the learned geometry-aware representation leads to about 10% relative performance improvement

over the baseline CNN on grasping objects from our dataset. (4) We further demonstrate that the model generalizes to novel viewpoints and object instances.

CHAPTER II

Learning Controllable Representations for Attribute-to-Image Generation

2.1 Introduction

Generative image modeling is of fundamental interest in computer vision and machine learning. Early works (*Srivastava et al.*, 2003; *Tu*, 2007; *Lee et al.*, 2009; *Ranzato et al.*, 2010; *Le Roux et al.*, 2011) studied statistical and physical principles of building generative models, but due to the lack of effective feature representations, their results are limited to textures or particular patterns such as well-aligned faces. Recent advances on representation learning using deep neural networks (*Krizhevsky et al.*, 2012; *Simonyan and Zisserman*, 2014) nourish a series of deep generative models that enjoy joint generative modeling and representation learning through Bayesian inference (*Tang and Salakhutdinov*, 2013; *Bengio et al.*, 2013; *Rezende et al.*, 2014; *Kingma and Welling*, 2014; *Kingma et al.*, 2014; *Gregor et al.*, 2015) or adversarial training (*Goodfellow et al.*, 2014; *Denton et al.*, 2015). Those works show promising results of generating natural images, but the generated samples are still in low resolution and far from being perfect because of the fundamental challenges of learning unconditioned generative models of images.

In this chapter, we are interested in generating object images from high-level de-

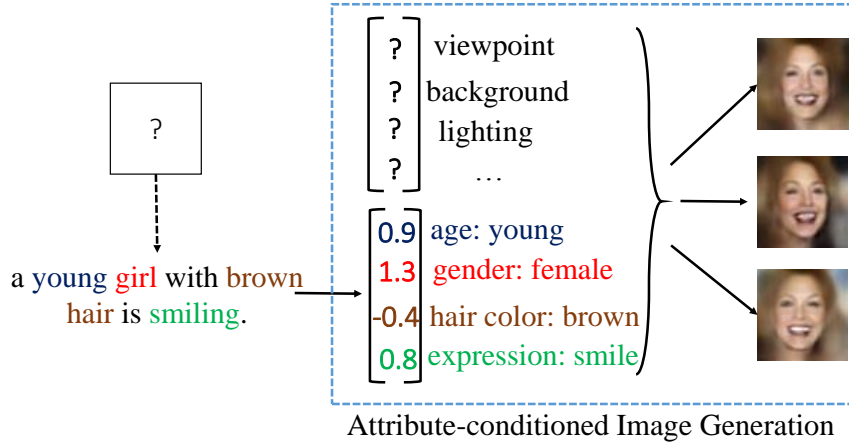


Figure 2.1: An example that demonstrates the problem of conditioned image generation from visual attributes. We assume a vector of visual attributes is extracted from a natural language description, and then this attribute vector is combined with learned latent factors to generate diverse image samples.

scription. For example, we would like to generate portrait images that all match the description “a young girl with brown hair is smiling” (Figure 2.1). This conditioned treatment reduces sampling uncertainties and helps generating more realistic images, and thus has potential real-world applications such as forensic art and semantic photo editing (Laput *et al.*, 2013; Yang *et al.*, 2011a; Kemelmacher-Shlizerman *et al.*, 2014). The high-level descriptions are usually natural languages, but what underlies its corresponding images are essentially a group of facts or visual attributes that are extracted from the sentence. In the example above, the attributes are (hair color: brown), (gender: female), (age: young) and (expression: smile). Based on this assumption, we propose to learn an attribute-conditioned generative model.

Indeed, image generation is a complex process that involves many factors. Other than enlisted attributes, there are many unknown or latent factors. It has been shown that those latent factors are supposed to be interpretable according to their semantic or physical meanings (Kulkarni *et al.*, 2015; Dosovitskiy *et al.*, 2015; Reed *et al.*, 2014). Inspired by layered image models (Wang and Adelson, 1994; Nitzberg and Mumford, 1990), we disentangle the latent factors into two groups: one related to

uncertain properties of foreground object and the other related to the background, and model the generation process as layered composition. In particular, the foreground is overlaid on the background so that the background visibility depends on the foreground shape and position. Therefore, we propose a novel layered image generative model with disentangled foreground and background latent variables. The entire background is first generated from background variables, then the foreground variables are combined with given attributes to generate object layer and its shape map determining the visibility of background and finally the image is composed by the summation of object layer and the background layer gated by its visibility map. We learn this layered generative model in an end-to-end deep neural network using a variational auto-encoder (Kingma and Welling, 2014) (Section 2.3). Our variational auto-encoder includes two encoders or recognition models for approximating the posterior distributions of foreground and background latent variables respectively, and two decoders for generating a foreground image and a full image by composition. Assuming the latent variables are Gaussian, the whole network can be trained end-to-end by back-propagation using the reparametrization trick.

Generating realistic samples is certainly an important goal of deep generative models. Moreover, generative models can be also used to perform Bayesian inference on novel images. Since the true posterior distribution of latent variables is unknown, we propose a general optimization-based approach for posterior inference using image generation models and latent priors (Section 2.4).

We evaluate the proposed model on two datasets, the Labeled Faces in the Wild (LFW) dataset (Huang *et al.*, 2007) and the Caltech-UCSD Birds-200-2011 (CUB) dataset (Wah *et al.*, 2011). In the LFW dataset, the attributes are 73-dimensional vectors describing age, gender, expressions, hair and many others (Kumar *et al.*, 2009). In the CUB dataset, the 312-dimensional binary attribute vectors are converted from descriptions about bird parts and colors. We organize our experiments in the fol-

lowing two tasks. First, we demonstrate the quality of attribute-conditioned image generation with comparisons to nearest-neighbor search, and analyze the disentangling performance of latent space and corresponding foreground-background layers. Second, we perform image reconstruction and completion on a set of novel test images by posterior inference with quantitative evaluation. Results from those experiments show the superior performance of the proposed model over previous art. The contributions of this chapter are summarized as follows:

- We propose a novel problem of conditioned image generation from visual attributes.
- We tackle this problem by learning conditional variational auto-encoders and propose a novel layered foreground-background generative model that significantly improves the generation quality of complex images.
- We propose a general optimization-based method for posterior inference on novel images and use it to evaluate generative models in the context of image reconstruction and completion.

2.2 Related Work

Image generation. In terms of generating realistic and novel images, there are several recent work (*Dosovitskiy et al.*, 2015; *Gregor et al.*, 2015; *Kulkarni et al.*, 2015; *Goodfellow et al.*, 2014; *Denton et al.*, 2015; *Radford et al.*, 2015) that are relevant to ours. *Dosovitskiy et al.* (2015) proposed to generate 3D chairs given graphics code using deep convolutional neural networks, and *Kulkarni et al.* (2015) used variational auto-encoders (*Kingma and Welling*, 2014) to model the rendering process of 3D objects. Both of these models *Kulkarni et al.* (2015); *Dosovitskiy et al.* (2015) assume the existence of a graphics engine during training, from which they have 1) virtually infinite amount of training data and/or 2) pairs of rendered images that differ only in

one factor of variation. Therefore, they are not directly applicable to natural image generation. While both work (*Kulkarni et al.*, 2015; *Dosovitskiy et al.*, 2015) studied generation of rendered images from complete description (e.g., object identity, view-point, color) trained from synthetic images (via graphics engine), generation of images from an incomplete description (e.g., class labels, visual attributes) is still under-explored. In fact, image generation from incomplete description is a more challenging task and the one-to-one mapping formulation of *Dosovitskiy et al.* (2015) is inherently limited. *Gregor et al.* (2015) developed recurrent variational auto-encoders with spatial attention mechanism that allows iterative image generation by patches. This elegant algorithm mimics the process of human drawing but at the same time faces challenges when scaling up to large complex images. Recently, generative adversarial networks (GANs) (*Goodfellow et al.*, 2014; *Gauthier*, 2015; *Denton et al.*, 2015; *Radford et al.*, 2015) have been developed for image generation. In the GAN, two models are trained to against each other: a generative model aims to capture the data distribution, while a discriminative model attempts to distinguish between generated samples and training data. The GAN training is based on a min-max objective, which is known to be challenging to optimize.

Layered modeling of images. Layered models or 2.1D representations of images have been studied in the context of moving or still object segmentation (*Wang and Adelson*, 1994; *Nitzberg and Mumford*, 1990; *Williams and Titsias*, 2004; *Yang et al.*, 2012b; *Isola and Liu*, 2013). The layered structure is introduced into generative image modeling (*Le Roux et al.*, 2011; *Tang et al.*, 2012). *Tang et al.* (2012) modeled the occluded images with gated restricted Boltzmann machines and achieved good in-painting and de-noising results on well cropped face images. *Le Roux et al.* (2011) explicitly modeled the occlusion layer in a masked restricted Boltzmann machine for separating foreground and background and demonstrated promising results on

small patches. Though similar to our proposed gating in the form, these models face challenges when applied to model large natural images due to its difficulty in learning hierarchical representation based on restricted Boltzmann machine.

Multimodal Learning. Generative models of image and text have been studied in multimodal learning to model joint distribution of multiple data modalities (*Ngiam et al.*, 2011; *Srivastava and Salakhutdinov*, 2012; *Sohn et al.*, 2014). For example, *Srivastava and Salakhutdinov* (2012) developed a multimodal deep Boltzmann machine that models joint distribution of image and text (e.g., image tag). *Sohn et al.* (2014) proposed improved shared representation learning of multimodal data through bi-directional conditional prediction by deriving a conditional prediction model of one data modality given the other and vice versa. Both of these works focused more on shared representation learning using hand-crafted low-level image features and therefore have limited applications such as conditional image or text retrieval than actual generation of images.

2.3 Attribute-conditioned Generative Modeling of Images

In this section, we describe our proposed method for attribute-conditioned generative modeling of images. We first describe a conditional variational auto-encoder, followed by the formulation of layered generative model and its variational learning.

2.3.1 Base Model: Conditional Variational Auto-Encoder (CVAE)

Given the attribute $y \in \mathbb{R}^{N_y}$ and latent variable $z \in \mathbb{R}^{N_z}$, our goal is to build a model $p_\theta(x|y, z)$ that generates realistic image $x \in \mathbb{R}^{N_x}$ conditioned on y and z . Here, we refer p_θ a generator (or generation model), parametrized by θ . Conditioned image generation is simply a two-step process in the following:

1. Randomly sample latent variable z from prior distribution $p(z)$;

2. Given y and z as conditioning variable, generate image x from $p_\theta(x|y, z)$.

Here, the purpose of learning is to find the best parameter θ that maximizes the log-likelihood $\log p_\theta(x|y)$. As proposed in (Rezende et al., 2014; Kingma and Welling, 2014), variational auto-encoders try to maximize the variational lower bound of the log-likelihood $\log p_\theta(x|y)$. Specifically, an auxiliary distribution $q_\phi(z|x, y)$ is introduced to approximate the true posterior $p_\theta(z|x, y)$. We refer the base model a conditional variational auto-encoder (CVAE) with the conditional log-likelihood

$$\log p_\theta(x|y) = KL(q_\phi(z|x, y)||p_\theta(z|x, y)) + \mathcal{L}_{\text{CVAE}}(x, y; \theta, \phi),$$

where the variational lower bound

$$\mathcal{L}_{\text{CVAE}}(x, y; \theta, \phi) = -KL(q_\phi(z|x, y)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x, y)}[\log p_\theta(x|y, z)] \quad (2.1)$$

is maximized for learning the model parameters.

Here, the prior $p_\theta(z)$ is assumed to follow isotropic multivariate Gaussian distribution, while two conditional distributions $p_\theta(x|y, z)$ and $q_\phi(z|x, y)$ are multivariate Gaussian distributions: $\mathcal{N}(\mu_\theta(z, y), \text{diag}(\sigma_\theta^2(z, y)))$ and $\mathcal{N}(\mu_\phi(x, y), \text{diag}(\sigma_\phi^2(x, y)))$, respectively. We refer the auxiliary proposal distribution $q_\phi(z|x, y)$ a recognition model and the conditional data distribution $p_\theta(x|y, z)$ a generation model.

The first term $KL(q_\phi(z|x, y)||p_\theta(z))$ is a regularization term that reduces the gap between the prior $p(z)$ and the proposal distribution $q_\phi(z|x, y)$, while the second term $\log p_\theta(x|y, z)$ is the log likelihood of samples. In practice, we usually take as a deterministic generation function the mean $x = \mu_\theta(z, y)$ of conditional distribution $p_\theta(x|z, y)$ given z and y , so it is convenient to assume the standard deviation function $\sigma_\theta(z, y)$ is a constant shared by all the pixels as the latent factors capture all the data variations. We will keep this assumption for the rest of the chapter if not particularly mentioned. Thus, we can rewrite the second term in the variational lower bound as

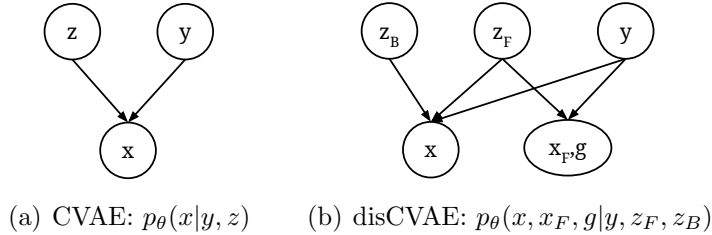


Figure 2.2: Graphical model representations of attribute-conditioned image generation models (a) without (CVAE) and (b) with (disCVAE) disentangled latent space.

reconstruction loss $L(\cdot, \cdot)$ (e.g., ℓ_2 loss):

$$\mathcal{L}_{\text{CVAE}} = -KL(q_\phi(z|x, y)||p_\theta(z)) - \mathbb{E}_{q_\phi(z|x, y)} L(\mu_\theta(y, z), x) \quad (2.2)$$

Note that the discriminator of GANs (*Goodfellow et al., 2014*) can be used as the loss function $L(\cdot, \cdot)$ as well, especially when ℓ_2 (or ℓ_1) reconstruction loss may not capture the true image similarities. We leave it for future study.

2.3.2 Disentangling CVAE with a Layered Representation

An image x can be interpreted as a composite of a foreground layer (or a foreground image x_F) and a background layer (or a background image x_B) via a matting equation (*Porter and Duff, 1984*):

$$x = x_F \odot (1 - g) + x_B \odot g, \quad (2.3)$$

where \odot denotes the element-wise product. $g \in [0, 1]^{N_x}$ is an occlusion layer or a gating function that determines the visibility of background pixels while $1 - g$ defines the visibility of foreground pixels. However, the model based on Equation (2.3) may suffer from the incorrectly estimated mask as it gates the foreground region with imperfect mask estimation. Instead, we approximate the following formulation that

is more robust to estimation error on mask:

$$x = x_F + x_B \odot g. \tag{2.4}$$

When lighting condition is stable and background is at a distance, we can safely assume foreground and background pixels are generated from independent latent factors. To this end, we propose a disentangled representation $z = [z_F, z_B]$ in the latent space, where z_F together with attribute y captures the foreground factors while z_B the background factors. As a result, the foreground layer x_F is generated from $\mu_{\theta_F}(y, z_F)$ and the background layer x_B from $\mu_{\theta_B}(z_B)$. The foreground shape and position determine the background occlusion so the gating layer g is generated from $s_{\theta_g}(y, z_F)$ where the last layer of $s(\cdot)$ is sigmoid function. In summary, we approximate the layered generation process as follows:

1. Sample foreground and background latent variables $z_F \sim p(z_F)$, $z_B \sim p(z_B)$;
2. Given y and z_F , generate foreground layer $x_F \sim \mathcal{N}(\mu_{\theta_F}(y, z_F), \sigma_0^2 I_{N_x})$ and gating layer $g \sim \text{Bernoulli}(s_{\theta_g}(y, z_F))$; here, σ_0 is a constant. The background layer (which correspond to x_B) is implicitly computed as $\mu_{\theta_B}(z_B)$.
3. Synthesize an image $x \sim \mathcal{N}(\mu_{\theta}(y, z_F, z_B), \sigma_0^2 I_{N_x})$ where $\mu_{\theta}(y, z_F, z_B) = \mu_{\theta_F}(y, z_F) + s_{\theta_g}(y, z_F) \odot \mu_{\theta_B}(z_B)$.

Learning. It is very challenging to learn our layered generative model in a fully-unsupervised manner since we need to infer about x_F , x_B , and g from the image x only. In this chapter, we further assume the foreground layer x_F (as well as gating variable g) is observable during the training and we train the model to maximize the joint log-likelihood $\log p_{\theta}(x, x_F, g|y)$ instead of $\log p_{\theta}(x|y)$. With disentangled latent variables z_F and z_B , we refer our layered model a disentangling conditional variational auto-encoder (disCVAE). We compare the graphical models of disCVAE with vanilla

CVAE in Figure 2.2. Based on the layered generation process, we write the generation model by

$$p_\theta(x_F, g, x, z_F, z_B|y) = p_\theta(x|z_F, z_B, y)p_\theta(x_F, g|z_F, y)p_\theta(z_F)p_\theta(z_B), \quad (2.5)$$

the recognition model by

$$q_\phi(z_F, z_B|x_F, g, x, y) = q_\phi(z_B|z_F, x_F, g, x, y)q_\phi(z_F|x_F, g, y) \quad (2.6)$$

and the variational lower bound $\mathcal{L}_{\text{disCVAE}}(x_F, g, x, y; \theta, \phi)$ is given by

$$\begin{aligned} \mathcal{L}_{\text{disCVAE}}(x_F, g, x, y; \theta, \phi) = & \\ & - KL(q_\phi(z_F|x_F, g, y)||p_\theta(z_F)) - \mathbb{E}_{q_\phi(z_F|x_F, g, y)} [KL(q_\phi(z_B|z_F, x_F, g, x, y)||p_\theta(z_B))] \\ & - \mathbb{E}_{q_\phi(z_F|x_F, g, y)} [L(\mu_{\theta_F}(y, z_F), x_F) + \lambda_g L(s_{\theta_g}(y, z_F), g)] \\ & - \mathbb{E}_{q_\phi(z_F, z_B|x_F, g, x, y)} L(\mu_\theta(y, z_F, z_B), x) \end{aligned} \quad (2.7)$$

where $\mu_\theta(y, z_F, z_B) = \mu_{\theta_F}(y, z_F) + s_{\theta_g}(y, z_F) \odot \mu_{\theta_B}(z_B)$ as in Equation (2.4). We further assume that $\log p_\theta(x_F, g|z_F, y) = \log p_\theta(x_F|z_F, y) + \lambda_g \log p_\theta(g|z_F, y)$, where we introduce λ_g as additional hyperparameter when decomposing the probability $p_\theta(x_F, g|z_F, y)$. For the loss function $L(\cdot, \cdot)$, we used reconstruction error for predicting x or x_F and cross entropy for predicting the binary mask g . See the Appendix A.1 for details of the derivation. All the generation and recognition models are parameterized by convolutional neural networks and trained end-to-end in a single architecture with back-propagation. We will introduce the exact network architecture in the experiment section.

2.4 Posterior Inference via Optimization

Once the attribute-conditioned generative model is trained, the inference or generation of image x given attribute y and latent variable z is straight-forward. However, the inference of latent variable z given an image x and its corresponding attribute y is unknown. In fact, the latent variable inference is quite useful as it enables model evaluation on novel images. For simplicity, we introduce our inference algorithm based on the vanilla CVAE and the same algorithm can be directly applied to the proposed dis-CVAE and the other generative models such as GANs (*Gauthier, 2015; Denton et al., 2015*). Firstly we notice that the recognition model $q_\phi(z|y, x)$ may not be directly used to infer z . On one hand, as an approximate, we don't know how far it is from the true posterior $p_\theta(z|x, y)$ because the KL divergence between them is thrown away in the variational learning objective; on the other hand, this approximation does not even exist in the models such as GANs. We propose a general approach for posterior inference via optimization in the latent space. Using Bayes' rule, we can formulate the posterior inference by

$$\begin{aligned} \max_z \log p_\theta(z|x, y) &= \max_z [\log p_\theta(x|z, y) + \log p_\theta(z|y)] \\ &= \max_z [\log p_\theta(x|z, y) + \log p_\theta(z)] \end{aligned} \quad (2.8)$$

Note that the generation models or likelihood terms $p_\theta(x|z, y)$ could be non-Gaussian or even a deterministic function (e.g. in GANs) with no proper probabilistic definition. Thus, to make our algorithm general enough, we reformulate the inference in (2.8) as an energy minimization problem,

$$\min_z E(z, x, y) = \min_z [L(\mu(z, y), x) + \lambda R(z)] \quad (2.9)$$

where $L(\cdot, \cdot)$ is the image reconstruction loss and $R(\cdot)$ is a prior regularization term. Taking the simple Gaussian model as an example, the posterior inference can be re-written as,

$$\min_z E(z, x, y) = \min_z [\|\mu(z, y) - x\|^2 + \lambda\|z\|^2] \quad (2.10)$$

Note that we abuse the mean function $\mu(z, y)$ as a general image generation function. Since $\mu(z, y)$ is a complex neural network, optimizing (2.9) is essentially error back-propagation from the energy function to the variable z , which we solve by the ADAM method (*Kingma and Ba, 2015*). Our algorithm actually shares a similar spirit with recently proposed neural network visualization (*Yosinski et al., 2015*) and texture synthesis algorithms (*Gatys et al., 2015*). The difference is that we use generation models for recognition while their algorithms use recognition models for generation. Compared to the conventional way of inferring z from recognition model $q_\phi(z|x, y)$, the proposed optimization contributed to an empirically more accurate latent variable z and hence was useful for reconstruction, completion, and editing.

2.5 Experiments

Datasets. We evaluated our model on two datasets: Labeled Faces in the Wild (LFW) (*Huang et al., 2007*) and Caltech-UCSD Birds-200-2011 (CUB) (*Wah et al., 2011*). For experiments on LFW, we aligned the face images using five landmarks (*Zhu et al., 2014*) and rescaled the center region to 64×64 . We used 73 dimensional attribute score vector provided by (*Kumar et al., 2009*) that describes different aspects of facial appearance such as age, gender, or facial expression. We trained our model using 70% of the data (9,000 out of 13,000 face images) following the training-testing split (View 1) (*Huang et al., 2007*), where the face identities are distinct between train and test sets. For experiments on CUB, we cropped the bird region using the

tight bounding box computed from the foreground mask and rescaled to 64×64 . We used 312 dimensional binary attribute vector that describes bird parts and colors. We trained our model using 50% of the data (6,000 out of 12,000 bird images) following the training-testing split (*Wah et al.*, 2011). For model training, we held-out 10% of training data for validation.

Data preprocessing and augmentation. To make the learning easier, we preprocessed the data by normalizing the pixel values to the range $[-1, 1]$. We augmented the training data with the following image transformations (*Krizhevsky et al.*, 2012; *Eigen et al.*, 2014): 1) flipping images horizontally with probability 0.5, 2) multiplying pixel values of each color channel with a random value $c \in [0.97, 1.03]$, and 3) augmenting the image with its residual with a random tradeoff parameter $s \in [0, 1.5]$. Specifically, for CUB experiments, we performed two extra transformations: 4) rotating images around the centering point by a random angle $\theta_r \in [-0.08, 0.08]$, 5) rescaling images to the scale of 72×72 and performing random cropping of 64×64 regions. Note that these methods are designed to be invariant to the attribute description.

Architecture design. For disCVAE, we build four convolutional neural networks (one for foreground and the other for background for both recognition and generation networks) for auto-encoding style training. The foreground encoder network consists of 5 convolution layers, followed by 2 fully-connected layers (convolution layers have 64, 128, 256, 256 and 1024 channels with filter size of 5×5 , 5×5 , 3×3 , 3×3 and 4×4 , respectively; the two fully-connected layers have 1024 and 192 neurons). The attribute stream is merged with image stream at the end of the recognition network. The foreground decoder network consists of 2 fully-connected layers, followed by 5 convolution layers with 2-by-2 upsampling (fully-connected layers have 256 and $8 \times 8 \times 256$ neurons; the convolution layers have 256, 256, 128, 64 and 3 channels with filter size of 3×3 , 5×5 , 5×5 , 5×5 and 5×5). The foreground prediction stream and

gating prediction stream are separated at the last convolution layer. We adopt the same encoder/decoder architecture for background networks but with fewer number of channels. For all the models, we fixed the latent dimension to be 256 and found this configuration is sufficient to generate 64×64 images in our setting. We adopt slightly different architectures for different datasets: we use 192 dimensions to foreground latent space and 64 dimensions to background latent space for experiments on LFW dataset; we use 128 dimensions for both foreground and background latent spaces on CUB dataset. Compared to vanilla CVAE, the proposed disCVAE has more parameters because of the additional convolutions introduced by the two-stream architecture. However, we found that adding more parameters to vanilla CVAE does not lead to much improvement in terms of image quality. Although both *Dosovitskiy et al.* (2015) and the proposed method use segmentation masks as supervision, naive mask prediction was not comparable to the proposed model in our setting based on the preliminary results. In fact, the proposed disCVAE architecture assigns foreground/background generation to individual networks and composite with gated interaction, which we found very effective in practice.

Implementation details. We used ADAM (*Kingma and Ba*, 2015) for stochastic optimization in all experiments. For training, we used mini-batch of size 32 and the learning rate 0.0003. We also added dropout layer of ratio 0.5 for the image stream of the encoder network before merging with attribute stream. For posterior inference, we used the learning rate 0.3 with 1000 iterations. The models are implemented using deep learning toolbox Torch7 (*Collobert et al.*, 2011).

Baselines. For the vanilla CVAE model, we used the same convolution architecture from foreground encoder network and foreground decoder network. To demonstrate the significance of attribute-conditioned modeling, we trained an unconditional variational auto-encoders with almost the same convolutional architecture as our CVAE.

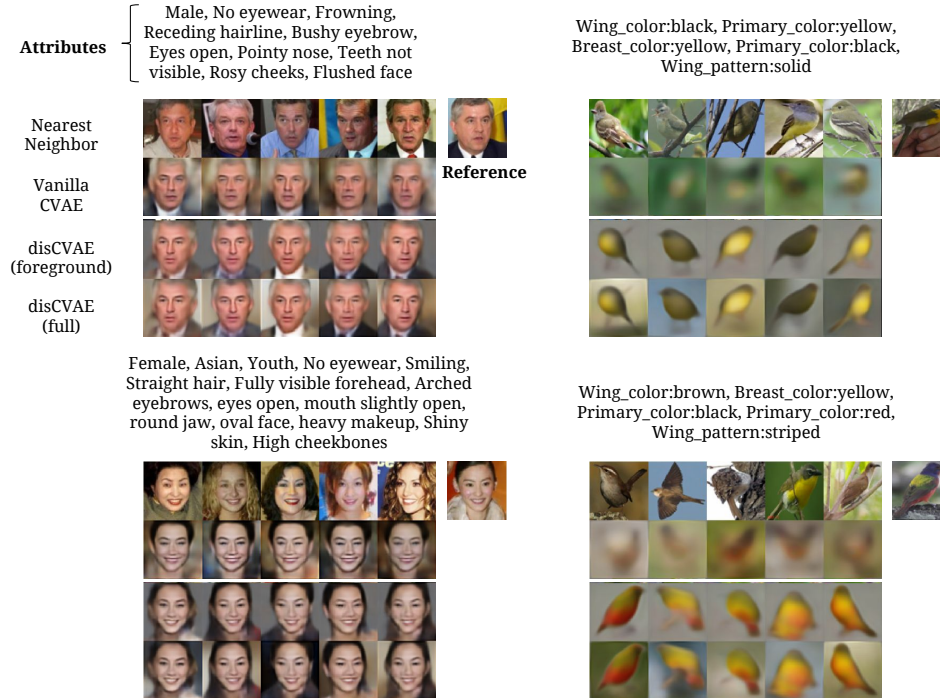


Figure 2.3: Attribute-conditioned image generation.

2.5.1 Attribute-conditioned Image Generation

To examine whether the model has the capacity to generate diverse and realistic images from given attribute description, we performed the task of attribute-conditioned image generation. For each attribute description from testing set, we generated 5 samples by the proposed generation process: $x \sim p_{\theta}(x|y, z)$, where z is sampled from isotropic Gaussian distribution. For vanilla CVAE, x is the only output of the generation. In comparison, for disCVAE, the foreground image x_F can be considered a by-product of the layered generation process. For evaluation, we visualized the samples generated from the model in Figure 2.3 and compared them with the corresponding image in the testing set, which we name as “reference” image. To demonstrate that model did not exploit the trivial solution of attribute-conditioned generation by memorizing the training data, we added a simple baseline as experimental comparison. Basically, for each given attribute description in the testing set, we conducted the nearest neighbor search in the training set. We used

the mean squared error as the distance metric for the nearest neighbor search (in the attribute space). For more visual results and code, please see the project website: <https://sites.google.com/site/attribute2image/>.

Attribute-conditioned face image generation. As we can see in Figure 2.3, face images generated by the proposed models look realistic and non-trivially different from each other, especially for view-point and background color. Moreover, it is clear that images generated by disCVAE have clear boundaries against the background. In comparison, the boundary regions between the hair area and background are quite blurry for samples generated by vanilla CVAE. This observation suggests the limitation of vanilla CVAE in modeling hair pattern for face images. This also justifies the significance of layered modeling and latent space disentangling in our attribute-conditioned generation process. Compared to the nearest neighbors in the training set, the generated samples can better reflect the input attribute description.

Attribute-conditioned bird image generation. Compared to the experiments on LFW database, the bird image modeling is more challenging because the bird images have more diverse shapes and color patterns and the binary-valued attributes are more sparse and higher dimensional. As we can see in Figure 2.3, there is a big difference between two versions of the proposed CVAE model. Basically, the samples generated by vanilla CVAE are blurry and sometimes blended with the background area. However, samples generated by disCVAE have clear bird shapes and reflect the input attribute description well. This confirms the strengths of the proposed layered modeling of images.

Attribute-conditioned Image Progression. To better analyze the proposed model, we generate images with interpolated attributes by gradually increasing or decreasing the values along each attribute dimension. We regard this process as

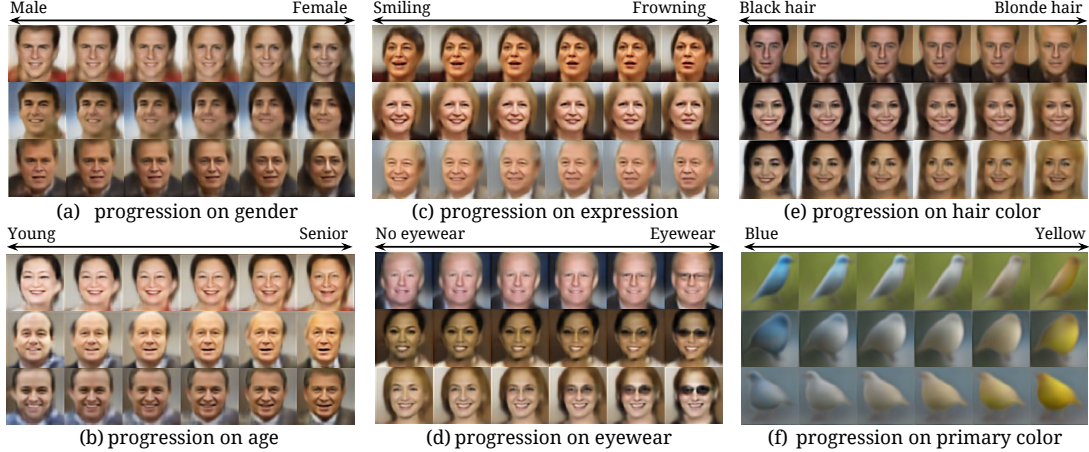


Figure 2.4: Attribute-conditioned image progression. The visualization is organized into six attribute groups (e.g., “gender”, “age”, “facial expression”, “eyewear”, “hair color” and “primary color (blue vs. yellow)”). Within each group, the images are generated from $p_{\theta}(x|y, z)$ with $z \sim \mathcal{N}(0, I)$ and $y = [y_{\alpha}, y_{\text{rest}}]$, where $y_{\alpha} = (1 - \alpha) \cdot y_{\text{min}} + \alpha \cdot y_{\text{max}}$. Here, y_{min} and y_{max} stands for the minimum and maximum attribute value respectively in the dataset along the corresponding dimension.

attribute-conditioned image progression. Specifically, for each attribute vector, we modify the value of one attribute dimension by interpolating between the minimum and maximum attribute value. Then, we generate images by interpolating the value of y between the two attribute vectors while keeping latent variable z fixed. For visualization, we use the attribute vector from testing set.

As we can see in Figure 2.4, samples generated by progression are visually consistent with attribute description. For face images, by changing attributes like “gender” and “age”, the identity-related visual appearance is changed accordingly but the viewpoint, background color, and facial expression are well preserved; on the other hand, by changing attributes like “facial expression”, “eyewear” and “hair color”, the global appearance is well preserved but the difference appears in the local region. For bird images, by changing the primary color from one to the other, the global shape and background color are well preserved. These observations demonstrated that the generation process of our model is well controlled by the input attributes.

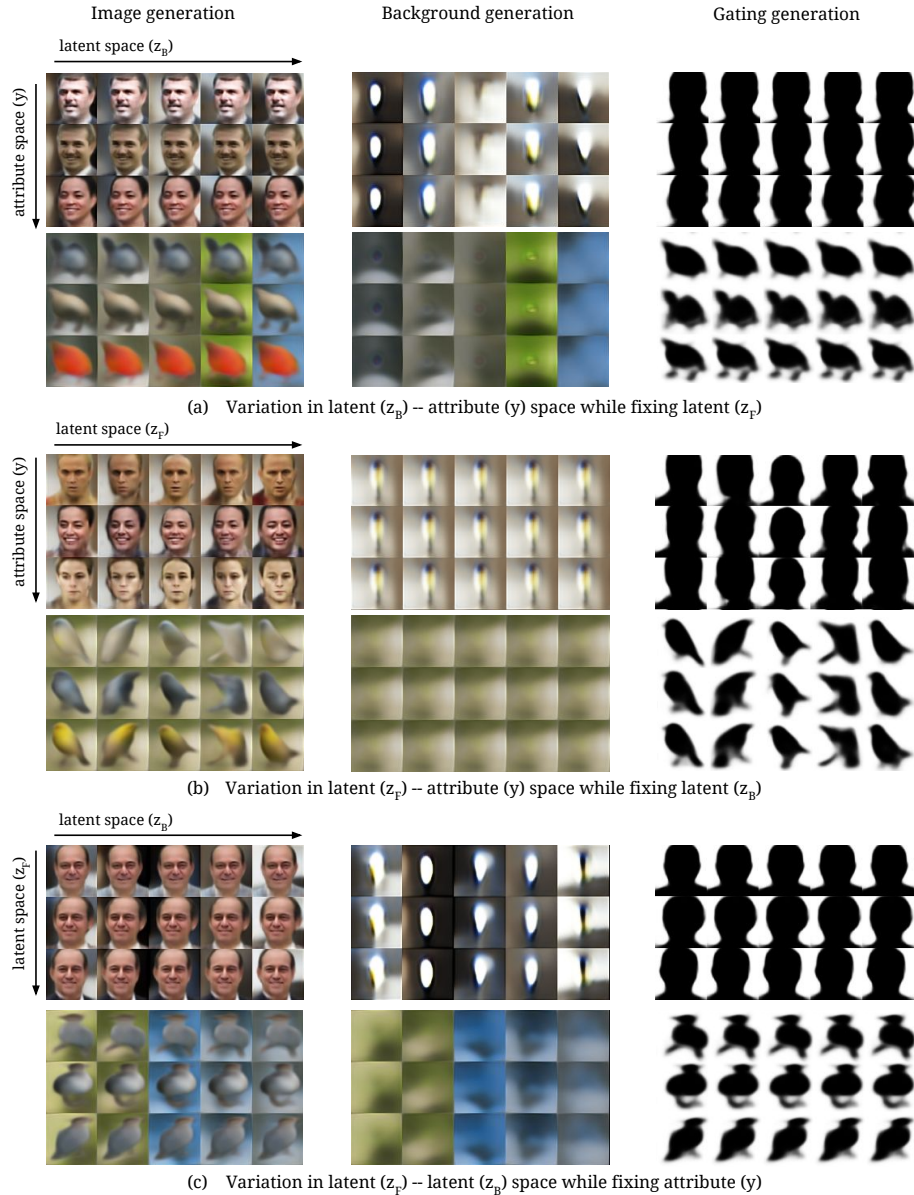


Figure 2.5: Analysis: Latent Space Disentangling.

Analysis: Latent Space Disentangling. To better analyze the disCVAE, we performed the following experiments on the latent space. In this model, the image generation process is driven by three factors: attribute y , foreground latent variable z_F and background latent variable z_B . By changing one variable while fixing the other two, we can analyze how each variable contributes to the final generation results. We visualize the samples x , the generated background x_B and the gating

variables g in Figure 2.5. We summarized the observations as follows: 1) The background of the generated samples look different but with identical foreground region when we change background latent variable z_B only; 2) the foreground region of the generated samples look diverse in terms of viewpoints but still look similar in terms of appearance and the samples have uniform background pattern when we change foreground latent variable z_F only. Interestingly, for face images, one can identify a “hole” in the background generation. This can be considered as the location prior of the face images, since the images are relatively aligned. Meanwhile, the generated background for birds are relatively uniform, which demonstrates our model learned to recover missing background in the training set and also suggests that foreground and background have been disentangled in the latent space.

2.5.2 Attribute-conditioned Image Reconstruction and Completion

Image reconstruction. Given a test image x and its attribute vector y , we find z that maximizes the posterior $p_\theta(z|x, y)$ following Equation (2.9).

Image completion. Given a test image with synthetic occlusion, we evaluate whether the model has the capacity to fill in the occluded region by recognizing the observed region. We denote the occluded (unobserved) region and observed region as x_u and x_o , respectively. For completion, we first find z that maximizes the posterior $p_\theta(z|x_o, y)$ by optimization (2.9). Then, we fill in the unobserved region x_u by generation using $p_\theta(x_u|z, y)$. For each face image, we consider four types of occlusions: occlusion on the eye region, occlusion on the mouth region, occlusion on the face region and occlusion on right half of the image. For occluded regions, we set the pixel value to 0. For each bird image, we consider blocks of occlusion of size 8×8 and 16×16 at random locations.

In Figure 2.6, we visualize the results of image reconstruction (a,b) and image

completion (c-h). As we can see, for face images, our proposed CVAE models are in general good at reconstructing and predicting the occluded region in unseen images (from testing set). However, for bird images, vanilla CVAE model had significant failures in general. This agreed with the previous results in attribute-conditioned image generation.

In addition, to demonstrate the significance of attribute-conditioned modeling, we compared our vanilla CVAE and disCVAE with unconditional VAE (attribute is not given) for image reconstruction and completion. It can be seen in Fig. 2.6(c)(d), the generated images using attributes actually perform better in terms of expression and eyewear (“smiling” and “sunglasses”).

For quantitative comparisons, we measured the pixel-level mean squared error on the entire image and occluded region for reconstruction and completion, respectively. We summarized the results in Table 2.1 (mean squared error and standard error). The quantitative analysis highlighted the benefits of attribute-conditioned modeling and the importance of layered modeling.

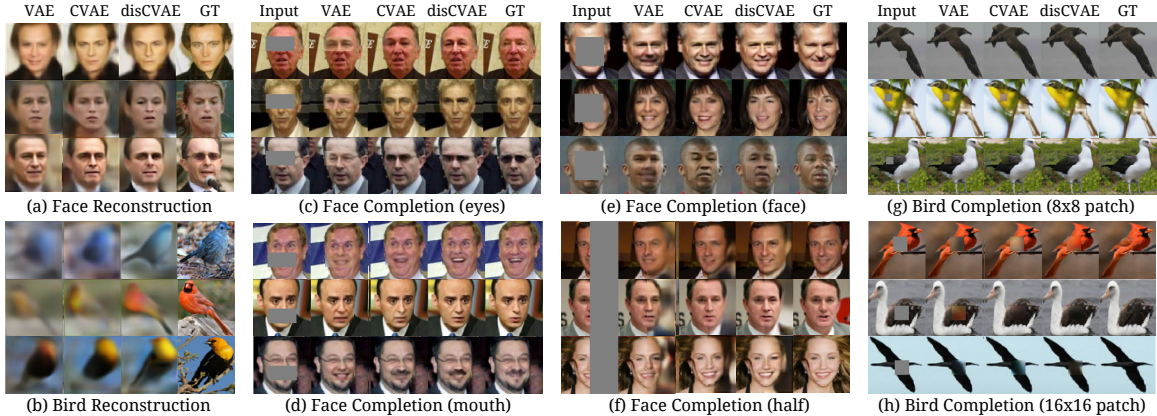


Figure 2.6: Attribute-conditioned image reconstruction and completion.

Table 2.1: Quantitative comparisons on face reconstruction and completion tasks.

Face	Recon: full	Recon: fg	Comp: eye	Comp: mouth	Comp: face	Comp: half
VAE	11.8 ± 0.1	9.4 ± 0.1	13.0 ± 0.1	12.1 ± 0.1	13.1 ± 0.1	21.3 ± 0.2
CVAE	11.8 ± 0.1	9.3 ± 0.1	12.0 ± 0.1	12.0 ± 0.1	12.3 ± 0.1	20.3 ± 0.2
disCVAE	10.0 ± 0.1	7.9 ± 0.1	10.3 ± 0.1	10.3 ± 0.1	10.9 ± 0.1	18.8 ± 0.2
Bird	Recon: full	Recon: fg	Comp: 8×8	Comp: 16×16		
VAE	14.5 ± 0.1	11.7 ± 0.1	1.8 ± 0.1	4.6 ± 0.1		
CVAE	14.3 ± 0.1	11.5 ± 0.1	1.8 ± 0.1	4.4 ± 0.1		
disCVAE	12.9 ± 0.1	10.2 ± 0.1	1.8 ± 0.1	4.4 ± 0.1		

2.6 Discussions

To conclude, this chapter studied a novel problem of attribute-conditioned image generation and proposed a solution with CVAEs. Considering the compositional structure of images, we proposed a novel disentangling CVAE (disCVAE) with a layered representation. Results on faces and birds demonstrate that our models can generate realistic samples with diverse appearance and especially disCVAE significantly improved the generation quality on bird images. To evaluate the learned generation models on the novel images, we also developed an optimization-based approach to posterior inference and applied it to the tasks of image reconstruction and completion with quantitative evaluation.

CHAPTER III

Learning Controllable and Structured Representations for Semantic Image Manipulation

3.1 Introduction

Learning to perceive, reason and manipulate images has been one of the core research problems in computer vision, machine learning and graphics for decades *Hoiem et al.* (2005, 2008); *Gupta et al.* (2010); *Barnes et al.* (2009). Recently the problem has been actively studied in interactive image editing using deep neural networks, where the goal is to manipulate an image according to the various types of user-controls, such as color strokes *Sangkloy et al.* (2017); *Zhu et al.* (2016a), key-points *Zhu et al.* (2016a); *Reed et al.* (2016), textures *Xian et al.* (2018), and holes-to-fill (inpainting) *Pathak et al.* (2016). While these interactive image editing approaches have made good advances in synthesizing high-quality manipulation results, they are limited to direct manipulation on natural image manifold.

The main focus of this chapter is to achieve semantic-level manipulation of images. Instead of manipulating images on natural image manifold, we consider semantic label map as an interface for manipulation. By editing the label map, users are able to specify the desired images at semantic-level, such as the location, object class, and object shape. Recently, approaches based on image-to-image translation *Isola et al.*

(2017); *Chen and Koltun (2017)*; *Wang et al. (2017)* have demonstrated promising results on semantic image manipulation. However, the existing works mostly focused on learning a *style* transformation function from label maps to pixels, while manipulation of *structure* of the labels remains fully responsible to users. The requirement on direct control over pixel-wise labels makes the manipulation task still challenging since it requires a precise and considerable amount of user inputs to specify the structure of the objects and scene. Although the problem can be partly addressed by template-based manipulation interface (e.g., adding the objects from the pre-defined sets of template masks *Wang et al. (2017)*), blind pasting of the object mask is problematic since the structure of the object should be determined adaptively depending on the surrounding context.

In this work, we tackle the task of semantic image manipulation as a hierarchical generative process. We start our image manipulation task from a coarse-level abstraction of the scene: a set of *semantic* bounding boxes which provide both semantic (what) and spatial (where) information of the scene in an object-level. Such representation is natural and flexible that enables users to manipulate the scene layout by adding, removing, and moving each bounding box. To facilitate the image manipulation from coarse-level semantic bounding boxes, we introduce a hierarchical generation model that predicts the image in multiple abstraction levels. Our model consists of two parts: layout and image generators. Specifically, our structure generator first infers the fine-grained semantic label maps from the coarse object bounding boxes, which produces structure (shape) of the manipulated object aligned with the context. Given the predicted layout, our image generator infers the style (color and textures) of the object considering the perceptual consistency to the surroundings. This way, when adding, removing, and moving semantic bounding boxes, our model can generate an appropriate image seamlessly integrated into the surrounding image.

We present two applications of the proposed method on interactive and data-

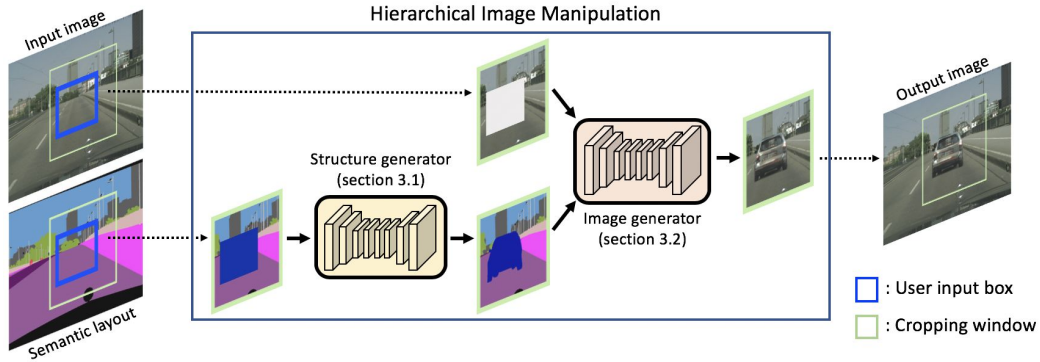


Figure 3.1: Overall pipeline of the proposed semantic manipulation framework.

driven image editing. In the experiment on interactive image editing, we show that our method allows users to easily manipulate images using object bounding boxes, while the structure and style of the generated objects are adaptively determined depending on the context and naturally blended into the scene. Also, we show that our simple object-level manipulation interface can be naturally extended to data-driven image manipulation, by automatically sampling the object boxes and generating the novel images.

The benefits of the hierarchical image manipulation are three-fold. First, it supports richer manipulation tasks such as adding, moving or removing objects through object-level control while the fine-grained object structures are inferred by the model. Second, when conditioned on coarse and fine-grained semantic representations, the proposed model produces better image manipulation results compared to models learned without structural control. Finally, we demonstrate the effectiveness of the proposed idea on interactive and automatic image manipulation.

3.2 Related Work

Deep visual manipulation. Visual manipulation is a task of synthesizing the new image by manipulating parts of a reference image. Thanks to the emergence of generative adversarial networks (GANs) *Goodfellow et al.* (2014) and perceptual fea-

tures discovered from deep convolutional neural networks *Krizhevsky et al. (2012)*; *Simonyan and Zisserman (2014)*; *Szegedy et al. (2015)*, neural image manipulation *Zhu et al. (2016a)*; *Sangkloy et al. (2017)*; *Li et al. (2017)*; *Yeh et al. (2017)*; *Pathak et al. (2016)*; *Denton et al. (2016)* has gained increasing popularity in recent years. *Zhu et al. (2016a)* presented a novel image editing framework with a direct constraint capturing real image statistics using deep convolutional generative adversarial networks *Radford et al. (2015)*. *Li et al. (2017)* and *Yeh et al. (2017)* investigated semantic face image editing and completion using convolution encoder-decoder architecture, jointly trained with a pixel-wise reconstruction constraint, a perceptual (adversarial) constraint, and a semantic structure constraint. In addition, *Pathak et al. (2016)* and *Denton et al. (2016)* studied context-conditioned image generation that performs pixel-level hole-filling given the surrounding regions using deep neural networks, trained with adversarial and reconstruction loss. Contrary to the previous works that manipulate images based on low-level visual controls such as visual context *Denton et al. (2016)*; *Pathak et al. (2016)*; *Yeh et al. (2017)*; *Li et al. (2017)* or color strokes *Zhu et al. (2016a)*; *Sangkloy et al. (2017)*, our model allows semantic control over manipulation process through labeled bounding box and the inferred semantic layout.

Structure-conditional image generation. Starting from the pixel-wise semantic structure, recent breakthroughs approached the structure-conditional image generation through direct image-to-image translation *Liu et al. (2017)*; *Isola et al. (2017)*; *Zhu et al. (2017a)*; *Chen and Koltun (2017)*. *Isola et al. (2017)* employed a convolutional encoder-decoder network with conditional adversarial objective to learn label-to-pixel mapping. Later approaches improved the generation quality by incorporating perceptual loss from a pre-trained classifier *Chen and Koltun (2017)* or feature-matching loss from multi-scale discriminators *Wang et al. (2017)*. In particu-

lar, *Wang et al.* (2017) demonstrated a high-resolution image synthesis results and its application to semantic manipulation by controlling the pixel-wise labels. Contrary to the previous works focusing on learning a direct mapping from label to image, our model learns hierarchical mapping from coarse bounding box to image by inferring intermediate label maps. Our work is closely related to *Hong et al.* (2018), which generates an image from a text description through multiple levels of abstraction consisting of bounding boxes, semantic layouts, and finally pixels. Contrary to *Hong et al.* (2018), however, our work focuses on manipulation of parts of an image, which requires incorporation of both semantic and visual context of surrounding regions in the hierarchical generation process in such a way that structure and style of the generated object are aligned with the other parts of an image.

3.3 Hierarchical Image Manipulation

Given an input image $I^{in} \in \mathbb{R}^{H \times W \times 3}$, our goal is to synthesize the new image I^{out} by manipulating its underlying semantic structure. Let $M^{in} \in \mathbb{R}^{H \times W \times C}$ denotes a semantic label map of the image defined over C categories, which is either given as ground-truth (in training time) or inferred by the pre-trained visual recognition model (in testing time). Then our goal is to synthesize the new image by manipulating M^{in} , which allows the semantically-guided manipulation of an image.

The key idea of this chapter is to introduce an object bounding box B as an abstracted interface to manipulate the semantic label map. Specifically, we define a controllable bounding box $B = \{\mathbf{b}, c\}$ as a combination of box corners $\mathbf{b} \in \mathbb{R}^4$ and a class label $c = \{0, \dots, C\}$, which represents the location, size and category of an object. By adding the new box or modifying the parameters of existing boxes, a user can manipulate the image through adding, moving or deleting the objects¹. Given an object-level specification by B , the image manipulation is then posed as a hierarchical

¹We used $c = 0$ to indicate a deletion operation, where the model fills the labels with surroundings.

generation task from a coarse bounding box to pixel-level predictions of structure and style.

Figure 3.1 illustrates the overall pipeline of the proposed algorithm. When the new bounding box B is given, our algorithm first extracts the local observations of label map $M \in \mathbb{R}^{S \times S \times C}$ and image $I \in \mathbb{R}^{S \times S \times 3}$ by cropping squared windows of size $S \times S$ centered around B . Then conditioned on M , I and B , our model generates the manipulated image by the following procedures:

- Given a bounding box B and the semantic label map M , the structure generator predicts the manipulated semantic label map by $\hat{M} = G^M(M, B)$ (Section 3.3.1)
- Given the manipulated label map \hat{M} and image I , the image generator predicts the manipulated image \hat{I} by $\hat{I} = G^I(\hat{M}, I)$ (Section 3.3.2)

After generating the manipulated image patch \hat{I} , we place it back to the original image to finish the manipulation task. The manipulation of multiple objects is performed iteratively by applying the above procedures for each box. In the following, we explain the manipulation pipeline on a single box B .

3.3.1 Structure generator

The goal of the structure generator is to infer the latent structure of the region specified by $B = \{\mathbf{b}, c\}$ in the form of pixel-wise class labels $\hat{M} \in \mathbb{R}^{S \times S \times C}$. The outputs of the structure generator should reflect the class-specific structure of the object defined by B as well as interactions of the generated object with the surrounding context (e.g., a person riding a motorcycle). To consider both conditions in the generation process, the structure generator incorporates the label map M and the bounding box B as inputs and performs a conditional generation by $\hat{M} = G^M(M, B)$.

Figure 3.2 illustrates the overall architecture of the structure generator. The structure generator takes in the masked layout $\bar{M} \in \mathbb{R}^{S \times S \times C}$ and the binary mask

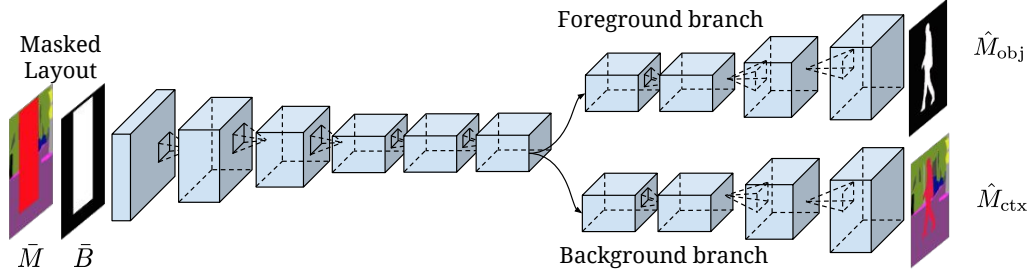


Figure 3.2: Architecture of the structure generator. Given a masked layout \bar{M} and a binary mask \bar{B} encoding the class and location of the object, respectively, the model produces the manipulated layout \hat{M} by the outputs from the two-stream decoder corresponding to the binary mask of object and semantic label map of entire region inside the box.

$\bar{B} \in \mathbb{R}^{S \times S \times 1}$, where $\bar{M}_{ijc} = 1$ and $\bar{B}_{ij} = 1$ for all pixels (i, j) inside the box for class c . Given these inputs, the model predicts the manipulated outcome using two decoding pathways.

Our design principle is motivated by generative layered image modeling *Yang et al. (2012b); Yan et al. (2016a); Vondrick et al. (2016); Reed et al. (2016); Yang et al. (2017)*, which generates foreground (i.e., object) and background (i.e., context) using separate output streams. In our model, the *foreground* output stream produces the predictions on binary object mask $\hat{M}_{\text{obj}} \in \mathbb{R}^{S \times S \times 1}$, which defines the object shape tightly bounded by object box B . The *background* output stream produces per-pixel label maps $\hat{M}_{\text{ctx}} \in \mathbb{R}^{S \times S \times C}$ inside B .

The objective for our structure generator is then given by

$$\mathcal{L}_{\text{layout}} = \mathcal{L}_{\text{adv}}(\hat{M}_{\text{obj}}, M_{\text{obj}}^*) + \lambda_{\text{obj}} \mathcal{L}_{\text{recon}}(\hat{M}_{\text{obj}}, M_{\text{obj}}^*) + \lambda_{\text{ctx}} \mathcal{L}_{\text{recon}}(\hat{M}_{\text{ctx}}, \bar{M}), \quad (3.1)$$

where M_{obj}^* is the ground-truth binary object mask on B and $\mathcal{L}_{\text{recon}}(\cdot, \cdot)$ is the reconstruction loss using cross-entropy. $\mathcal{L}_{\text{adv}}(\hat{M}_{\text{obj}}, M_{\text{obj}}^*)$ is the conditional adversarial loss defined on object mask ensuring the perceptual quality of predicted object shape,

which is given by

$$\mathcal{L}_{\text{adv}}(\hat{M}_{\text{obj}}, M_{\text{obj}}^*) = \mathbb{E}_{M_{\text{obj}}^*} [\log(D^M(M_{\text{obj}}^*, \bar{M}))] + \mathbb{E}_{\hat{M}_{\text{obj}}} [1 - \log(D^M(\hat{M}_{\text{obj}}, \bar{M}))], \quad (3.2)$$

where $D^M(\cdot, \cdot)$ is a conditional discriminator.

During inference, we construct the manipulated layout \hat{M} using \hat{M}_{obj} and \hat{M}_{ctx} . Contrary to the prior works on layered image model, we selectively choose outputs from either foreground or background streams depending on the manipulation operation defined by B . The output \hat{M} is given by

$$\hat{M} = \begin{cases} \hat{M}_{\text{ctx}} & \text{if } c = 0 \text{ (deletion)} \\ (\hat{M}_{\text{obj}} \mathbb{1}_c) + (\mathbf{1} - (\hat{M}_{\text{obj}} \mathbb{1}_c)) \odot M & \text{otherwise (addition)} \end{cases}, \quad (3.3)$$

where c is the class label of the bounding box B and $\mathbb{1}_c \in \{0, 1\}^{1 \times C}$ is the one-hot encoded vector of the class c .

3.3.2 Image generator

Given an image I and the manipulated layout \hat{M} obtained by the structure generator, the image generator outputs a pixel-level prediction of the contents inside the regions defined by B . To make the prediction being semantically meaningful and perceptually plausible, the output from the image generator should reflect the semantic structure defined by the layout while being coherent in its style (e.g., color and texture) with the surrounding image. We design the conditional image generator G^I such that $\hat{I} = G^I(I, \hat{M})$, where $I, \hat{I} \in \mathbb{R}^{S \times S \times 3}$ represent the local image before and after manipulation with respect to bounding box B .

Figure 3.3 illustrates the overall architecture of the image generator. The model takes the masked image \bar{I} whose pixels inside the box are filled with 0 and the manipulated layout \hat{M} as inputs, and produces the manipulated image \hat{I} as output. We design

the image generator G^I to have a two-stream convolutional encoder and a single-stream convolutional decoder connected by an intermediate feature map F . As we see in the figure, the convolutional encoder has two separate downsampling streams, which we referred to as image encoder $f_{\text{image}}(\bar{I})$ and layout encoder $f_{\text{layout}}(\hat{M})$, respectively. The intermediate feature F is obtained by an element-wise feature interaction layer gated by the binary mask B_F on object location.

$$F = f_{\text{layout}}(\hat{M}) \odot B_F + f_{\text{image}}(\bar{I}) \odot (\mathbf{1} - B_F). \quad (3.4)$$

Finally, the convolutional decoder $g_{\text{image}}(\cdot)$ takes the fused feature F as input and produces the manipulated image through $\hat{I} = g_{\text{image}}(F)$. Note that we use the ground-truth layout M during model training but the predicted layout \hat{M} is used at the inference time during testing.

We define the following loss for the image generation task.

$$\mathcal{L}_{\text{image}} = \mathcal{L}_{\text{adv}}(\hat{I}, I) + \lambda_{\text{feature}} \mathcal{L}_{\text{feature}}(\hat{I}, I). \quad (3.5)$$

The first term $\mathcal{L}_{\text{adv}}(\hat{I}, I)$ is the conditional adversarial loss defined by

$$\mathcal{L}_{\text{adv}}(\hat{I}, I) = \mathbb{E}_I \left[\log(D^I(I, \hat{M})) \right] + \mathbb{E}_{\hat{I}} \left[1 - \log(D^I(\hat{I}, \hat{M})) \right]. \quad (3.6)$$

The second term $\mathcal{L}_{\text{feature}}(\hat{I}, I)$ is the feature matching loss *Wang et al.* (2017). Specifically, we compute the distance between the real and manipulated images using the intermediate features from the discriminator by

$$\mathcal{L}_{\text{feature}}(\hat{I}, I) = \mathbb{E}_{I, \hat{I}} \sum_{i=1} \|D^{(i)}(I, \hat{M}) - D^{(i)}(\hat{I}, \hat{M})\|_F^2, \quad (3.7)$$

where $D^{(i)}$ is the outputs of i^{th} layer in discriminator, $\|\cdot\|_F$ is the Frobenius norm.

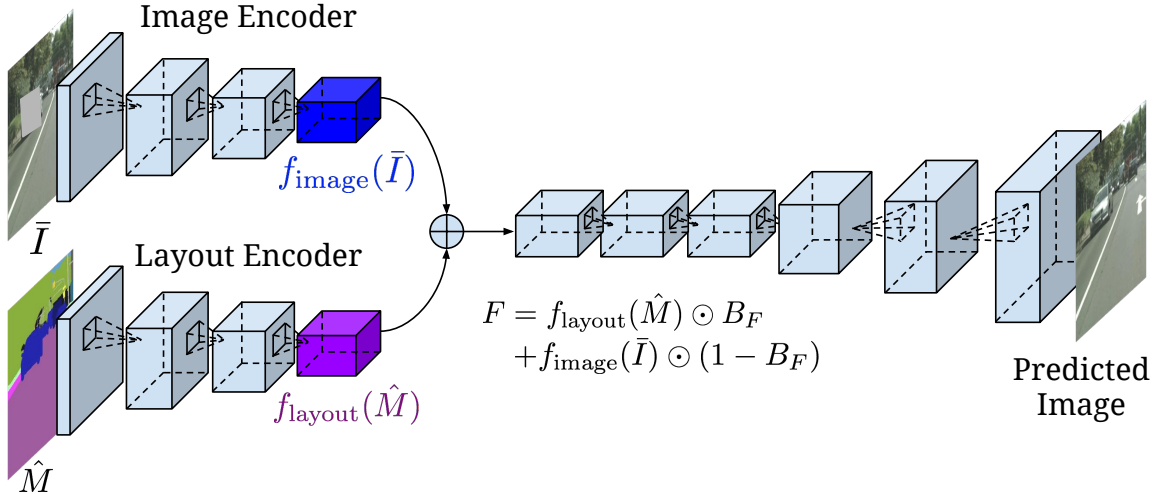


Figure 3.3: Architecture of the image generator. Given a masked image \bar{I} and the semantic layout \hat{M} , the model encodes visual style and semantic structure of the object using separate encoding pathways and produces the manipulated image.

Discussions. The proposed model encodes both image and layout using two-stream encoding pathways. With only image encoder, it simply performs image in-painting *Pathak et al. (2016)*, which attempts to fill the hole with patterns coherent with the surrounding region. On the other hand, our model with only layout encoder becomes similar to image-to-image translation models *Isola et al. (2017)*; *Chen and Koltun (2017)*; *Wang et al. (2017)*, which translates the pixel-wise semantic label maps to the RGB pixel values. Intuitively, by combining information from both encoders, the model learns to manipulate images that reflect the underlying image structure defined by the label map with appearance patterns naturally blending into the surrounding context, which is semantically meaningful and perceptually plausible.

3.4 Experiments

3.4.1 Implementation Details

Datasets. We conduct both quantitative and qualitative evaluations on the Cityscape dataset *Cordts et al. (2016)*, a semantic understanding benchmark of European urban

street scenes containing 5,000 high-resolution images with fine-grained annotations including instance-wise map and semantic map from 30 categories. Among 30 semantic categories, we treat 10 of them including `person`, `rider`, `car`, `truck`, `bus`, `caravan`, `trailer`, `train`, `motorcycle`, `bicycle` as our foreground object classes while leaving the rest as background classes for image editing purpose. For evaluation, we measure the generation performance on each of the foreground object bounding box in 500 validation images. To further demonstrate the image manipulation results on more complex scene, we also conduct qualitative experiments on bedroom images from ADE20K dataset *Zhou et al. (2017a)*. Among 49 object categories frequently appearing in a bedroom, we select 31 movable ones as foreground objects for manipulation.

Training. As one challenge, collecting ground-truth examples before and after image manipulation is expensive and time-consuming. Instead, we simulate the addition ($c \in \{1, \dots, C\}$) and deletion ($c = 0$) operations by sampling boxes from the object and random image regions, respectively. For training, we employ an Adam optimizer *Kingma and Ba (2015)* with learning rate 0.0002, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and linearly decrease the learning rate after the first 100-epochs for training. The hyperparameters λ_{obj} , λ_{ctx} , λ_{feature} are set to 10. Our `PyTorch` implementation will be open-sourced.

Evaluation metrics. We employ three metrics to measure the perceptual and conditional generation quality. We use Structural Similarity Index (SSIM) *Wang et al. (2004)* to evaluate the similarity of the ground-truth and predicted images based on low-level visual statistics. To measure the quality of layout-conditional image generation, we apply a pre-trained semantic segmentation model DeepLab v3 *Chen et al. (2018)* to the generated images, and measure the consistency between the input layout and the predicted segmentation labels in terms of pixel-wise accuracy (layout \rightarrow image \rightarrow layout). Finally, we conduct user study using Mechanical Turk

	Layout	SSIM	Segmentation (%)	Human eval. (%)
SingleStream-Image	-	0.285	59.6	12.3
SingleStream-Layout	GT	0.291	71.5	9.2
SingleStream-Concat	GT	0.331	76.7	23.2
TwoStream	GT	0.336	78.5	33.6
TwoStream-Pred	Predicted	0.299	77.9	20.7

Figure 3.4: Comparisons between variants of the proposed method. The last two rows (**TwoStream** and **TwoStream-Pred**) correspond to our full model using ground-truth and predicted layout, respectively.

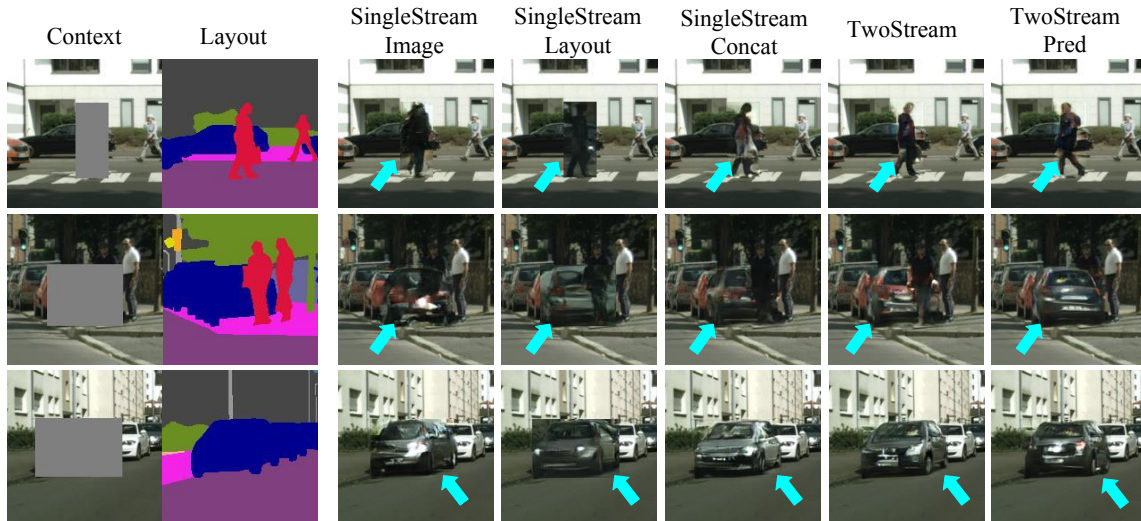


Figure 3.5: Qualitative comparisons to the baselines in Table 3.4. The first two columns show the masked image and ground-truth layout used as input to the models (except **TwoStream-Pred**). The manipulated objects are indicated by blue arrows. Best viewed in color.

(AMT) to evaluate the perceptual generation quality. We present the manipulation results of different methods together with the input image and the class label of the bounding box and ask users to choose the best method based on how natural the manipulated images are. We collect the results for 1,000 examples, each of which is evaluated by 5 different Turkers. We report the performance of each method based on the ratio that method ranked as the best in AMT.

3.4.2 Quantitative Evaluation

Ablation study. To analyze the effectiveness of the proposed method, we conduct an ablation study on several variants of our method. We first consider three baselines of our image generator: conditioned only on image context (**SingleStream-Image**), conditioned only on semantic layout (**SingleStream-Layout**), or conditioned on both by concatenation but using a single encoding pathway (**SingleStream-Concat**). We also compare our full model using a ground-truth and the predicted layouts (**TwoStream** and **TwoStream-Pred**). Table 3.4 summarize the results.

As shown in Table 3.4, conditioning the generation with only image or layout leads to either poor class-conditional generation (**SingleStream-Image**) or less perceptually plausible results (**SingleStream-Layout**). It is because the former misses the semantic layout encoding critical information on which object to generate, while the later misses color and textures of the image that makes the generation results visually consistent with its surroundings. Combining both (**SingleStream-Concat**), the generation quality improves in all metrics, which shows complementary benefits of both conditions in image manipulation. In addition, a comparison between **SingleStream-Concat** and **TwoStream** shows that modeling the image and layout conditions using separate encoding pathways further improves the generation quality. Finally, replacing the layout condition from the ground-truth (**TwoStream**) to the predicted one (**TwoStream-Pred**) leads to a small degree of degradation in perceptual quality partly due to the prediction errors in layout generation. However, clear improvement of **TwoStream-Pred** over **SingleStream-Image** shows the effectiveness of layout prediction in image generation.

Figure 3.5 shows the qualitative comparisons of the baselines. Among all variants, our **TwoStream** model tends to exhibit most recognizable and coherent appearance with the surrounding environment. Interestingly, our model with predicted layout **TwoStream-Pred** generates objects different from the ground-truth layout but still

	Layout	SSIM	Segmentation (%)	Human eval. (%)
Context Encoder	-	0.265	26.4	1.1
Context Encoder++	GT	0.269	35.7	1.4
Pix2PixHD	GT	0.288	79.6	18.0
TwoStream	GT	0.336	78.5	79.5

Table 3.1: Quantitative comparison to other methods. Context Encoder and Pix2PixHD refer to the previous work (*Pathak et al., 2016*) and (*Wang et al., 2017*), respectively.

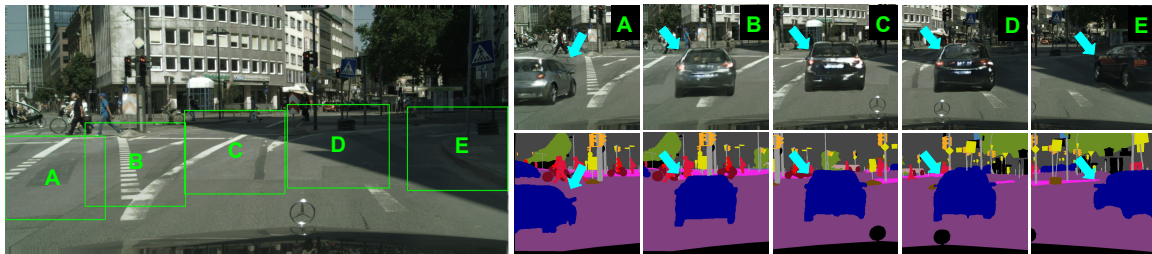


Figure 3.6: Generation results on various locations in an image.

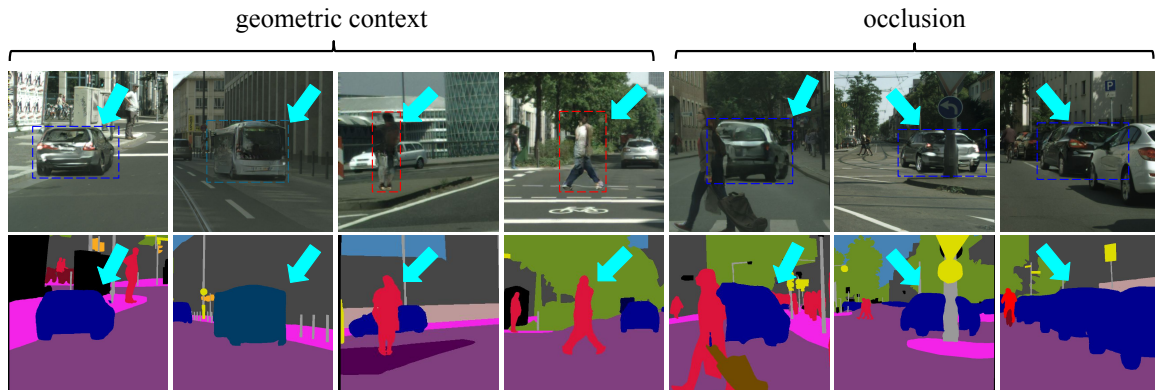


Figure 3.7: Generation results in diverse contexts.

match the bounding box condition. such as a person walking in the different direction (the first row) and objects placed in different order (the second row).

Comparison to other methods. We also compare against a few existing work on context hole-filing and structure-conditioned image generation. First, we consider the recent work on high-resolution pixel-to-pixel translation *Wang et al. (2017)* (referred as Pix2PixHD in Table 3.1). Compared to our `SingleStream-Layout` model, Pix2PixHD model generates full-sized image from semantic layout in one shot. Second, we consider the work for context-driven image in-painting *Pathak et al. (2016)* (re-

ferred as `ContextEncoder`) as another baseline. Similar to our `SingleStream-Image`, `ContextEncoder` does not have access to the semantic layout during training and testing. For fair comparisons, we extended `ContextEncoder` so that it takes segmentation layout as additional input. We refer this model as `ContextEncoder++` in Table 3.1. As we see in the Table 3.1, our two-stream model still achieves the best SSIM and competitive segmentation pixel-wise accuracy. The segmentation accuracy of `Pix2PixHD` is higher than ours since it generates higher resolution images and object textures non-relevant to the input image, but our method still achieves perceptually plausible results consistent with the surrounding image. Note that the motivations of `Pix2PixHD` and our model are different, as `Pix2PixHD` performs image generation from scratch while our focus is local image manipulation.

3.4.3 Qualitative Analysis

Semantic object manipulation. To demonstrate how our hierarchical model manipulates structure and style of an object depending on the context, we conduct qualitative analysis in various settings. In Figure 3.6, we present the manipulation results by moving the same bounding box of a car to different locations in an image. As we see in the figure, our model generates a car with diverse but reasonably-looking shape and appearance when we move its bounding box from one side of the road to another. Interestingly, the shape, orientation, and appearance of the car also change according to the scene layout and shadow in the surrounding regions. Figure 3.7 illustrates generation results in more diverse contexts. It shows that our model generates appropriate structure and appearance of the object considering the context (e.g., occlusion with other objects, interaction with the scene, etc). In addition to generating object matching the surroundings, we can also easily extend our framework to allow users to directly control object style, which we demonstrate below.

Extension to style manipulation Although we focused on addition and deletion of objects as manipulation tasks, we can easily extend our framework to add control over object styles by conditioning the image generator with additional style vector \mathbf{s} by $G^I(\hat{M}, I, \mathbf{s})$. To demonstrate this capability, we define the style vector as a mean color of the object, and synthesized objects while changing the input color (Figure 3.8). The results show that our model successfully synthesizes various objects with the specified color while retaining other parts of the images unchanged. Modeling more complicated styles (e.g., texture) can be achieved by learning a style-encoder $\mathbf{s} = E(X)$ where X is an object template that user can select, but we leave it as a future work.



Figure 3.8: Controlling object color with style vector. Colors used for manipulation are presented at left-upper corners.

Interactive and data-driven image editing. As one of the key applications, we perform interactive image manipulation by adding, removing, and moving object bounding boxes. Figure 3.9 illustrates the results. It shows that our method generates reasonable semantic layouts and images that smoothly augment content of the original image. In addition to interactive manipulation, we can also automate the manipulation process by sampling bounding boxes in an image in a data-driven manner. To demonstrate this idea, we present an application of data-driven image manipulation in Figure 3.10. In this demo, we implement box sampling using a simple non-parametric approach; Given a query image, we first compute its nearest neighbor from the training set based on low-level similarity. Then we compute the geometric

transformation between a query and a training image using SIFT Flow *Liu et al.* (2016), and move object bounding boxes from one scene to another based on the scene-level transformation. As shown in the figure, the proposed methods reasonably sample boxes from appropriate locations. However, directly placing the object (or its mask) may lead to unnatural manipulation due to mismatches in scene configuration (e.g., occlusion and orientation). Our hierarchical model generates objects adaptive to the new context.

Results on indoor scene dataset. In addition to Cityscape dataset, we conduct qualitative experiments on bedroom images using ADE20K datasets *Zhou et al.* (2017a). Figure 3.11 illustrates the interactive image manipulation results. Since objects in the indoor images involve much more diverse categories and appearances, generating appropriate object shapes and textures aligned with other components in a scene is much more challenging than the street images. We observe that the generated objects by the proposed method usually are looking consistent the surrounding context.

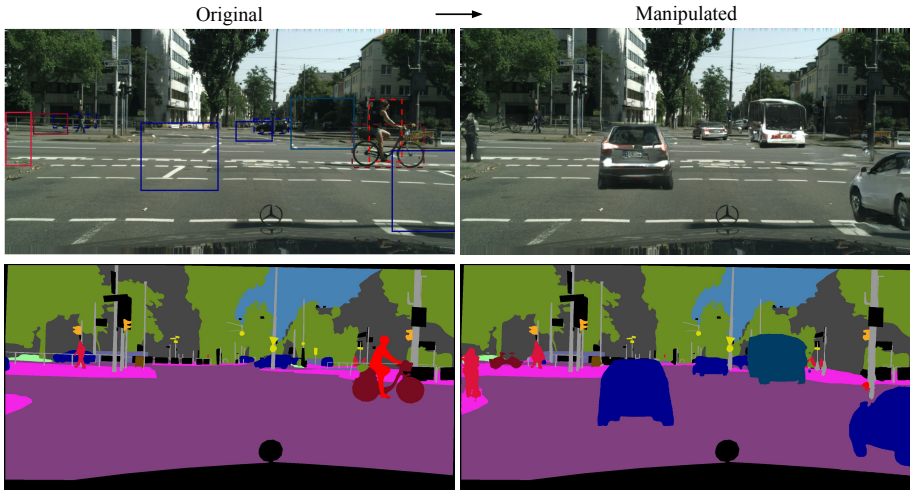


Figure 3.9: Examples of manipulation of multiple objects in images. The line style indicates manipulation operation (solid: addition, dotted: deletion) and the color indicates the object class.

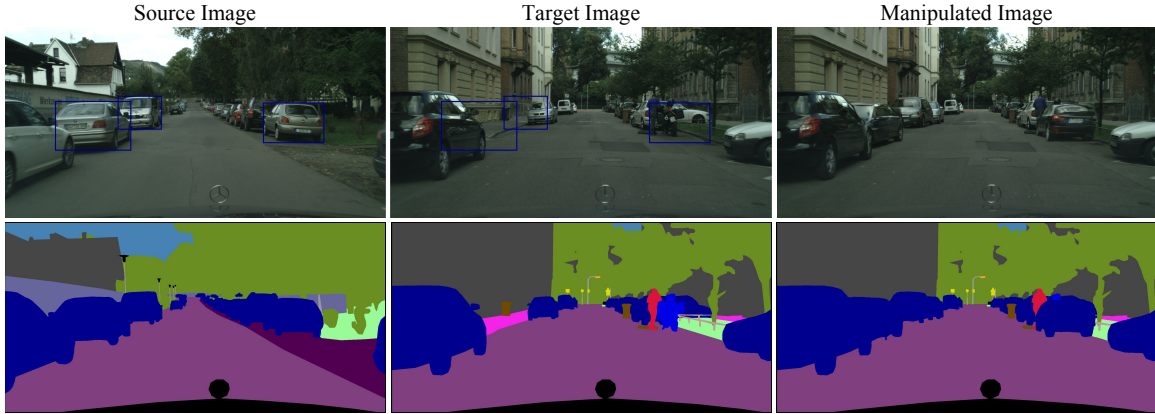


Figure 3.10: Example of data-driven image manipulation. We manipulate the target image by transferring bounding boxes from source image (blue boxes).

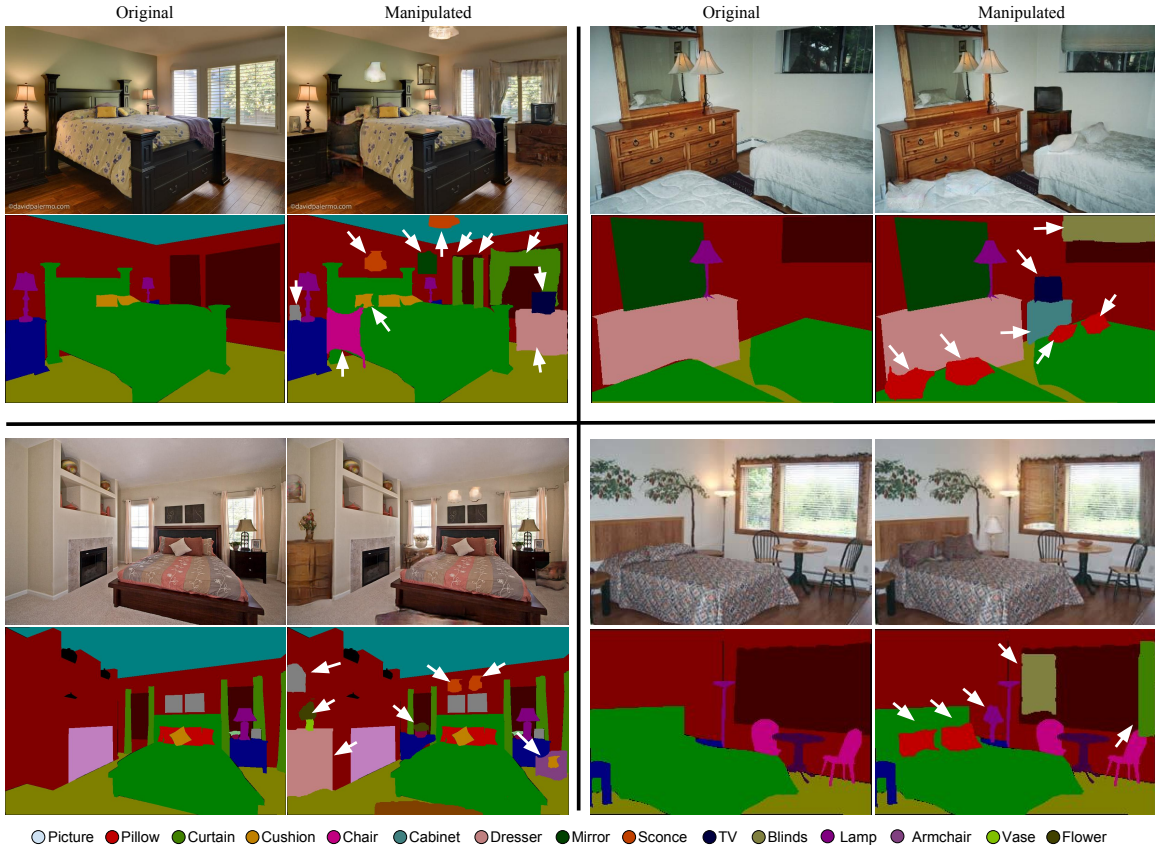


Figure 3.11: Examples of image manipulation results on indoor images.

3.5 Discussions

In this chapter, we presented a hierarchical framework for semantic image manipulation. We first learn to generate the pixel-wise semantic label maps given the

initial object bounding boxes. Then we learn to generate the manipulated image from the predicted label maps. Such framework allows the user to manipulate images at object-level by adding, removing, and moving an object bounding box at a time. Experimental evaluations demonstrate the advantages of the hierarchical manipulation framework over existing image generation and context hole-filling models, both qualitatively and quantitatively. We further demonstrate its practical benefits in semantic object manipulation, interactive image editing and data-driven image editing. Future research directions include preserving the object identity and providing affordance as additional user input during image manipulation.

CHAPTER IV

Learning Structured Representations for Human Motion Generation

4.1 Introduction

Modeling the dynamics of human motion — both facial and full body motion — is a fundamental problem in computer vision, graphics, and machine intelligence, with applications ranging from virtual characters (*de Aguiar et al.*, 2008; *Beeler et al.*, 2011), video-based animation and editing (*Yang et al.*, 2011a; *Suwajanakorn et al.*, 2015, 2017), and human-robot interfaces (*Sermanet et al.*, 2017). Human motion is known to be highly structured and can be modeled as a sequence of atomic units that we refer to as *motion modes*. A motion mode captures the *short-term* temporal dynamics of a human action (e.g., smiling or walking), including its related stylistic attributes (e.g., how wide is the smile, how fast is the walk). Over the *long-term*, a human action sequence can be segmented into a series of motion modes with transitions between them (e.g., a transition from a neutral expression to smiling to laughing). This structure is well known (referred to as basis motions (*Rose et al.*, 1996) or walk cycles) and widely used in computer animation.

This chapter leverages this structure to learn to generate human motion sequences, i.e., given a short human action sequence (present motion mode), we want to synthe-

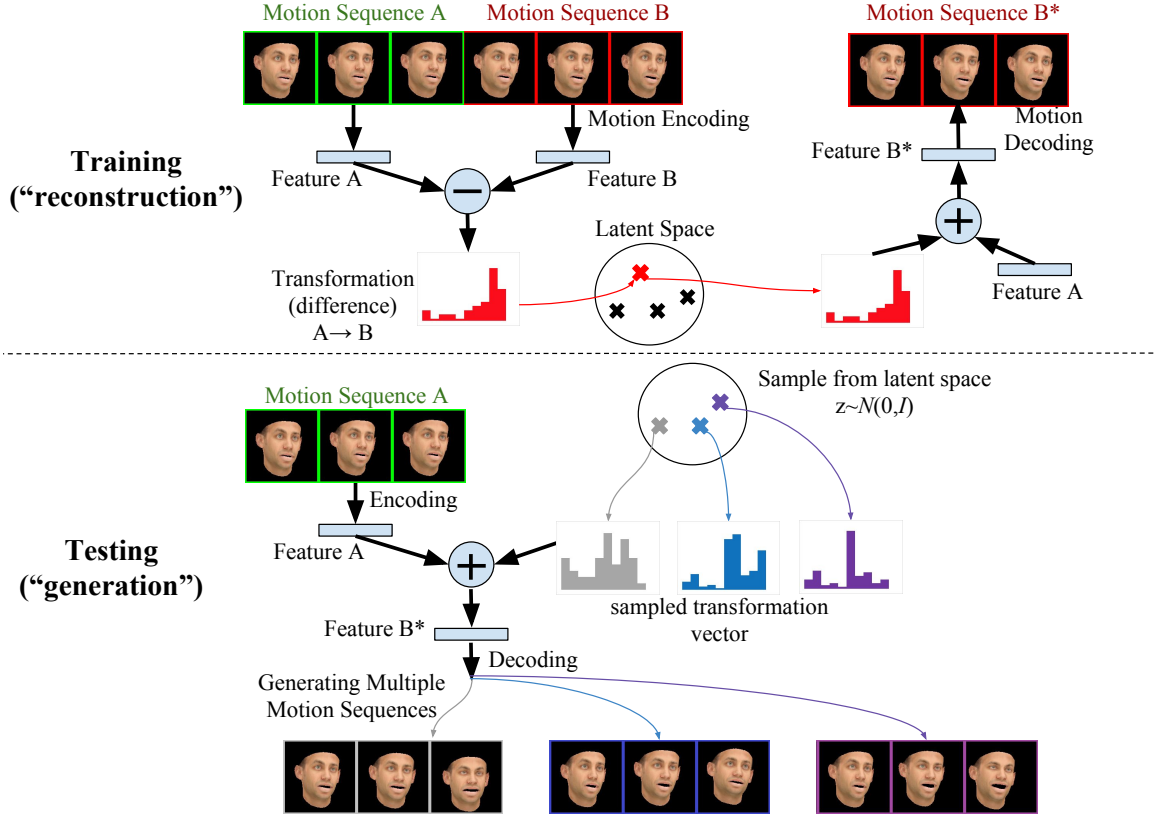


Figure 4.1: Top: Learning motion sequence generation using *Motion Transformation VAE*. Bottom: Generating multiple future motion sequences from the transformation space.

size the action going forward (future motion mode). We hypothesize that (1) each motion mode can be represented as a low-dimensional feature vector, and (2) transitions between motion modes can be modeled as *transformations* of these features. As shown in Figure 4.1, we present a novel model termed *Motion Transformation Variational Auto-Encoders (MT-VAE)* for learning motion sequence generation. Our MT-VAE is implemented using an LSTM encoder-decoder that embeds each short sub-sequence into a feature vector that can be decoded to reconstruct the motion. We further assume that the transition between current and future modes can be captured by a certain transformation. In the chapter, we demonstrate that the proposed MT-VAE learns a motion feature representation in an unsupervised way.

A challenge with human motion is that it is inherently multimodal, i.e., the same

initial motion mode could transition into different motion modes (e.g., a smile could transition to a frown, or a smile while looking left, or a wider smile, etc.). A deterministic model would not be able to learn these variations and may collapse to a single-mode distribution. Our MT-VAE supports a stochastic sampling of the feature transformations to generate multiple plausible output motion modes from a single input. This allows us to model transitions that may be rare (or potentially absent) in the the training set.

We demonstrate our approach on both facial and full human body motions. In both domains, we conduct extensive ablation studies and comparisons with previous work showing that our generation results are more plausible (i.e., better preserve the structure of human dynamics) and diverse (i.e., explore multiple motion modes). We further demonstrate applications like 1) analogy-based motion transfer (e.g., transferring the act of smiling from one pose to another pose) and 2) future video synthesis (i.e., generating multiple possible future videos given input frames with human motions). Our key contributions are summarized as follows:

- We propose a generative motion model that consists of a sequence-level motion feature embedding and feature transformations, and show that it can be trained in an unsupervised manner.
- We show that stochastically sampling the transformation space is able to generate future motion dynamics that are diverse and plausible.
- We demonstrate applications of the learned model to challenging tasks like motion transfer and future video synthesis for both facial and human body motions.

4.2 Related Work

Understanding and modeling human motion dynamics has been a long-standing problem for decades (*Bregler, 1997; Efros et al., 2003; Gorelick et al., 2007*). Due to the high dimensionality of video data, early work mainly focused on learning hierarchical spatio-temporal representations for video event and action recognition (*Laptev, 2005; Wang et al., 2011, 2012*). In recent years, predicting and synthesizing motion dynamics using deep neural networks has become a popular research topic. *Walker et al. (2015)* and *Fischer et al. (2015)* learn to synthesize dense flow in the future from a single image. *Walker et al. (2016)* extended the deterministic prediction framework by modeling the flow uncertainty using variational auto-encoders. *Chao et al. (2017)* proposed a recurrent neural network to generate movement of 3D human joints from a single observation with a 3D in-network projection layer. Taking one step further, *Villegas et al. (2017b)*, *Walker et al. (2017)* explored hierarchical structure (e.g., 2D human joints) for motion prediction in the future using recurrent neural networks. *Li et al. (2018)* proposed an auto-conditional recurrent framework to generate long-term human motion dynamics through time. Besides human motion, face synthesis and editing is another interesting topic in vision and graphics. Methods for reenacting and interpolating face sequences in video have been developed (*Yang et al., 2011b, 2012a; Thies et al., 2016; Averbuch-Elor et al., 2017*) based on a 3D morphable face representation *Blanz and Vetter (1999)*. Very recently, *Suwajanakorn et al. (2017)* introduced a speech-driven face synthesis system that learns to generate lip motions with a recurrent neural network.

Besides the flow representation, motion synthesis has been explored in a broader context, namely, video generation. For example, synthesizing video sequence in the future from a single or multiple video frames as initialization. Early works employed patch-based method for short-term video generation using mean squared mean squared loss (*Srivastava et al., 2015*) or perceptual loss (*Mathieu et al., 2016*).

Given an atomic action as additional condition, previous works extended with action-conditioned (i.e., rotation, location, etc) architectures that enable better semantic control in video generation (*Hinton et al.*, 2011; *Oh et al.*, 2015; *Finn et al.*, 2016b; *Yang et al.*, 2015). Due to the difficulty in holistic video frame prediction, the idea of disentangling video factors into motion and content is explored in *Villegas et al.* (2017a); *Denton and Birodkar* (2017); *Xue et al.* (2016); *Vondrick et al.* (2016); *Tulyakov et al.* (2017); *Wichers et al.*. Video generation has also been approached with architectures that output multinomial distribution vectors over the possible pixel values for each pixel in the generated frame (*Kalchbrenner et al.*, 2016).

The notion of feature transformations has also been exploited for other tasks. *Mikolov et al.* (2013) showcased the composition additive property of word vectors learned in an unsupervised way from language data; *Kulkarni et al.* (2015), *Reed et al.* (2015) suggested that additive transformation can be achieved via reconstruction or prediction task by learning from parallel paired image data. In the video domain, *Wang et al.* (2016) studied a transformation-aware representation for semantic human action classification; *Zhou and Berg* (2016) investigated time-lapse video generation given additional class labels.

Multimodal conditional generation has recently been explored for images (*Sohn et al.*, 2015; *Zhu et al.*, 2017b), sketch drawings (*Ha and Eck*, 2018), natural language (*Bowman et al.*, 2015; *Hu et al.*, 2017), and video prediction (*Babaeizadeh et al.*, 2018; *Denton and Fergus*, 2018). As noted in previous work, learning to generate diverse and plausible visual data is very challenging for the following reasons: first, mode collapse may occur without one-to-many pairs. Collecting sequence data where one-to-many pairs exist is non-trivial. Second, posterior collapse could happen when the generation model is based on a recurrent neural network.

4.3 Problem Formulation and Methods

We start by giving an overview of our problem. We are given a sequence of T observations $S_A = [x_1, x_2, \dots, x_T]$, where $x_t \in \mathbb{R}^D$ is a D dimensional vector representing the observation at time t . These observations encode the structure of the moving object and can be represented in different ways, for e.g., as keypoint locations or shape and pose parameters. *Changes* in these observations encode the motion that we are interested in modeling. We refer to the entire sequence as a *motion mode*. Given a motion mode, $S_A \in \mathbb{R}^{T \times D}$, we aim to build a model that is capable of predicting a future motion mode, $S_B = [y_1, y_2, \dots, y_T]$, where $y_t \in \mathbb{R}^D$ represents the predicted t -th step in the *future*, i.e., $y_1 = x_{T+1}$. We first start with a discussion of two potential baseline models that could be used for this task (Section 4.3.1), and then present our method (Section 4.3.2).

4.3.1 Preliminaries

Prediction LSTM for Sequence Generation. Figure 4.2(a) shows a simple encoder-decoder LSTM (*Hochreiter and Schmidhuber, 1997; Srivastava et al., 2015*) as a baseline for the motion prediction task. At time t , the encoder LSTM takes the motion x_t as input and updates its internal representation. After going through the entire motion mode S_A , it outputs a fixed-length feature $e_A \in \mathbb{R}^{N_e}$ as an intermediate representation. We initialize the internal representation of decoder LSTM using the feature e_A computed. At time t of the decoding stage, the decoder LSTM predicts the motion y_t . This way, the decoder LSTM gradually predicts the entire motion mode $S_B^* = [y_1, y_2, \dots, y_T]$ in the future within T steps. We denote the encoder LSTM as function $f : \mathbb{R}^{T \times D} \rightarrow \mathbb{R}^{N_e}$ and the decoder LSTM as function $g : \mathbb{R}^{N_e} \rightarrow \mathbb{R}^{T \times D}$. As a design choice, we initialize the decoder LSTM with additional input x_T for smoother prediction.

Vanilla VAE for Sequence Generation. As the deterministic LSTM model fails to reflect the multimodal nature of human motion, we consider a statistical model, $p_\theta(S_B|S_A)$, parameterized by θ . Given the observed sequence S_A , the model estimates a probability for the possible future sequence S_B instead of a single outcome. To model the multimodality (i.e., S_A can transition to different S_B 's), a latent variable z (sampled from prior distribution) is introduced to capture the inherent uncertainty. The future sequence S_B is generated as follows:

1. Sample latent variable $z \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$;
2. Given S_A and z , generate a sequence of length T : $S_B \sim p_\theta(S_B|z, S_A)$;

Following previous work on VAEs (*Kingma and Welling, 2014; Sohn et al., 2015; Gregor et al., 2015; Yan et al., 2016a; Walker et al., 2016; Xue et al., 2016; Walker et al., 2017*), the objective is to maximize the variational lower-bound of the conditional log-probability $\log p_\theta(S_B|S_A)$:

$$\mathcal{L}_{\text{VAE}} = -KL(q_\phi(z|S_B, S_A)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|S_B, S_A)}[\log p_\theta(S_B|S_A, z)] \quad (4.1)$$

In Eq. 4.1, $q_\phi(z|S_B, S_A)$ is referred as an auxiliary posterior that approximates the true posterior $p_\theta(z|S_B, S_A)$. Specifically, the prior $p_\theta(z)$ is assumed to be $\mathcal{N}(\mathbf{0}, \mathbf{I})$. The posterior $q_\phi(z|S_B, S_A)$ is a multivariate Gaussian distribution with mean and variance μ_ϕ and σ_ϕ^2 , respectively. Intuitively, the first term in Eq. 4.1 regularizes the auxiliary posterior $q_\phi(z|S_B, S_A)$ with prior $p_\theta(z)$. The second term $\log p_\theta(S_B|S_A, z)$ can be considered as an auto-encoding loss, where we refer to $q_\phi(z|S_B, S_A)$ as an encoder or recognition model, and $p_\theta(S_B|z, S_A)$ as a decoder or generation model.

As shown in Figure 4.2(b), the vanilla VAE model adopts similar LSTM encoder and decoder for sequence processing. In contrast to Prediction LSTM model, the vanilla VAE decoder takes both motion feature e_A and latent variable z into account. Ideally, this allows to generate diverse motion sequences by drawing different samples

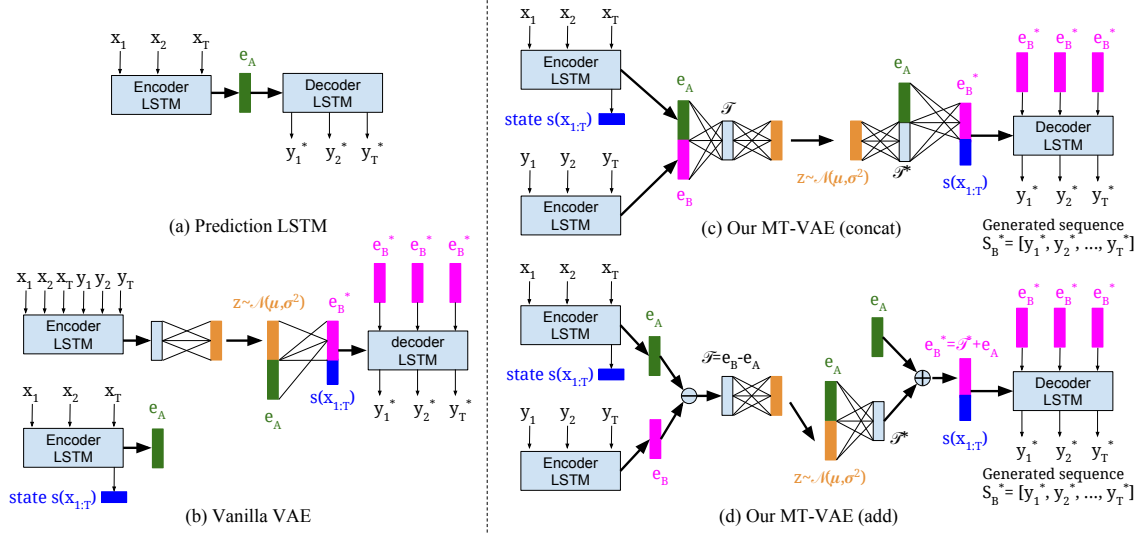


Figure 4.2: Illustrations of different models for motion sequence generation. $s(x_{1:T})$ indicates the hidden state of the Encoder LSTM at time T .

from the latent space. However, the semantic role of the latent variable z in this vanilla VAE model is not straight-forward and may not effectively represent long-term trends (e.g., dynamics in a specific motion mode or during change of modes).

4.3.2 Motion-to-Motion Transformations in Latent Space

To further improve motion sequence generation beyond vanilla VAE, we propose to explicitly enforce the structure of motion modes in the latent space. We assume that (1) each motion mode can be represented as low-dimensional feature vector, and (2) transitions between motion modes can be modeled as *transformations* of these features. Our design is also supported by early studies on hierarchical motion modeling and prediction (Bregler, 1997; Smith and Vul, 2013; Lan et al., 2014).

We present a *Motion Transformation VAE (or MT-VAE)* (Fig. 4.2(c)) with four components:

1. An LSTM encoder $f : \mathbb{R}^{T \times D} \rightarrow \mathbb{R}^{N_e}$ maps the input sequences into motion features through $e_A = f(S_A)$ and $e_B = f(S_B)$, respectively.
2. A latent encoder $h_{e \rightarrow z} : \mathbb{R}^{2 \times N_e} \rightarrow \mathbb{R}^{N_z}$ computes the transformation in the latent

space $z = h_{e \rightarrow z}([e_A, e_B])$ by concatenating motion features e_A and e_B . Here, N_z indicates the latent space dimension.

3. A latent decoder $h_{z \rightarrow e} : \mathbb{R}^{N_z + N_e} \rightarrow \mathbb{R}^{N_e}$ synthesizes the motion feature in the future from latent transformation z and current motion feature e_A via $e_B^* = h_{z \rightarrow e}([z, e_A])$.
4. An LSTM decoder $g : \mathbb{R}^{N_e} \rightarrow \mathbb{R}^{T \times D}$ synthesizes the future sequence given motion feature: $S_B^* = g(e_B^*)$.

Similar to the Prediction LSTM, we use an LSTM encoder/decoder to map motion modes into feature space. The MT-VAE further maps these features into *latent transformations* and stochastically samples these transformations. As we demonstrate, this change makes the model more expressive and leads to more plausible results. Finally, in the sequence decoding stage of MT-VAE, we feed the synthesized motion feature e_B^* as input to the decoder LSTM, with internal state initialized using the same motion feature e_B^* with an additional input x_t .

4.3.3 Additive Transformations in Latent Space

Although MT-VAE explicitly models motion transformations in latent space, this space might be unconstrained because the transformations are computed from vector concatenation of motion features e_A and e_B in our latent encoder $h_{e \rightarrow z}$. To better regularize the transformation space, we present an additive variant of MT-VAE, that is depicted in Figure 4.2(d). To distinguish between the two variants, we call the previous model *MT-VAE (concat)* and this model *MT-VAE (add)*, respectively. Our model is inspired by recent success of *deep analogy-making* methods (Reed et al., 2015; Villegas et al., 2017a) where a relation (or transformation) between two examples can be represented as a difference in the embedding space. In this model, we strictly constrain the latent encoding and decoding steps as follows:

1. Our latent encoder $h_{\mathcal{T} \rightarrow z} : \mathbb{R}^{N_e} \rightarrow \mathbb{R}^{N_z}$ computes the difference between two motion features e_A and e_B via $\mathcal{T} = e_B - e_A$; then it maps the difference feature \mathcal{T} into a transformation in the latent space via $z = h_{\mathcal{T} \rightarrow z}(\mathcal{T})$.
2. Our latent decoder $h_{z \rightarrow \mathcal{T}} : \mathbb{R}^{N_z + N_e} \rightarrow \mathbb{R}^{N_e}$ reconstructs the difference feature \mathcal{T}^* from latent variable z and current motion feature e_A via $\mathcal{T}^* = h_{z \rightarrow \mathcal{T}}(z, e_A)$.
3. Finally, we apply a simple additive interaction to reconstruct the motion feature via $e_B^* = e_A + \mathcal{T}^*$;

In step one, we infer the latent variable using $h_{\mathcal{T} \rightarrow z}$ from the difference of e_A and e_B (instead of applying a linear layer on concatenated vectors). Intuitively, the latent code is expected to capture the mode transition from the current motion to the future motion rather than a concatenation of two modes. In step two, we reconstruct the transformation from the latent variable via $h_{z \rightarrow \mathcal{T}}(z, e_A)$ where z is obtained from recognition model. In this design, the feature difference is dependent on both latent transformation z and current motion feature e_A . Alternatively, we can make our latent decoder $h_{z \rightarrow \mathcal{T}}$ context-free by removing input from motion feature e_A . This way, the latent decoder is supposed to hallucinate the motion difference solely from the latent space. We provide this ablation study in Section 4.4.1.

Besides the architecture-wise regularization, we introduce two additional objectives while training our model.

Cycle Consistency. As mentioned previously, our training objective \mathcal{L}_{VAE} in Eq. 4.1 is composed of a KL term and a reconstruction term at each frame. The KL term regularizes the latent space, while the reconstruction term ensures that the data can be explained by our generative model. However, we do not have direct regularization in the feature space. We therefore introduce a cycle-consistency loss in Eq. 4.2 (for

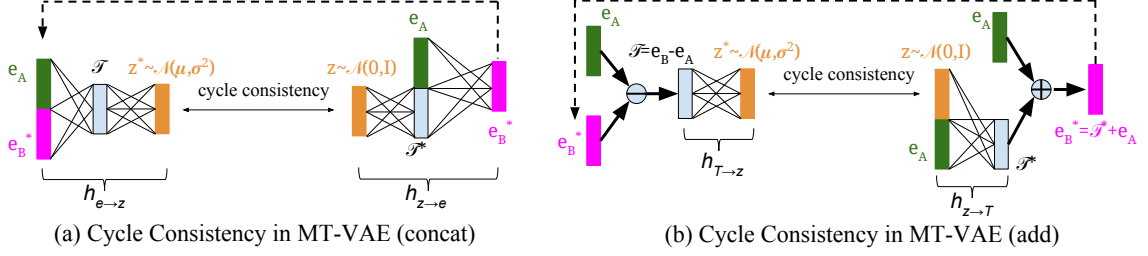


Figure 4.3: Illustrations of cycle consistency in MT-VAE variations.

MT-VAE (concat)) and Eq. 4.3 (for MT-VAE (add)). Figure 4.3 illustrates the cycle consistency in details.

$$\mathcal{L}_{\text{cycle}}^{\text{concat}} = \|z^* - z\|, \text{ where } z^* = h_{e \rightarrow z}([e_A, h_{z \rightarrow e}(z, e_A)]) \text{ and } z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (4.2)$$

$$\mathcal{L}_{\text{cycle}}^{\text{add}} = \|z^* - z\|, \text{ where } z^* = h_{T \rightarrow z}(h_{z \rightarrow T}(z, e_A)) \text{ and } z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (4.3)$$

In our preliminary experiments, we also investigated a consistency loss with a bigger cycle (involving the actual motion sequences) during training but we found it ineffective as a regularization term in our setting. We hypothesize that vanishing or exploding gradients make the cycle-consistency objective less effective, which is a known issue when training recurrent neural networks.

Motion Coherence. Specific to our motion generation task, we introduce a motion coherence loss in Eq. 4.4 that encourages a smooth transition in velocity in the first K steps of prediction. We define the velocity $v_1 = y_1 - x_T$ and $v_k = y_k - y_{k-1}$ when $k \geq 2$. Intuitively, such loss prevents the generated sequence from deviating too far from the future sequence sampled from the prior.

$$\mathcal{L}_{\text{motion}} = \frac{1}{K} \sum_{t=1}^K \|v_t^* - v_t\|, \text{ where } g(e_B^z) = [y_1^*, \dots, y_T^*] \text{ and } z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (4.4)$$

Finally, we summarize our overall loss in Eq. 4.5, where λ_{cycle} and λ_{motion} are two

balancing hyper-parameters for cycle consistency and motion coherence, respectively.

$$\mathcal{L}_{\text{MT-VAE}} = \mathcal{L}_{\text{VAE}} + \lambda_{\text{cycle}}\mathcal{L}_{\text{cycle}} + \lambda_{\text{motion}}\mathcal{L}_{\text{motion}} \tag{4.5}$$

4.4 Experiments

Datasets. The evaluation is conducted on the datasets involving two representative human motion modeling tasks: Affect-in-the-wild (**Aff-Wild**) (*Zafeiriou et al.*) for facial motions and **Human3.6M** (*Ionescu et al., 2014*) for full body motions. To better focus on face motion modeling (e.g., expressions and head movements), we leveraged the 3D morphable face model (*Paysan et al., 2009; Blanz and Vetter, 1999*) (e.g., face identity, face expression, and pose) in our experiments. We fitted 198-dim identity coefficients, 29-dim expression coefficients, and 6-dim pose parameters to each frame with a pre-trained 3DMM-CNN model (*Tran et al., 2017*), followed by a face fitting algorithm (*Zhu et al., 2016b*) based on optimization. Human3.6M is a large-scale database containing more than 800 human motion sequences captured by 11 professional actors (3.6 million frames in total) in an indoor environment. For experiments on Human3.6M, we used the raw 2D trajectories of 32 keypoints and further normalized the data into coordinates within the range $[-1, 1]$.

Architecture Design. Our MT-VAE model consists of four components: sequence encoder network, sequence decoder network, latent encoder network, and latent decoder network. We build our sequence encoder and decoder using Long Short-term Memory units (LSTMs) (*Hochreiter and Schmidhuber, 1997*). We used 1-layer LSTM with 1,024 hidden units for both networks. Given past and future motion features extracted from our sequence encoder network, we build three fully-connected layers with skip connections within our latent encoding network. We adopted a similar architecture (three fully-connected layers with skip connections) for our latent decoder

network.

Please see the website for more visualizations: <https://goo.gl/2Q69Ym>.

4.4.1 Multimodal Motion Generation

Table 4.1: Quantitative evaluations for multimodal motion generation. We compare against two simple data-driven baselines for quantitative comparison: *Last-step Motion* that recursively applies the motion (velocity only) from the last step observed; *Sequence Motion* that recursively adds the average sequence velocity from the observed frames. Top: Results on *Aff-Wild* with facial expression coefficients. Bottom: Results on *Human3.6M* with 2D joints.

Method / Metric	R-MSE \downarrow ($\times 10^{-1}$)		S-MSE \downarrow ($\times 10^{-1}$)		Test CLL \uparrow ($\times 10^3$)
	train	test	train	test	
Last-step Motion	—	—	63.8 ± 1.31	74.7 ± 5.59	0.719 ± 0.077
Sequence Motion	—	—	18.4 ± 0.25	19.1 ± 1.02	1.335 ± 0.057
Prediction LSTM	—	—	1.53 ± 0.01	3.03 ± 0.06	2.232 ± 0.003
Vanilla VAE	0.32 ± 0.00	1.28 ± 0.02	0.79 ± 0.00	1.79 ± 0.03	2.749 ± 0.012
MT-VAE (concat)	0.22 ± 0.00	0.73 ± 0.01	1.04 ± 0.00	1.76 ± 0.03	2.817 ± 0.023
MT-VAE (add)	0.20 ± 0.00	0.47 ± 0.01	1.02 ± 0.00	1.54 ± 0.04	3.147 ± 0.018

Method / Metric	R-MSE \downarrow		S-MSE \downarrow		Test CLL \uparrow ($\times 10^4$)
	train	test	train	test	
Last-step Motion	—	—	35.2 ± 0.49	32.1 ± 0.80	0.390 ± 0.004
Sequence Motion	—	—	37.8 ± 0.49	35.2 ± 0.73	0.406 ± 0.003
Prediction LSTM	—	—	1.69 ± 0.02	11.2 ± 0.17	0.602 ± 0.002
Vanilla VAE	0.36 ± 0.00	1.05 ± 0.02	3.18 ± 0.02	3.88 ± 0.05	0.993 ± 0.011
MT-VAE (concat)	0.36 ± 0.00	0.97 ± 0.02	2.26 ± 0.03	2.84 ± 0.05	1.033 ± 0.010
MT-VAE (add)	0.25 ± 0.00	0.75 ± 0.01	2.37 ± 0.02	2.87 ± 0.05	1.141 ± 0.009

We evaluate our model’s capacity to generate diverse and plausible future motion patterns for a given sequence on the *Aff-Wild* and *Human3.6M* test sets. Given sequence S_A as initialization, we generated multiple motion trajectories in the future using our proposed sampling and generation process. For the Prediction LSTM model, we only sample one motion trajectory in the future since the predicted future is deterministic.

Quantitative Evaluations. We evaluate our model and baselines quantitatively using the minimum squared error metric and conditional log-likelihood metric, which have been used in evaluating conditional generative models (*Sohn et al.*, 2015; *Walker*

et al., 2016; Yan *et al.*, 2016a; Babaeizadeh *et al.*, 2018). As defined in Eq. 4.6, *Reconstruction* minimum squared error (or R-MSE) measures the squared error of the closest reconstruction to ground-truth when sampling latent variables from the recognition model. This is a measure of the quality of reconstruction given both current and future sequences. As defined in Eq. 4.7, *Sampling* minimum squared error (or S-MSE) measures the squared error of the closest sample to ground-truth when sampling latent variables from prior. This is a measure of how close our samples are to the reference future sequences.

$$\text{R-MSE} = \min_{1 \leq k \leq K} \|S_B - S_B^*(z^{(k)})\|^2, \text{ where } z^{(k)} \sim q_\phi(z|S_A, S_B). \quad (4.6)$$

$$\text{S-MSE} = \min_{1 \leq k \leq K} \|S_B - S_B^*(z^{(k)})\|^2, \text{ where } z^{(k)} \sim p_\theta(z). \quad (4.7)$$

In terms of generation diversity and quality, a good generative model is expected to achieve low R-MSE and S-MSE values, given sufficient number of samples. Note that *posterior collapse* issue is usually featured by low S-MSE but high R-MSE, as latent z sampled from the recognition model is being ignored to some extent. In addition, we measure the test conditional log-likelihood of the ground-truth sequences under our model via Parzen window estimation (with a bandwidth determined based on the validation set). We believe that Parzen window estimation is a reasonable approach for our setting as the dimensionality of data (sequence of keypoints) is not too high (unlike in the case of high-resolution videos). For each example, we used 50 samples to compute R-MSE metric, and 500 samples to compute S-MSE and conditional log-likelihood metrics. On Aff-Wild, we evaluate the models on 32-step expression coefficients prediction ($29 \times 32 = 928$ dimensions in total). On Human3.6M, we evaluate the models on 64-step 2D joints prediction ($64 \times 64 = 4096$ dimensions in total). Please note that such measurements are approximate, as we do not evaluate the model performance for every sub-sequence (e.g., essentially, every frame can serve

as a starting point). Instead, we repeat the evaluations every 16 frames on Aff-Wild dataset and every 100 frames on Human3.6M dataset.

As we see in Table 4.1, data-driven approaches that simply repeat the motion computed from last-step velocity or averaged over the observed sequence performed poorly on both datasets. In contrast, the Prediction LSTM (*Villegas et al., 2017b*) baseline greatly reduces the S-MSE metric compared to simple data-driven approaches, due to the deep sequence encoder and decoder architecture in modeling more complex motion dynamics through time. Among all three models using latent variables, our MT-VAE (add) model achieve the best quantitative performance. Compared to MT-VAE (concat) that adopts vector concatenation, our additive version achieves lower reconstruction error with similar sampling error. This suggests that the MT-VAE (add) model is able to regularize the learning of motion transformation further.

Qualitative Results. We provide qualitative side-by-side comparisons across different models in Figure 4.4. For Aff-Wild, we render 3D face models using the generated expression-pose parameters along with the original identity parameters. For Human3.6M, we directly visualize the generated 2D keypoints. As shown in the generated sequences, our MT-VAE model is able to generate multiple diverse and plausible sequences in the future. In comparison, the sequences generated by Vanilla VAE are less realistic. For example, given a sitting down motion (lower-left part in Fig. 4.4) as initialization, the vanilla model fails to predict the motion trend (sitting down), while creating some artifacts (e.g., scale change) in the future prediction. Also note that MT-VAE produces more natural transitions from the last observed frame to the first generated one (see mouth shapes in the face motion examples and distances between two legs in full-body examples). This demonstrates that MT-VAE learns a more robust and structure-preserving representation of motion sequences compared to other baselines.



Figure 4.4: Multimodal Sequence Generation. Given an input sequence (green boundary), we generate future sequences (red boundary). We predict 32 frames given 8 frames for face motion, and 64 frames given 16 frames for human body motion. Given the initial frames as condition, we demonstrate (top to bottom) the ground truth sequence, Prediction LSTM, Vanilla VAE, and our MT-VAE model. Overall, our model produces (1) diverse and structured motion patterns and (2) more natural transitions from the last frame observed to the first frame generated (See the subtle mouth shape and scale change from the last observed frame to the first generated one).

Table 4.2: Crowd-sourced Human Evaluations on Human3.6M. *We did not include Prediction LSTM for the diversity evaluation, as it makes deterministic prediction.

Metric	Vanilla VAE	SVG	Our MT-VAE (add)	Pred LSTM
Realism (%)	19.2	23.8	26.4	30.6
Diversity (%)	51.6	22.3	26.1	0.0*

Table 4.3: Ablation Study on Different variants of MT-VAE (add) model: We evaluate models trained without motion coherence objective, without cycle consistency objective, and the model with context-free latent decoder.

Method / Metric	R-MSE (test) ↓	S-MSE (test) ↓	Test CLL ↑ ($\times 10^4$)
ADD	0.75 ± 0.01	2.87 ± 0.05	1.141 ± 0.009
ADD w/o Motion Coherence	1.01 ± 0.02	2.93 ± 0.04	1.012 ± 0.014
ADD w/o Cycle Consistency	1.18 ± 0.03	2.71 ± 0.05	0.927 ± 0.019
ADD Context-free Decoder	0.31 ± 0.05	4.05 ± 0.05	1.299 ± 0.007

Crowd-sourced Human Evaluations. We conducted crowd-sourced human evaluations via Amazon Mechanical Turk (AMT) on 50 videos (10 Turkers per video) from Human3.6M dataset. This evaluation presents the past action, and 5 generated future actions for each method to a human evaluator and asks the person to select the most (1) realistic and (2) diverse results. In this evaluation, we also added comparisons to a recently published work (*Denton and Fergus, 2018*) on stochastic video prediction, which we refer to as SVG. Table 4.2 presents the percentage of users who selected each method for each task. The Prediction LSTM produces the most realistic but the least diverse result; *Babaeizadeh et al. (2018)* produces the most diverse but the least realistic result; Our MT-VAE model (we use the additive variant here) achieves a good balance between realism and diversity.

Ablation Study. We analyze variations of our MT-VAE (add) models on Human3.6M. As we see in Table 4.3, removing the cycle consistency or motion coherence results in a drop in reconstruction performance. This shows that cycle consistency and motion coherence encourage the motion feature to preserve motion structure and hence be more discriminative in nature. We also evaluate a *context-free* version of the MT-VAE (add) model, where the the transformation vector \mathcal{T}^* is not conditioned on

input feature e_A . This version produces poor S-MSE value since it is challenging for the additive latent decoder to hallucinate transformation vector \mathcal{T}^* solely from latent variable z .

4.4.2 Analogy-based Motion Transfer

We evaluate our model on an additional task of *transfer by analogy*. In this analogy-making experiment, we are given three motion sequences A, B (which is the subsequent motion of A), and C (which is a different motion sequence). The objective is to recognize the *transition* from A to B and transfer it to C. This experiment can demonstrate whether our learned latent space models the mode transition across motion sequences. Moreover, this task has numerous graphics applications like transferring expressions and their styles, video dubbing, gait style transfer, and video-driven animation (*Thies et al., 2016*).

In this experiment, we compare Prediction LSTM, Vanilla VAE, and our MT-VAE variants. For the stochastic models, we compute the latent variable z from motion sequence A and B via the latent encoder, i.e., $z = h_{\mathcal{T} \rightarrow z}(e_B - e_A)$, and then decode using motion sequence C as $e_D^* = h_{z \rightarrow \mathcal{T}}(z, e_C)$. For Prediction LSTM model, we directly performed the analogy-making in the feature space $e_D^* = e_B - e_A + e_C$ since there is no notion of a latent space in that model. As shown in Figure 4.5, our MT-VAE model is able to combine the transformation learned from A to B transitions with the structure in sequence C. The other baselines failed at either adapting the mode transition from A to B or preserving the structure in C. The analogy-based motion transfer task is significantly more challenging than motion generation, since the combination of three reference motion sequences A, B, and C may never appear in the training data. Yet, our model is able to synthesize realistic motions. Please note that motion modes may not explicitly correspond to semantic motions, as we learn the motion transformation in an unsupervised manner.

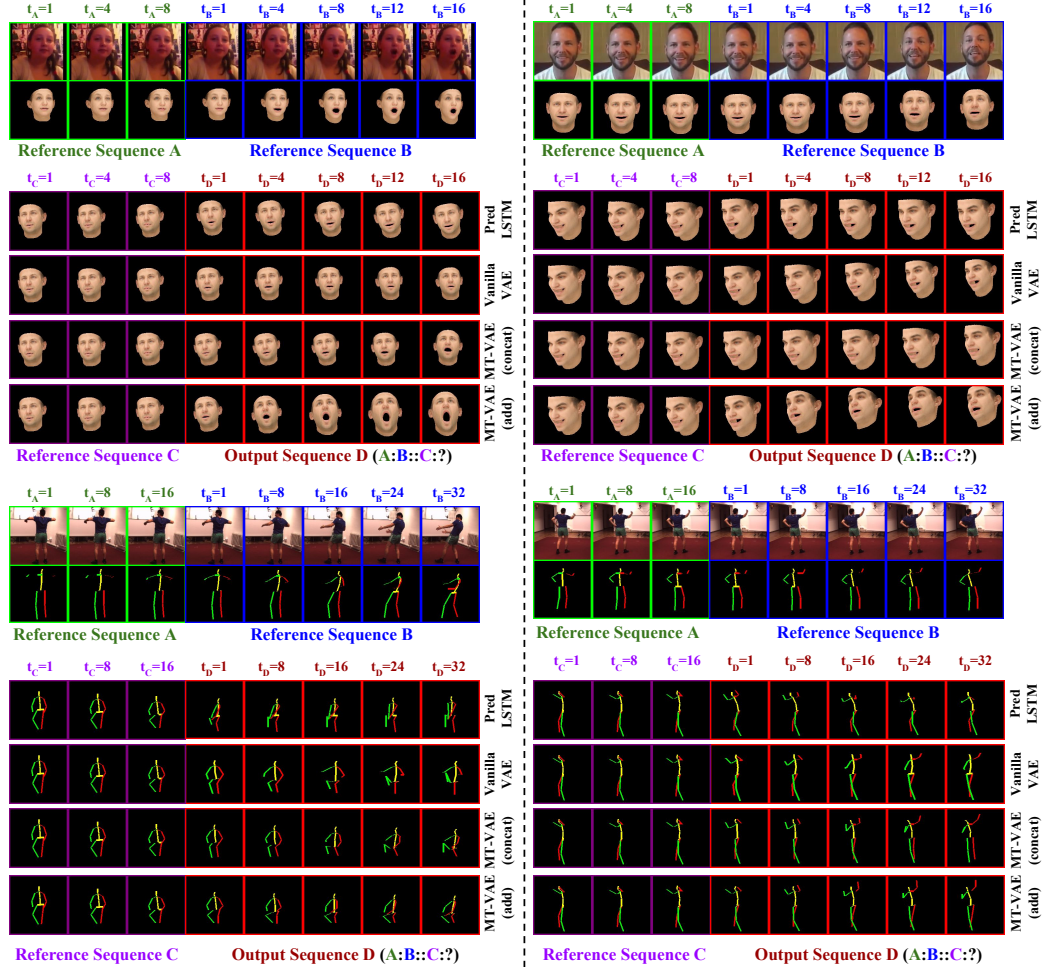


Figure 4.5: Analogy-based motion transfer. Given three motion sequences A, B, and C from test set, the objective is to extract the motion mode transition from A to B and then apply it to animate the future starting from sequence C. For fair comparison, we set the encoder Gaussian distribution parameter σ to zero during evaluation.

4.4.3 Towards Multimodal Hierarchical Video Generation

As an application, we showcase that our multimodal motion generation framework can be directly used for generating diverse and realistic pixel-level video frames in the future. We trained the keypoint-conditioned image generation model *Villegas et al.* (2017b) that takes both previous image frame A and predicted motion structure B (e.g., rendered face or human joints) as input and hallucinates image C by combining the image content adapted from A but with motion adapted from B. In Figure 4.6, we show a comparison of video generated in a deterministic way by Prediction LSTM

(i.e., single future), and in a stochastic way driven by the predicted motion sequence (i.e., multiple futures) from our MT-VAE (add) model. We use our generated motion sequences for performing video generation experiments on the Aff-Wild (with 8 input frames observed) and Human3.6M (with 16 input frames observed).

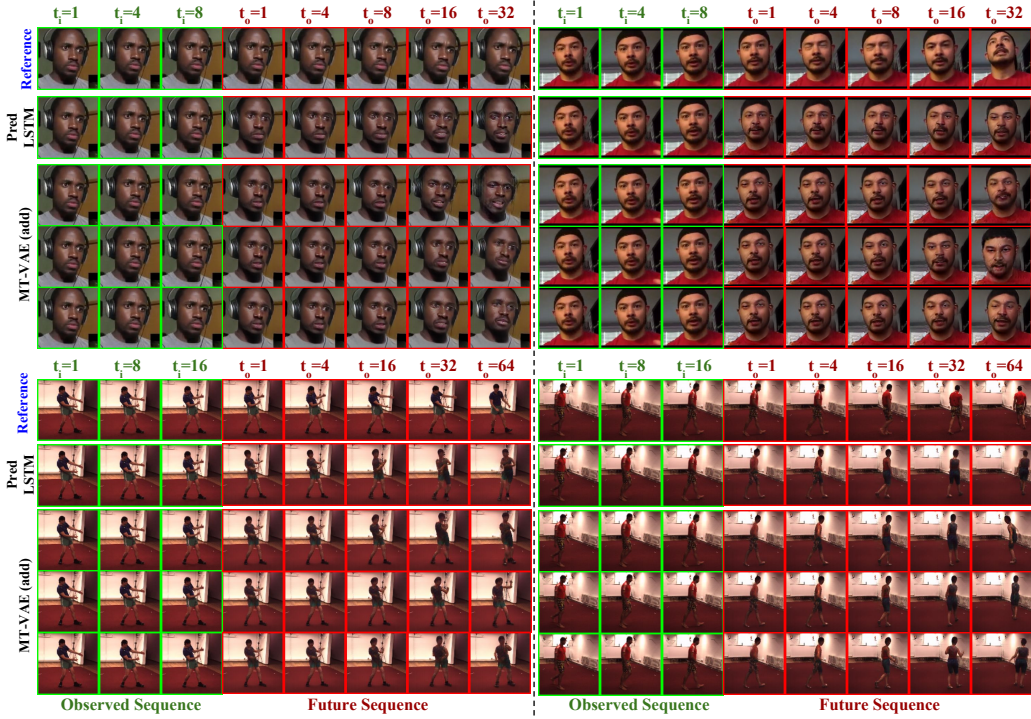


Figure 4.6: Multimodal Hierarchical video generation. Top rows: Face video generation results from 8 observed frames. Bottom rows: Human video generation results from 16 observed frames.

4.5 Discussions

Our goal in this work is to learn a conditional generative model for human motions. This is an extremely challenging problem in the general case and can require significant amount of training data to generate realistic results. Our work demonstrates that this can be accomplished with minimal supervision by enforcing a strong structure on the problem. In particular, we model long-term human dynamics as a set of motion modes with transitions between them, and construct a novel network ar-

chitecture that strongly regularizes this space and allows for stochastic sampling. We have demonstrated that this same idea can be used to model both facial and full body motion, independent of the representation used (i.e., shape parameters, keypoints).

CHAPTER V

Learning Geometry Representations for Single-View 3D Object Reconstruction

5.1 Introduction

Understanding the 3D world is at the heart of successful computer vision applications in robotics, rendering and modeling (*Szeliski, 2010*). It is especially important to solve this problem using the most convenient visual sensory data: 2D images. In this chapter, we propose an end-to-end solution to the challenging problem of predicting the underlying true shape of an object given an arbitrary single image observation of it. This problem definition embodies a fundamental challenge: Imagery observations of 3D shapes are interleaved representations of intrinsic properties of the shape itself (e.g., geometry, material), as well as its extrinsic properties that depend on its interaction with the observer and the environment (e.g., orientation, position, and illumination). Physically principled shape understanding should be able to efficiently disentangle such interleaved factors.

This observation leads to insight that an end-to-end solution to this problem from the perspective of learning agents (neural networks) should involve the following properties: 1) the agent should understand the physical meaning of how a 2D observation is generated from the 3D shape, and 2) the agent should be conscious about the out-

come of its interaction with the object; more specifically, by moving around the object, the agent should be able to correspond the observations to the viewpoint change. If such properties are embodied in a learning agent, it will be able to disentangle the shape from the extrinsic factors because these factors are trivial to understand in the 3D world. To enable the agent with these capabilities, we introduce a built-in camera system that can transform the 3D object into 2D images in-network. Additionally, we architect the network such that the latent representation disentangles the shape from view changes. More specifically, our network takes as input an object image and predicts its volumetric 3D shape so that the perspective transformations of predicted shape match well with corresponding 2D observations.

We implement this neural network based on a combination of image encoder, volume decoder and perspective transformer (similar to spatial transformer as introduced by *Jaderberg et al. (2015)*). During training, the volumetric 3D shape is gradually learned from single-view input and the feedback of other views through back-propagation. Thus at test time, the 3D shape can be directly generated from a single image. We conduct experimental evaluations using a subset of 3D models from ShapeNetCore (*Chang et al., 2015*). Results from single-class and multi-class training demonstrate excellent performance of our network for volumetric 3D reconstruction. Our main contributions are summarized below.

- We show that neural networks are able to predict 3D shape from single-view without using the ground truth 3D volumetric data for training. This is made possibly by the geometry-aware 2D silhouette loss.
- We train a single network for multi-class 3D object volumetric reconstruction and show its generalization potential to unseen categories.
- Compared to training with full azimuth angles, we demonstrate comparatively similar results when training with partial views.

5.2 Related Work

Representation learning for 3D objects. Recently, advances have been made in learning deep neural networks for 3D objects using large-scale CAD databases (*Wu et al.*, 2015; *Chang et al.*, 2015). *Wu et al.* (2015) proposed a deep generative model that extends the convolutional deep belief network (*Lee et al.*, 2009) to model volumetric 3D shapes. Different from *Wu et al.* (2015) that uses volumetric 3D representation, *Su et al.* (2015) proposed a multi-view convolutional network for 3D shape categorization with a view-pooling mechanism. These methods focus more on 3D shape recognition instead of 3D shape reconstruction. Recent work (*Tatarchenko et al.*, 2016; *Qi et al.*, 2016; *Girdhar et al.*, 2016; *Choy et al.*, 2016) attempt to learn a joint representation for both 2D images and 3D shapes. *Tatarchenko et al.* (2016) developed a convolutional network to synthesize unseen 3D views from a single image and demonstrated the synthesized images can be used them to reconstruct 3D shape. *Qi et al.* (2016) introduced a joint embedding by combining volumetric representation and multi-view representation together to improve 3D shape recognition performance. *Girdhar et al.* (2016) proposed a generative model for 3D volumetric data and combined it with a 2D image embedding network for single-view 3D shape generation. *Choy et al.* (2016) introduce a 3D recurrent neural network (3D-R2N2) based on long-short term memory (LSTM) to predict the 3D shape of an object from a single view or multiple views. Compared to these single-view methods, our 3D reconstruction network is learned end-to-end and the network can be even trained without ground truth volumes.

Concurrent to our work, *Rezende et al.* (2016) introduced a general framework to learn 3D structures from 2D observations with 3D-2D projection mechanism. Their 3D-2D projection mechanism either has learnable parameters or adopts non-differentiable component using MCMC, while our perspective projection network is both differentiable and parameter-free.

Representation learning by transformations. Learning from transformed sensory data has gained attention (*Memisevic and Hinton, 2007; Hinton et al., 2011; Reed et al., 2014; Michalski et al., 2014; Yang et al., 2015; Jaderberg et al., 2015; Yumer and Mitra, 2016*) in recent years. *Memisevic and Hinton (2007)* introduced a gated Boltzmann machine that models the transformations between image pairs using multiplicative interaction. *Reed et al. (2014)* showed that a disentangled hidden unit representations of Boltzmann Machines (disBM) could be learned based on the transformations on data manifold. *Yang et al. (2015)* learned out-of-plane rotation of rendered images to obtain disentangled identity and viewpoint units by curriculum learning. *Kulkarni et al. (2015)* proposed to learn a semantically interpretable latent representation from 3D rendered images using variational auto-encoders (*Kingma and Welling, 2014*) by including specific transformations in mini-batches. Complimentary to convolutional networks, *Jaderberg et al. (2015)* introduced a differentiable sampling layer that directly incorporates geometric transformations into representation learning. Concurrent to our work, *Wu et al. (2016a)* proposed a 3D-2D projection layer that enables the learning of 3D object structures using 2D keypoints as annotation.

5.3 Problem Formulation

In this section, we develop neural networks for reconstructing 3D objects. From the perspective of a learning agent (e.g., neural network), a natural way to understand one 3D object X is from its 2D views by transformations. By moving around the 3D object, the agent should be able to recognize its unique features and eventually build a 3D mental model of it as illustrated in Figure 5.1(a). Assume that $I^{(k)}$ is the 2D image from the k -th viewpoint $\alpha^{(k)}$ by projection $I^{(k)} = P(X; \alpha^{(k)})$, or rendering in graphics. An object X in a certain scene is the entanglement of shape, color and texture (its intrinsic properties) and the image $I^{(k)}$ is the further entanglement with viewpoint and illumination (extrinsic parameters). The general goal of understanding 3D objects

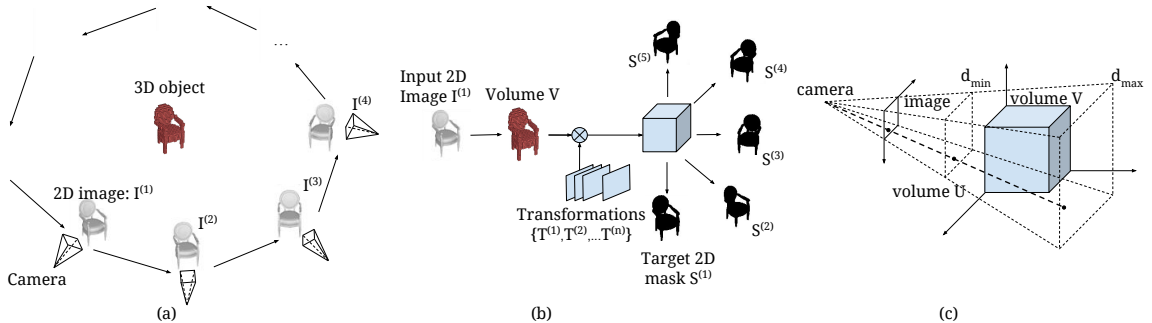


Figure 5.1: (a) Understanding 3D object from learning agent’s perspective; (b) Single-view 3D volume reconstruction with perspective transformation. (c) Illustration of perspective projection. The minimum and maximum disparity in the screen coordinates are denoted as d_{min} and d_{max} .

can be viewed as disentangling intrinsic properties and extrinsic parameters from a single image.

In this chapter, we focus on the 3D shape learning by ignoring the color and texture factors, and we further simplify the problem by making the following assumptions: 1) the scene is clean white background; 2) the illumination is constant natural lighting. We use the volumetric representation of 3d shape \mathbf{V} where each voxel \mathbf{V}_i is a binary unit. In other words, the voxel equals to one, i.e., $\mathbf{V}_i = 1$, if the i -th voxel space is occupied by the shape; otherwise $\mathbf{V}_i = 0$. Assuming the 2D silhouette $S^{(k)}$ is obtained from the k -th image $I^{(k)}$, we can specify the 3D-2D projection $S^{(k)} = P(\mathbf{V}; \alpha^{(k)})$. Note that 2D silhouette estimation is typically solved by object segmentation in real-world but it becomes trivial in our case due to the white background.

In the following sub-sections, we propose a formulation for learning to predict the volumetric 3D shape \mathbf{V} from an image $I^{(k)}$ with and without the 3D volume supervision.

5.3.1 Learning to Reconstruct Volumetric 3D Shape from Single-View

We consider single-view volumetric 3D reconstruction as a dense prediction problem and develop a convolutional encoder-decoder network for this learning task de-

noted by $\hat{\mathbf{V}} = f(I^{(k)})$. The encoder network $h(\cdot)$ learns a *viewpoint-invariant* latent representation $h(I^{(k)})$ which is then used by the decoder $g(\cdot)$ to generate the volume $\hat{\mathbf{V}} = g(h(I^{(k)}))$. In case the ground truth volumetric shapes \mathbf{V} are available, the problem can be easily considered as learning volumetric 3D shapes with a regular reconstruction objective in 3D space: $\mathcal{L}_{vol}(I^{(k)}) = \|f(I^{(k)}) - \mathbf{V}\|_2^2$.

In practice, however, *the ground truth volumetric 3D shapes may not be available for training*. For example, the agent observes the 2D silhouette via its built-in camera without accessing the volumetric 3D shape. Inspired by the space carving theory (Kutulakos and Seitz, 2000), we propose a silhouette-based volumetric loss function. In particular, we build on the premise that a 2D silhouette $\hat{S}^{(j)}$ projected from the generated volume $\hat{\mathbf{V}}$ under certain camera viewpoint $\alpha^{(j)}$ should match the ground truth 2D silhouette $S^{(j)}$ from image observations. In other words, if all the generated silhouettes $\hat{S}^{(j)}$ match well with their corresponding ground truth silhouettes $S^{(j)}$ for all j 's, then we hypothesize that the generated volume $\hat{\mathbf{V}}$ should be as good as one instance of *visual hull* equivalent class of the ground truth volume \mathbf{V} (Kutulakos and Seitz, 2000). Therefore, we formulate the learning objective for the k -th image as

$$\mathcal{L}_{proj}(I^{(k)}) = \sum_{j=1}^n \mathcal{L}_{proj}^{(j)}(I^{(k)}; S^{(j)}, \alpha^{(j)}) = \frac{1}{n} \sum_{j=1}^n \|P(f(I^{(k)}); \alpha^{(j)}) - S^{(j)}\|_2^2, \quad (5.1)$$

where j is the index of output 2D silhouettes, n is the number of silhouettes used for each input image and $P(\cdot)$ is the 3D-2D projection function. Note that the above training objective Eq. (5.1) enables training without using ground-truth volumes. The network diagram is illustrated in Figure 5.1(b). A more general learning objective is given by a combination of both objectives:

$$\mathcal{L}_{comb}(I^{(k)}) = \lambda_{proj} \mathcal{L}_{proj}(I^{(k)}) + \lambda_{vol} \mathcal{L}_{vol}(I^{(k)}), \quad (5.2)$$

where λ_{proj} and λ_{vol} are constants that control the tradeoff between the two losses.

5.3.2 Perspective Transformer Networks

As defined previously, 2D silhouette $S^{(k)}$ is obtained via perspective projection given input 3D volume \mathbf{V} and specific camera viewpoint $\alpha^{(k)}$. In this work, we implement the perspective projection (see Figure 5.1(c)) with a 4-by-4 transformation matrix $\Theta_{4 \times 4}$, where \mathbf{K} is camera calibration matrix and (\mathbf{R}, \mathbf{t}) is extrinsic parameters.

$$\Theta_{4 \times 4} = \begin{bmatrix} \mathbf{K} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (5.3)$$

For each point $\mathbf{p}_i^s = (x_i^s, y_i^s, z_i^s, 1)$ in 3D world coordinates, we compute the corresponding point $\mathbf{p}_i^t = (x_i^t, y_i^t, 1, d_i^t)$ in screen coordinates (plus disparity d_i^t) using the perspective transformation: $\mathbf{p}_i^s \sim \Theta_{4 \times 4} \mathbf{p}_i^t$.

Similar to the spatial transformer network introduced in *Jaderberg et al. (2015)*, we propose a 2-step procedure: (1) performing dense sampling from input volume (in 3D world coordinates) to output volume (in screen coordinates), and (2) flattening the 3D spatial output across disparity dimension. In the experiment, we assume that transformation matrix is always given as input, parametrized by the viewpoint α . Again, the 3D point (x_i^s, y_i^s, z_i^s) in input volume $\mathbf{V} \in \mathbb{R}^{H \times W \times D}$ and corresponding point (x_i^t, y_i^t, d_i^t) in output volume $\mathbf{U} \in \mathbb{R}^{H' \times W' \times D'}$ is linked by perspective transformation matrix $\Theta_{4 \times 4}$. Here, (W, H, D) and (W', H', D') are the width, height and depth of input and output volume, respectively.

We summarize the dense sampling step and channel-wise flattening step as follows.

$$U_i = \sum_n^H \sum_m^W \sum_l^D V_{nml} \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|) \max(0, 1 - |z_i^s - l|)$$

$$S_{n'm'} = \max_{l'} U_{n'm'l'} \quad (5.4)$$

Here, U_i is the i -th voxel value corresponding to the point (x_i^t, y_i^t, d_i^t) (where $i \in \{1, \dots, W' \times H' \times D'\}$). Note that we use the max operator for projection instead of summation along one dimension since the volume is represented as a binary cube where the solid voxels have value 1 and empty voxels have value 0. Intuitively, we have the following two observations: (1) each empty voxel will not contribute to the foreground pixel of S from any viewpoint; (2) each solid voxel can contribute to the foreground pixel of S only if it is visible from a specific viewpoint.

5.3.3 Training

As the same volumetric 3D shape is expected to be generated from different images of the object, the encoder network is required to learn a 3D view-invariant latent representation

$$h(I^{(1)}) = h(I^{(2)}) = \dots = h(I^{(k)}) \quad (5.5)$$

This sub-problem itself is a challenging task in computer vision (*Yang et al.*, 2015; *Kulkarni et al.*, 2015). Thus, we adopt a two-stage training procedure: first, we learn the encoder network for a 3D view-invariant latent representation $h(I)$ and then train the volumetric decoder with perspective transformer networks. As shown in *Yang et al.* (2015), a disentangled representation of 2D synthetic images can be learned from consecutive rotations with a recurrent network, we pre-train the encoder of our network using a similar curriculum strategy so that the latent representation only contains 3D view-invariant identity information of the object. Once we obtain an encoder network that recognizes the identity of single-view images, we next learn the volume generator regularized by the perspective transformer networks. To encourage the volume decoder to learn a consistent 3D volume from different viewpoints, we include the projections from neighboring viewpoints in each mini-batch so that the network has relatively sufficient information to reconstruct the 3D shape.

5.4 Experiments

ShapeNetCore. This dataset contains about 51,300 unique 3D models from 55 common object categories (*Chang et al., 2015*). Each 3D model is rendered from 24 azimuth angles (with steps of 15°) with fixed elevation angles (30°) under the same camera and lighting setup. We then crop and rescale the centering region of each image to $64 \times 64 \times 3$ pixels. For each ground truth 3D shape, we create a volume of $32 \times 32 \times 32$ voxels from its canonical orientation (0°).

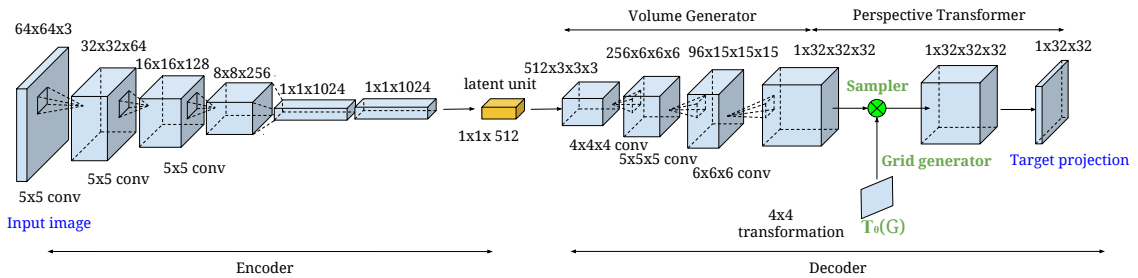


Figure 5.2: Illustration of network architecture.

Network Architecture. As shown in Figure 5.2, our encoder-decoder network has three components: a 2D convolutional encoder, a 3D up-convolutional decoder and a perspective transformer networks. The 2D convolutional encoder consists of 3 convolution layers, followed by 3 fully-connected layers (convolution layers have 64, 128 and 256 channels with fixed filter size of 5×5 ; the three fully-connected layers have 1024, 1024 and 512 neurons, respectively). The 3D convolutional decoder consists of one fully-connected layer, followed by 3 convolution layers (the fully-connected layer have $3 \times 3 \times 3 \times 512$ neurons; convolution layers have 256, 96 and 1 channels with filter size of $4 \times 4 \times 4$, $5 \times 5 \times 5$ and $6 \times 6 \times 6$). For perspective transformer networks, we used perspective transformation to project 3D volume to 2D silhouette where the transformation matrix is parametrized by 16 variables and sampling grid is set to $32 \times 32 \times 32$. We use the same network architecture for all the experiments.

Implementation Details. We used the ADAM (*Kingma and Ba, 2015*) solver for stochastic optimization in all the experiments. During the pre-training stage (for encoder), we used mini-batch of size 32, 32, 8, 4, 3 and 2 for training the RNN-1, RNN-2, RNN-4, RNN-8, RNN-12 and RNN-16 as used in *Yang et al. (2015)*. We used the learning rate 10^{-4} for RNN-1, and 10^{-5} for the rest of recurrent neural networks. During the fine-tuning stage (for volume decoder), we used mini-batch of size 6 and learning rate 10^{-4} . For each object in a mini-batch, we include projections from all 24 views as supervision. The models including the perspective transformer nets are implemented using Torch (*Collobert et al., 2011*). To download the code, please refer to the project webpage: <http://goo.gl/YEJ2H6>.

Experimental Design. As mentioned in the formulation, there are several variants of the model depending on the hyper-parameters of learning objectives λ_{proj} and λ_{vol} . In the experimental section, we denote the model trained with projection loss only, volume loss only, and combined loss as **PTN-Proj** (PR), **CNN-Vol** (VO), and **PTN-Comb** (CO), respectively.

In the experiments, we address the following questions: (1) Will the model trained with combined loss achieve better single-view 3D reconstruction performance over model trained on volume loss only (PTN-Comb vs. CNN-Vol)? (2) What is the performance gap between the models with and without ground-truth volumes (PTN-Comb vs. PTN-Proj)? (3) How do the three models generalize to instances from unseen categories which are not present in the training set? To answer the questions, we trained the three models under two experimental settings: single category and multiple categories.









































































Input	GT (310)	GT (130)	PR (310)	PR (130)	CO (310)	CO (130)	VO (310)	VO (130)
								
								
								
								
								
								
								
								

Figure 5.3: Single-class results. GT: ground truth, PR: PTN-Proj, CO: PTN-Comb, VO: CNN-Vol (Best viewed in digital version. Zoom in for the 3D shape details). The angles are shown in the parenthesis. Please also see more examples and video animations on the project webpage.

Table 5.1: Prediction IU using the models trained on **chair** category. Below, **chair** corresponds to the setting where each object is observable with full azimuth angles, while **chair-N** corresponds to the setting where each object is only observable with a narrow range (subset) of azimuth angles.

Method / Evaluation Set	chair		chair-N	
	training	test	training	test
PTN-Proj:single (no vol. supervision)	0.5712	0.5027	0.4882	0.4583
PTN-Comb:single (vol. supervision)	0.6435	0.5067	0.5564	0.4429
CNN-Vol:single (vol. supervision)	0.6390	0.4983	0.5518	0.4380
NN search (vol. supervision)	—	0.3557	—	0.3073

5.4.1 Training on a single category

We select **chair** category as the training set for single category experiment. For model comparisons, we first conduct quantitative evaluations on the generated 3D volumes from the test set single-view images. For each instance in the test set, we generate one volume per view image (24 volumes generated in total). Given a pair of ground-truth volume and our generated volume (threshold is 0.5), we computed its intersection-over-union (IU) score and the average IU score is calculated over 24 volumes of all the instances in the test set. In addition, we provide a baseline method based on nearest neighbor (NN) search. Specifically, for each of the test image, we extract VGG feature from **fc6** layer (4096-dim vector) (*Simonyan and Zisserman, 2014*) and retrieve the nearest training example using Euclidean distance in the feature space. The ground-truth 3D volume corresponds to the nearest training example is naturally regarded as the retrieval result.

As shown in Table 5.1, the model trained without volume supervision (projection loss) performs as good as model trained with volume supervision (volume loss) on the **chair** category (testing set). In addition to the comparisons of overall IU, we measured the view-dependent IU for each model. As shown in Figure 5.4, the average prediction error (mean IU) changes as we gradually move from the first view to the last view (15° to 360°). For visual comparisons, we provide a side-by-side analysis for each of the three models we trained. As shown in Figure 5.3, each row shows

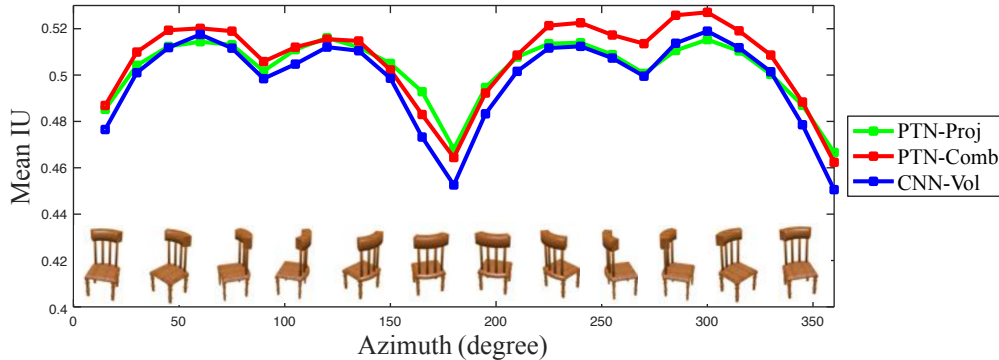


Figure 5.4: View-dependent IU. For illustration, images of a sample chair with corresponding azimuth angles are shown below the curves. For example, 3D reconstruction from 0° is more difficult than from 30° due to self-occlusion.

an independent comparison. The first column is the 2D image we used as input of the model. The second and third column show the ground-truth 3D volume (same volume rendered from two views for better visualization purpose). Similarly, we list the model trained with projection loss only (PTN-Proj), combined loss (PTN-Comb) and volume loss only (CNN-Vol) from the fourth column up to the ninth column. The volumes predicted by PTN-Proj and PTN-Comb faithfully represent the shape. However, the volumes predicted by CNN-Vol do not form a solid chair shape in some cases.

Training with partial views. We also conduct control experiments where each object is only observable from a narrow range of azimuth angles (e.g., 8 out of 24 views such as $0^\circ, 15^\circ, \dots, 105^\circ$). As shown in Table 5.1 (last two columns), performances of all three models drop a little bit but the conclusion is similar: the proposed network (1) learns better 3D shape with projection regularization and (2) is capable of learning the 3D shape by providing 2D observations only.

Table 5.2: Prediction IU using the models trained on large-scale datasets.

Test Category	airplane	bench	dresser	car	chair	display	lamp
PTN-Proj:multi	0.5556	0.4924	0.6823	0.7123	0.4494	0.5395	0.4223
PTN-Comb:multi	0.5836	0.5079	0.7109	0.7381	0.4702	0.5473	0.4158
CNN-Vol:multi	0.5747	0.5142	0.6975	0.7348	0.4451	0.5390	0.3865
NN search	0.5564	0.4875	0.5713	0.6519	0.3512	0.3958	0.2905

Test Category	loudspeaker	rifle	sofa	table	telephone	vessel	
PTN-Proj:multi	0.5868	0.5987	0.6221	0.4938	0.7504	0.5507	
PTN-Comb:multi	0.5675	0.6097	0.6534	0.5146	0.7728	0.5399	
CNN-Vol:multi	0.5478	0.6031	0.6467	0.5136	0.7692	0.5445	
NN search	0.4600	0.5133	0.5314	0.3097	0.6696	0.4078	

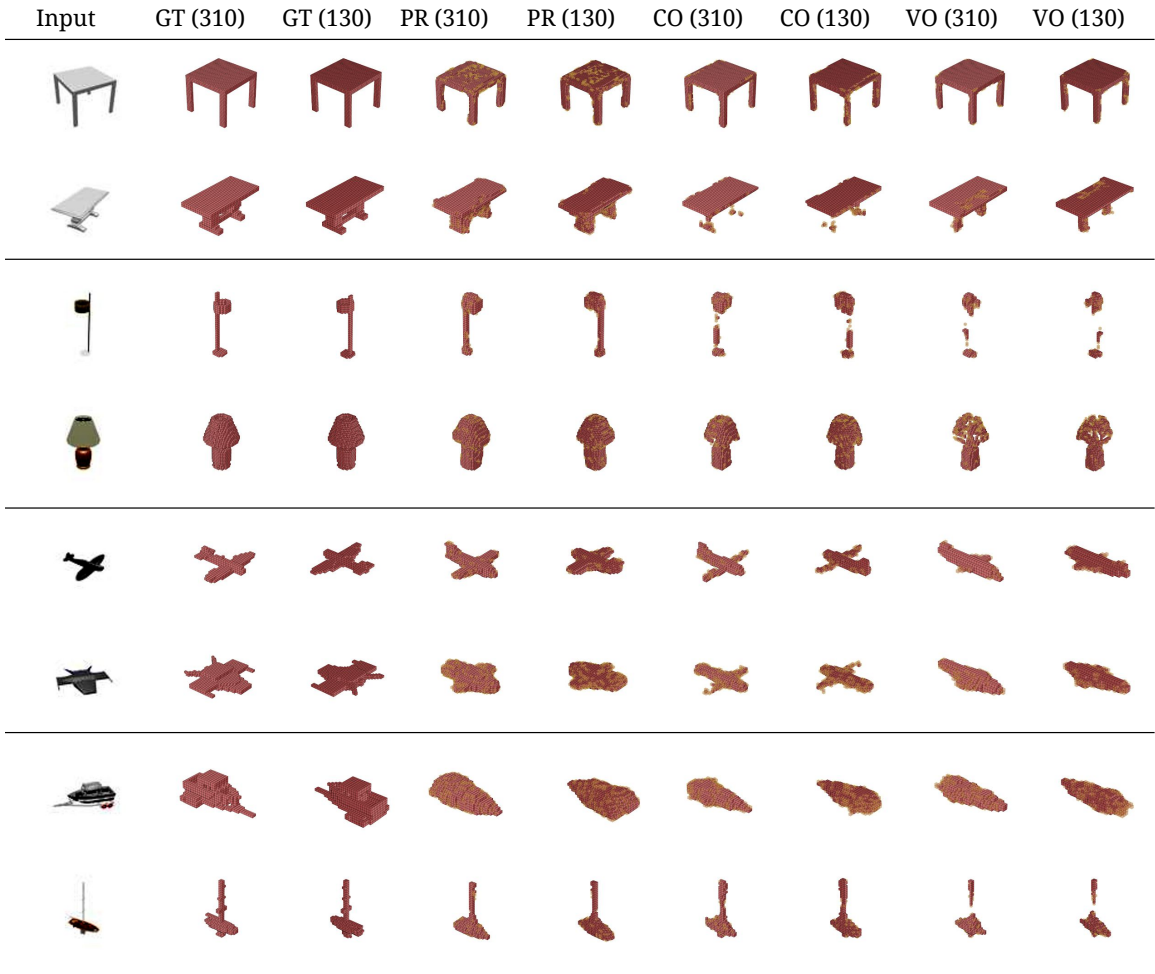


Figure 5.5: Multiclass results. GT: ground truth, PR: PTN-Proj, CO: PTN-Comb, VO: CNN-Vol (Best viewed in digital version. Zoom in for the 3D shape details). The angles are shown in the parenthesis. Please also see more examples and video animations on the project webpage.

Table 5.3: Prediction IU in out-of-category tests.

Method / Test Category	bed	bookshelf	cabinet	motorbike	train
PTN-Proj:single	0.1801	0.1707	0.3937	0.1189	0.1550
PTN-Comb:single	0.1507	0.1186	0.2626	0.0643	0.1044
CNN-Vol:single	0.1558	0.1183	0.2588	0.0580	0.0956
PTN-Proj:multi	0.1944	0.3448	0.6484	0.3216	0.3670
PTN-Comb:multi	0.1647	0.3195	0.5257	0.1914	0.3744
CNN-Vol:multi	0.1586	0.3037	0.4977	0.2253	0.3740

5.4.2 Training on multiple categories

We conducted multiclass experiment using the same setup in the single-class experiment. For multi-category experiment, the training set includes 13 major categories: airplane, bench, dresser, car, chair, display, lamp, loudspeaker, rifle, sofa, table, telephone and vessel. We preserved 20% of instances from each category as testing data. As shown in Table 5.2, the quantitative results demonstrate (1) model trained with combined loss is superior to volume loss in most cases and (2) model trained with projection loss perform as good as volume/combined loss. From the visualization results shown in Figure 5.5, all three models predict volumes reasonably well. There is only subtle performance difference in object part such as the wing of airplane.

5.4.3 Out-of-Category Tests

Ideally, an intelligent agent should have the ability to generalize the knowledge learned from previously seen categories to unseen categories. To this end, we design out-of-category tests for both models trained on a single category and multiple categories, as described in Section 5.4.1 and Section 5.4.2, respectively. We select 5 unseen categories from ShapeNetCore: bed, bookshelf, cabinet, motorbike and train for out-of-category tests. Here, the two categories cabinet and train are relatively easier than other categories since there might be instances in the training set with similar shapes (e.g., dresser, vessel, and airplane). But the bed, bookshelf and motorbike can be considered as completely novel categories in terms of shape.

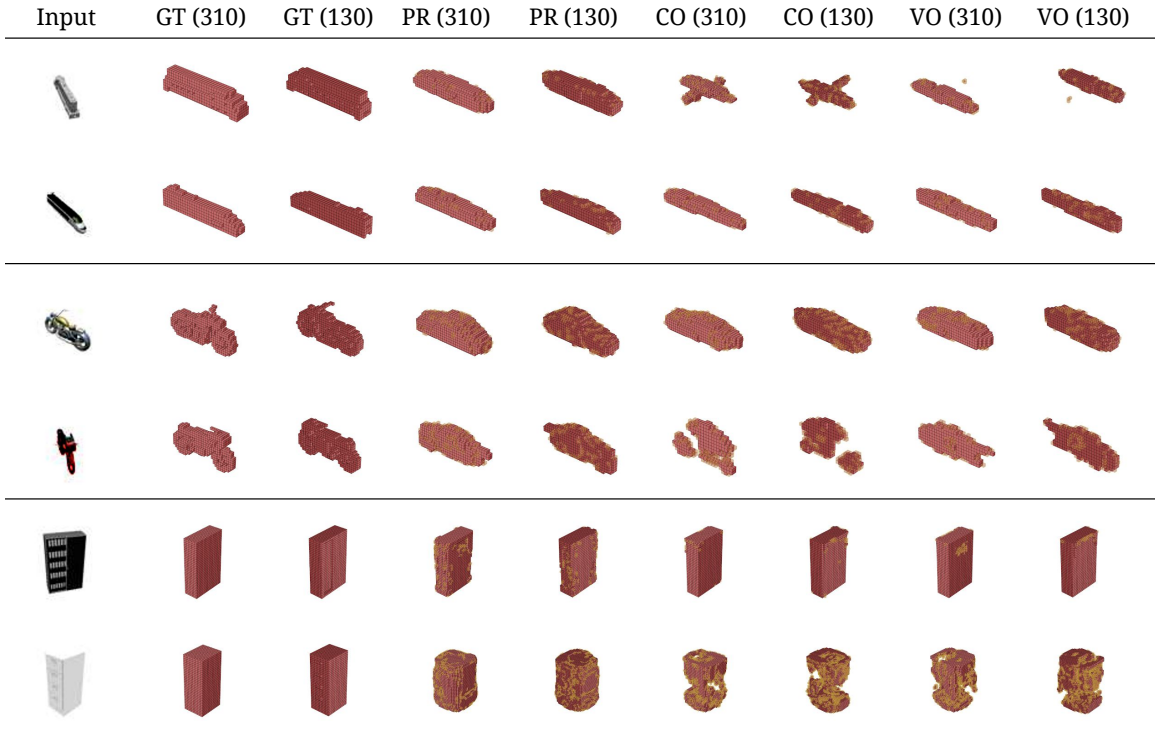


Figure 5.6: Out-of-category results. GT: ground truth, PR: PTN-Proj, CO: PTN-Comb, VO: CNN-Vol (Best viewed in digital version. Zoom in for the 3D shape details). The angles are shown in the parenthesis. Please also see more examples and video animations on the project webpage.

We summarized the quantitative results in Table 5.3. Surprisingly, the model trained on multiple categories still achieves reasonably good overall IU. As shown in Figure 5.6, the proposed projection loss generalizes better than model trained using combined loss or volume loss on **train**, **motorbike** and **cabinet**. The observations from the out-of-category tests suggest that (1) generalization from a single category is very challenging, but training from multiple categories can significantly improve generalization, and (2) the projection regularization can help learning a robust representation for better generalization on unseen categories.

5.5 Discussions

In this chapter, we investigate the problem of single-view 3D shape reconstruction from a learning agent’s perspective. By formulating the learning procedure as the interaction between 3D shape and 2D observation, we propose to learn an encoder-decoder network which takes advantage of the projection transformation as regularization. Experimental results demonstrate (1) excellent performance of the proposed model in reconstructing the object even without ground-truth 3D volume as supervision and (2) the generalization potential of the proposed model to unseen categories.

CHAPTER VI

Learning Geometry-aware Deep Representation for 6-DOF Grasping

6.1 Introduction

Learning to interact with and grasp objects is a fundamental and challenging problem in robot learning that combines perception, motion planning, and control. The problem is challenging because it not only requires understanding geometry (the global shape of an object, the local surface around the interaction space) but it also requires estimating physical properties, such as weight, density, and friction. Furthermore, it requires invariance to illumination, object location, and viewpoint. To handle this, current data-driven approaches (*Lenz et al.*, 2015; *Pinto and Gupta*, 2016; *Levine et al.*; *Mahler et al.*, 2016, 2017) use hundreds of thousands of examples to learn a solution.

While further scaling may help improve performance of these methods, we postulate shape is core to interaction and that additional shape signals to focus learning will boost performance. The notion of using shape and geometry has been pioneered in grasping research (*Goldfeder et al.*, 2009; *León et al.*; *Bohg and Kragic*, 2010; *Li et al.*, 2016; *Vahrenkamp et al.*, 2016).

Inspired by these approaches, we propose the concept of a deep **geometry-aware**

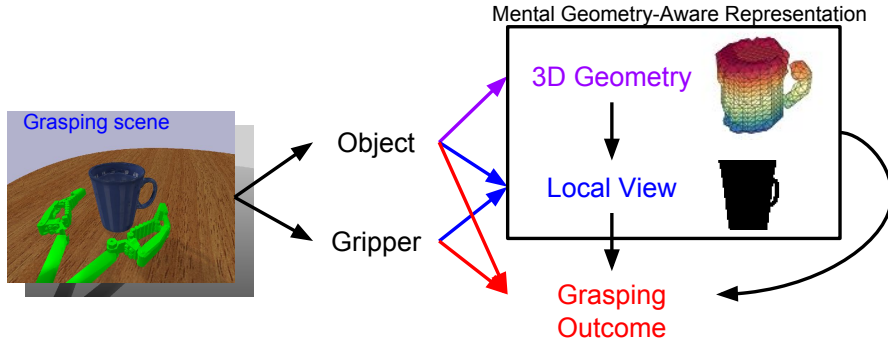


Figure 6.1: Learning grasping interactions from demonstrations with deep geometry-aware representations. First, we learn to build mental geometry-aware representation by reconstructing the 3D scene with 2.5D training data. Second, we learn to predict grasping outcome with its internal geometry-aware representation.

representation (e.g., (Wu et al., 2015; Girdhar et al., 2016; Choy et al., 2016; Wu et al., 2016b; Maturana and Scherer, 2015; Rezende et al., 2016; Yan et al., 2016b; Tulsiani et al., 2017; Godard et al., 2016; Gadelha et al., 2016)) for grasping. Key to our approach is that we first build a mental representation by *recognizing* and *reconstructing* the 3D geometry of the scene from RGBD input, as demonstrated in Figure 6.1. With the built-in 3D geometry-aware representation, we can hallucinate a local view of the object’s geometric surface from the gripper perspective that will be directly useful for grasping interaction. In contrast with black-box models that do not have explicit notion of 3D geometry and prior shape-based grasping approaches, our approach has the following features: (1) it performs 3D shape reconstruction as an auxiliary task; (2) it hallucinates the local view using a learning-free physical projection operator; and (3) it explicitly reuses the learned geometry-aware representation for grasping outcome prediction.

In this work, we design an end-to-end deep geometry-aware grasping network for learning this representation. Our geometry-aware network has two components: a shape generation network and a grasping outcome prediction network. The shape generation network learns to recognize and reconstruct the 3D geometry of the scene with an image encoder and voxel decoder. The image encoder transforms the RGBD

input into a high-level geometry representation that involves shape, location, and orientation of the object. The voxel decoder network takes in the geometry representation and outputs the occupancy grid of the object. To further hallucinate the local view from gripper perspective, we propose a novel learning-free image projection layer similar to *Yan et al. (2016b)*; *Rezende et al. (2016)*. Building upon the shape generation network, our grasping outcome prediction network learns to produce a grasping outcome (e.g., success or failure) based on the action (i.e. gripper pose), the current visual state (e.g., object and gripper), and the learned geometry-aware 3D representation. Unlike our end-to-end multi-objective learning framework, existing data-driven grasping pipelines (*Pinto and Gupta, 2016*; *Mahler et al., 2016, 2017*) can be viewed as models without a shape generation component. They require either an additional camera to capture the global object shape or extra processing steps, such as object detection and patch alignment. Furthermore, these methods learn over a constrained grasp space, typically either 3-DOF or 4-DOF. We relax this constraint to learn fully generalized 6-DOF grasp poses.

We have built a large database consisting of 101 everyday objects with around 150K grasping demonstrations in Virtual Reality with both human and augmented synthetic interactions. For each object, we collect 10-20 grasping attempts with a parallel jaw gripper from right-handed users. For each attempt, we record a pre-grasping status which includes the location and orientation of the object and gripper, as well as the grasping outcome (e.g., success or failure given if the object is between the gripper fingers after closing and lifting). To acquire sufficient data for learning, we generate additional synthetic data by perturbing the gripper location and orientation from human demonstrations using PyBullet (*Coumans et al.*). More information about our geometry-aware grasping project can be found at <https://goo.gl/gPzPhm>.

Our main contributions are summarized below:

- To best of our knowledge, we are presenting for the first time a method to learn

a 6-DOF deep grasping neural network from RGBD input.

- We build a database with rich visual sensory data and grasping annotations with a virtual reality system and propose a data augmentation strategy for effective learning with only modest amount of human demonstrations.
- We demonstrate that the proposed geometry-aware grasping network is able to learn the shape as well as grasping outcome significantly better than models without notion of geometry.
- We demonstrate that the proposed model has advantages in guiding grasping exploration and achieves better generalization to novel viewpoints and novel object instances.

6.2 Related Work

A common approach for robotic grasping is to detect the optimal grasping location from 2D or 2.5D visual inputs (RGB or RGBD images, respectively) (*Saxena et al.*, 2008; *Montesano and Lopes*, 2012; *Lenz et al.*, 2015; *Pinto and Gupta*, 2016; *Gualtieri et al.*, 2016; *Kopicki et al.*, 2016; *Osa et al.*, 2016). Earlier work (*Saxena et al.*, 2008; *Montesano and Lopes*, 2012) studied the planar grasping problem using visual features extracted from 2D sensory input and adopted logistic regression for fitting optimal grasping location with visual features. *Lenz et al.* (2015) proposed a two-step detection pipeline (object detection and grasping part detection) with deep neural networks. *Pinto and Gupta* (2016) built a robotic system for learning grasping from large-scale real-world trial-and-error experiments. In this work, a deep convolutional neural network was trained on 700 hours of robotic grasping data collected from the system.

Fine-grained grasping planning and control often involves 3D modeling of object shape, modeling dynamics of robot hands, and local surface modeling (*Goldfeder*

et al., 2009; *León et al.*; *Johns et al.*, 2016; *Varley et al.*, 2016; *Li et al.*, 2016; *Vahrenkamp et al.*, 2016; *Mahler et al.*, 2016, 2017). Some work focused on analytic modeling of robotic grasps with known object shape information (*Goldfeder et al.*, 2009; *León et al.*). *Varley et al.* (2016) proposed a shape completion model that reconstructs the 3D occupancy grid for robotic grasping from partial observations, where ground-truth 3D occupancy grid is used during model training. In comparison, our approach does not require full 3D volume supervision for training (e.g., occupancy grid). Similar to our work, *Bohg and Kragic* (2010) use a learned shape-context to help predict grasps. Unlike their work, we use the shape to build a virtual global geometric representation along with a local gripper centric model to sequentially propose and evaluate grasp proposals. *Li et al.* (2016) investigated the hand pose estimation in robotic grasping by decoupling contact points and hand configuration with parametrized object shape. Building upon the compositional aspect of everyday objects, *Vahrenkamp et al.* (2016) proposed a part-based model for robotic grasping that has better generalization to novel object. Very recently, effort was also made in building DexNet (*Mahler et al.*, 2016, 2017), a large-scale point cloud database for planar grasping (from top-down). In addition to general robotic grasping, several recent work investigated the semantic or task-specific grasping (*Dang and Allen*, 2014; *Katz et al.*, 2014; *Nikandrova and Kyrki*, 2015).

In contrast to existing learning frameworks applied to robotic grasping (either top-down grasping or side-grasping), our approach features (1) providing a method to learn a 6D grasping network from RGBD input (2) an end-to-end deep learning framework for generative 3D shape modeling and leveraging it for predictive 6D grasping interaction, and (3) learning-free projection layer that links the 2D observations with 3D object shape which allows for learning the shape representation without explicit 3D volume supervision.

6.3 Multi-objective framework with geometry-aware representation

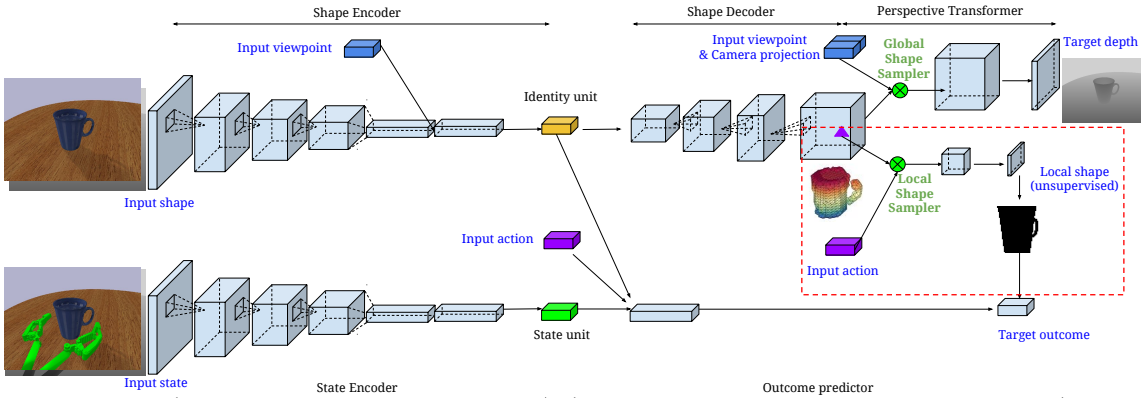


Figure 6.2: Illustration of DGGN (deep geometry-aware grasping network). Our DGGN has a shape generation network and an outcome prediction network. The shape generation network has a 2D CNN encoder, 3D CNN decoder, and a global sampling layer (detailed in Sec. 6.3.2). Our outcome prediction network has a 2D CNN encoder, a local sampling layer (detailed in Sec. 6.3.4), and a fully-connected prediction network.

In this section, we develop a multi-objective learning framework that performs 3D shape generation and grasping outcome prediction.

6.3.1 Learning generative geometry-aware representation from RGBD input

Being able to *recognize* and *reconstruct* the 3D geometry given RGBD input is a very important step during grasping planning. In our formulation, we propose a reconstruction of a 3D occupancy grid (Wu et al., 2015; Girdhar et al., 2016; Choy et al., 2016; Wu et al., 2016b; Rezende et al., 2016; Yan et al., 2016b; Tulsiani et al., 2017; Godard et al., 2016; Gadelha et al., 2016) that encodes the shape, location, and orientation of the object as our geometry-aware representation. Previous work generate normalized 3D occupancy grids centered at the origin. Our formulated geometry-aware representation differs in that (1) it takes location and orientation

into consideration (the orientation of a novel object is usually undefined); (2) it is invariant to camera viewpoint and distance (we obtain the same representation from arbitrary camera setting).

Given an RGBD input \mathcal{I} and a corresponding 3D occupancy grid \mathbf{V} , the task is to learn a functional mapping $f^V : \mathcal{I} \rightarrow \mathbf{V}$. Simply following this formulation, previous work (*Wu et al.*, 2015; *Girdhar et al.*, 2016; *Choy et al.*, 2016; *Wu et al.*, 2016b; *Maturana and Scherer*, 2015) that use 3D supervision obtained reasonable quality in generating normalized 3D volumes by using thousands of shape instances. However, in our problem setting, these methods would require even more data considering the entangled factors from shape, location, and orientation.

6.3.2 Depth supervision with in-network projection layer

Recent breakthroughs in reconstructing 3D geometry with 2D supervision (*Rezende et al.*, 2016; *Yan et al.*, 2016b; *Tulsiani et al.*, 2017; *Zhou et al.*, 2017b; *Godard et al.*, 2016; *Gadelha et al.*, 2016; *Fan et al.*, 2017; *Tung et al.*, 2017) suggest that (1) the quality of reconstructed 3D geometry is as good as previous work with 3D supervision; (2) the learned representation generalizes better to novel settings than previous work with 3D supervision; and (3) learning becomes more efficient with 2D supervision. Inspired by these findings, we tackle the 3D reconstruction in a weakly supervised manner without explicit 3D shape supervision. In *Yan et al.* (2016b), an in-network projection layer is introduced for 3D shape learning from 2D masks (e.g. 2D silhouette of object). Unfortunately, 2D silhouette is usually insufficient supervision signal to reconstruct objects with concave 3D parts (e.g., containers). For these reasons, we chose to use a depth signal in our shape reconstruction. Additionally, RGBD sensors are commonly available in most robot platforms.

To enable depth supervision in our shape generation component, we propose a novel in-network OpenGL projection operator that utilizes a 2D depth map \mathcal{D} as

supervision signal for learning to reconstruct the 3D geometry. We formulate the projection operation by $f^D : \mathbf{V} \times \mathbf{P} \rightarrow \mathcal{D}$ that transforms a 3D shape into a 2D depth map with the camera transformation matrix \mathbf{P} . Here, the camera transformation matrix decomposes as $\mathbf{P} = \mathbf{K}[\mathbf{R}; \mathbf{t}]$, where \mathbf{K} is the camera intrinsic matrix, \mathbf{R} is the camera rotation matrix, and \mathbf{t} is the camera translation vector. In our implementation, we also use a 2D silhouette as an object mask \mathcal{M} for learning. Empirically, this additional objective makes the learning stable and efficient.

Following the OpenGL camera transformation standard, for each point $\mathbf{p}^s = (x^s, y^s, z^s, 1)$ in 3D world frame, we compute the corresponding point $\mathbf{p}^n = (x^n, y^n, z^n, 1)$ in the normalized device coordinate system ($-1 \leq x^n, y^n, z^n \leq 1$) using the transformation: $\mathbf{p}^n \sim \mathcal{P}\mathbf{p}^s$. Here, the conversion from depth buffer z^n to real depth z^e is given by $z^e = f^e(z^n) = -1/(\alpha * z^n + \beta)$ where $\alpha = \frac{Z_{near} - Z_{far}}{2Z_{near}Z_{far}}$ and $\beta = \frac{Z_{near} + Z_{far}}{2Z_{near}Z_{far}}$. Here, Z_{far} and Z_{near} represents the far and near clipping planes of the camera.

Similar to the “transformer networks” proposed in *Yan et al. (2016b)*; *Jaderberg et al. (2015)*, our depth projection can be seen as: (1) performing dense sampling from input volume (in the 3D world frame) to output volume (in normalized device coordinates); and (2) flattening the 3D spatial output across one dimension. Again, j -th point (x_j^n, y_j^n, z_j^n) in output volume $\mathbf{U} \in \mathbb{R}^{H' \times W' \times D'}$ (j -th point is indexed by $[n', m', l']$ in the volume space) and corresponding point (x_j^s, y_j^s, z_j^s) in input volume $\mathbf{V} \in \mathbb{R}^{H \times W \times D}$ are related by the transformation matrix \mathbf{P} . Here, (W, H, D) and (W', H', D') are the width, height, and depth of the input and output volume, respectively. We define the dense sampling step and channel-wise flattening step as

follows:

$$\begin{aligned}
 U[n', m', l'] &= \sum_{n=1}^H \sum_{m=1}^W \sum_{l=1}^D V[n, m, l] \max(0, 1 - |x_j^s - m|) \\
 &\quad \max(0, 1 - |y_j^s - n|) \max(0, 1 - |z_j^s - l|) \\
 \hat{\mathcal{M}}[n', m'] &= \max_{l'} U[n', m', l'] \\
 \hat{\mathcal{D}}[n', m'] &= \begin{cases} Z_{far}, & \text{if } \hat{\mathcal{M}}[n', m'] = 0 \\ f^e(\frac{2l'}{D'} - 1), & \\ \text{where } l' = \arg \min_{l'} (U[n', m', l'] > 0.5) & \\ Z_{near}, & \text{otherwise} \end{cases}
 \end{aligned} \tag{6.1}$$

In our implementation, we pre-computed the actual depth $f^e(\frac{2l'}{D'} - 1)$ given the difficulty that arg min is not back-propagatable. As we will see in the following section, the network will be trained to match these predictions $\hat{\mathcal{M}}$ and $\hat{\mathcal{D}}$ to the ground-truth \mathcal{M} and \mathcal{D} . Please note that our in-network projection layer is *learning-free* as it implements the exact ray-tracing algorithm without extra free parameters involved. We note that the concept of depth projection is also explored in some very recent work (*Wu et al., 2017; Tewari et al., 2017; Zhou et al., 2017b*), but their implementations are not exactly the same as our OpenGL projection layer in Eq. 6.1.

6.3.3 Viewpoint-invariant geometry-aware representation with multi-view supervision

Learning to reconstruct 3D geometry from single-view RGBD sensory input is a challenging task in computer vision due to shape ambiguity. We adopt the shape consistency learning that enforces viewpoint-invariance across multi-view observations (*Choy et al., 2016; Yan et al., 2016b; Tulsiani et al., 2017*). More specifically,

we (1) use the averaged identity units from multiple viewpoints as input to shape decoder network and (2) provide multiple projections for supervising the 3D shape reconstruction during training. Such shape consistency learning encourages an image taken from one viewpoint sharing the same representation with the image taken from another viewpoint. At testing time, we only provide RGBD input from single viewpoint. Given a series of n observations $\mathcal{I}_1, \mathcal{I}_2, \dots, \mathcal{I}_n$ of the scene, the 3D reconstruction can be formulated as $f^V : \{\mathcal{I}_i\}_{i=1}^n \rightarrow \mathbf{V}$. Similarly, the projection operator from i -th viewpoint is $f^D : \mathbf{V} \times \mathbf{P}_i \rightarrow \mathcal{D}_i$, where \mathcal{D}_i and \mathbf{P}_i are the depth and camera transformation matrix from corresponding viewpoint, respectively. Finally, we define the shape reconstruction loss \mathcal{L}^{shape} in Eq. 6.2.

$$\mathcal{L}_\theta^{shape} = \lambda_{\mathcal{D}} \sum_{i=1}^n \mathcal{L}_\theta^{depth}(\hat{\mathcal{D}}_i, \mathcal{D}_i) + \lambda_{\mathcal{M}} \sum_{i=1}^n \mathcal{L}_\theta^{mask}(\hat{\mathcal{M}}_i, \mathcal{M}_i) \quad (6.2)$$

Here, $\lambda_{\mathcal{D}}$ and $\lambda_{\mathcal{M}}$ are the constant coefficients for the depth and mask prediction terms, respectively.

6.3.4 Learning predictive grasping interaction with geometry-aware representation.

As demonstrated in previous work (*Oh et al., 2015; Finn et al., 2016a; Dosovitskiy and Koltun, 2016; Yang et al., 2015; Pinto et al., 2016*) that learn interactions from demonstrations, *prediction* of the future state can be a metric for understanding the physical interaction. In our grasping setting, we define the RGBD input \mathcal{I} as current state, the 6D pre-grasping parameters \mathbf{a} (position and orientation of the parallel jaw gripper) as action, and the grasping outcome l (e.g., binary label representing a successful grasp or not) as future state. The future prediction task can be solved by learning a functional mapping $f_{baseline}^l : \mathcal{I} \times \mathbf{a} \rightarrow l$. We refer to this method as a baseline grasping interaction prediction model, which has been a basis of several

recent state-of-the-art grasping methods using deep learning (e.g., (Lenz et al., 2015; Levine et al.; Mahler et al., 2017)). These work managed to learn such mapping with either (a) millions of randomly generated grasps, (b) additional view from eye/hand perspective, or (c) additional processing steps such as object detection and image alignment.

In comparison, our geometry-aware model is an end-to-end architecture which constrains its prediction with geometry information. As we learn to reconstruct the 3D geometry, we argue that the *local surface view* (typically from a wrist camera perspective) can be *directly inferred from our viewpoint-invariant geometry-aware representation* $\hat{\mathcal{I}}^{local} = f^D(\hat{\mathbf{V}}, \mathbf{P}(\mathbf{a}))$, where $\hat{\mathbf{V}} = f^V(\mathcal{I})$. Here, we treat the gripper as a virtual camera with the transformation matrix $\mathbf{P}(\mathbf{a})$ with its world-space coordinates given by the 6D pre-grasping parameters \mathbf{a} . In addition to the *local view*, our geometry-aware representation provides a *global view* of the scene \mathbf{V} that takes a shape prior, location, and orientation of object into consideration. Finally, given a current observation \mathcal{I} , proposed action \mathbf{a} , and inferred 3D shape representation \mathbf{V} , we fit a functional mapping $f_{geometry-aware}^l : \mathcal{I} \times \mathbf{a} \times \mathbf{V} \rightarrow l$, where l is the binary outcome.

6.3.5 DGGN: Deep geometry-aware grasping network.

To implement the two components proposed in the previous sections, we introduce **DGGN (deep geometry-aware grasping network)** (see Figure 6.2), composed of a shape generation network and an outcome prediction network. The shape generation network has a 2D convolutional shape encoder and a 3D deconvolutional shape decoder followed by a global projection layer. Our shape encoder network takes RGBD images of resolution 128×128 and corresponding 4-by-4 camera view matrices as input; the network outputs identity units as an intermediate representation. Our shape decoder is a 3D deconvolutional neural network that outputs voxels at

a resolution of $32 \times 32 \times 32$. We implemented the projection layer (given camera view and projection matrices) that transforms the voxels back into foreground object silhouettes and depth maps at an input resolution (128×128). Here, the purpose of generative pre-training is to learn viewpoint invariant units (e.g., object identity units) through object segmentation and depth prediction. The outcome prediction network has a 2D convolutional state encoder and a fully connected outcome predictor with an additional local shape projection layer. Our state encoder takes RGBD input (the pre-grasp scene) of resolution 128×128 and corresponding actions (position and orientation of the gripper end-effector) and outputs state units as intermediate representation. Our outcome predictor takes both current state (e.g., the pre-grasp scene and gripper action) and geometry features (e.g., viewpoint-invariant global and local geometry from the local projection layer) into consideration. Note that the local dense-sampling transforms the surface area around the gripper fingers into a foreground silhouette and a depth map at resolution 48×48 .

6.4 Experiments

This section describes our data collection and augmentation process, as well as experimental evaluation on grasping outcome prediction and grasping trials.

6.4.1 Dataset collection

Human demonstrations in VR We collected grasping demonstrations on seven categories of objects, which include a total of 101 everyday objects. To collect grasping demonstrations, we set up the HTC Vive system in Virtual Reality (VR) and assign target objects randomly to five right-handed users (three males and two females). In total, 1597 human grasps are demonstrated, with an average of 15 grasps per object (with lowest and highest number of grasps at 7 and 39 for a plate and a wine glass, respectively). We randomly split 101 objects into three sets (e.g., training, validation

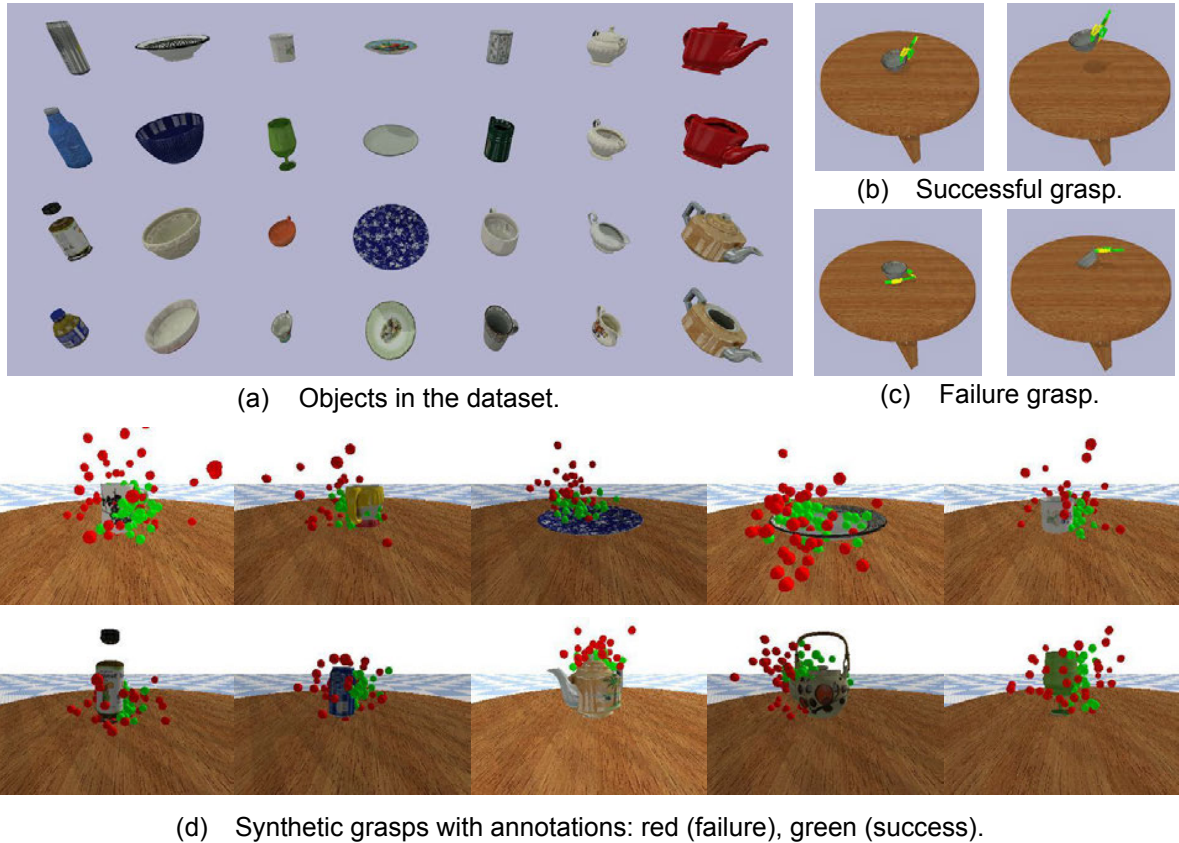


Figure 6.3: Illustrations of our VR-Grasping-101 dataset.

and testing) and make sure each set covers the seven categories (70% for training, 10% for validation and 20% for testing).

Data augmentation In order to collect sufficient grasping demonstrations for model training and evaluation, we generate synthetic grasps by perturbing the human demonstrations using PyBullet (*Coumans et al.*). This significantly helps in increasing the number of grasps by adding perturbations to the demonstrations. In total, we collected 150K grasping demonstrations covering 101 objects. Figure 6.3 illustrates examples of objects in the dataset, successful and unsuccessful grasping trials from human demonstrations, and synthetic grasps (visualized by gripper positions) for successful and unsuccessful trials that were generated by this augmentation process. More details are described in the Appendix.

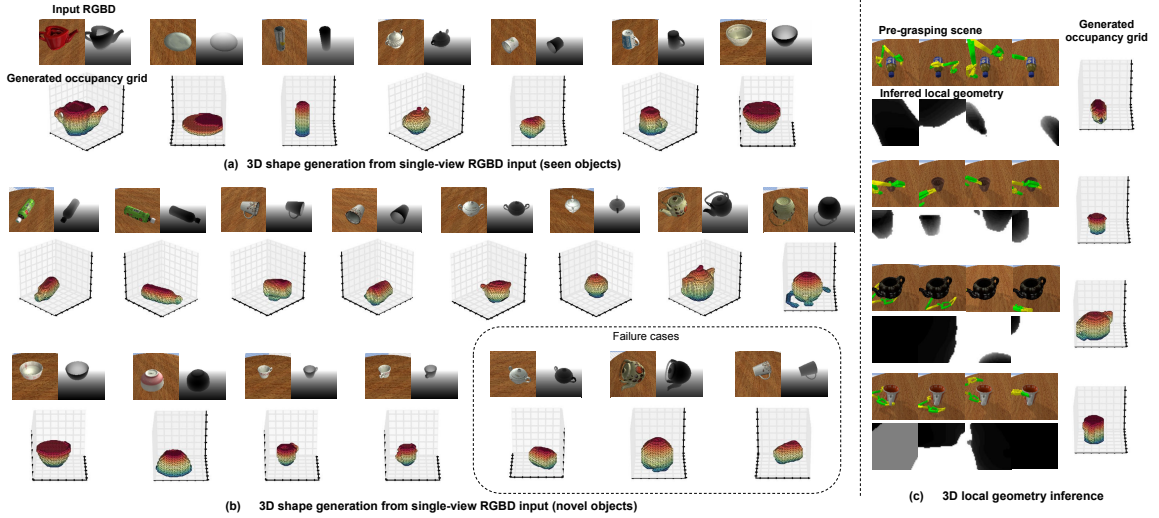


Figure 6.4: Visualization: 3D shape generation from single-view RGBD. (a) The performance on training (seen) objects. (b) The performance on testing (novel) objects. (c) Local geometry inference from generated occupancy grid.

For each demonstration, we take a snapshot of the pre-grasping scene (e.g., before closing the two gripper fingers). by randomly setting the camera at a distance (ranging between 35 centimetres and 45 centimetres). We draw a camera target position from a normal distribution with its mean as the object center and a desired variance (in our experiment, we use 3 centimetres as standard deviation). Furthermore, we set up the camera around the target position from 8 different azimuth angles (with steps of 45 degrees) and adjust the elevation from 4 different angles (e.g., 15, 30, 45, and 60 degrees). Finally, we save a state of the scene without a gripper, which is used for shape pre-training; this will be referred to as the static scene in this chapter. We include only two elevation angles (e.g., 15 and 45 degrees) in the training set while leaving the rest for evaluation.

6.4.2 Implementation details

Deep CNN baseline. We adopt the current data-driven framework as our grasping baseline by removing the shape encoder and shape decoder from our deep geometry-aware grasping model. This baseline can be interpreted as the grasping quality

CNN (Mahler et al., 2017) without an additional view from a top-down camera. We trained the model using the ADAM optimizer with a learning rate of 10^{-5} for 200K iterations and a mini-batch of size of 4. As an ablation study, we added view and static scene as an additional input channel on top of the baseline model but didn't observe significant improvements.

Training DGGN. We adopted a two-stage training procedure: First, we pre-trained the shape generation model (shape encoder and shape decoder) using the ADAM optimizer with a learning rate of 10^{-5} for 400K iterations and a mini-batch of size of 4. In each batch, we sample 4 random viewpoints for the purpose of multi-view supervision in the training time. We observed that this setting led to a more stable shape generation performance compared to single-view training. In addition, we used L_1 loss for foreground depth prediction and L_2 loss for silhouette prediction with coefficients $\lambda_D = 0.5$ and $\lambda_M = 10.0$. In the second stage, we fine-tuned the state encoder and outcome predictor using the ADAM optimizer with a learning rate of $3 * 10^{-6}$ for 200K iterations and a mini-batch of size of 4. We used cross-entropy as our objective function since the grasping prediction is formulated as a binary classification task.

In our experiments, all the models are trained using 20 GPU workers and 32 parameter servers with asynchronous updates. Both baseline and our geometry-aware model adopt convolutional encoder-decoder architecture with residual connections. The bottleneck layer (e.g., the identity unit in the geometry-aware model) is a 768 dimensional vector.

6.4.3 Visualization: 3D shape generation

We evaluate the quality of the shape generation model by visualizing the geometry representations through the shape encoder and decoder network. In our evaluations, we used single-view RGBD input and corresponding camera view matrix as input

Method / Category	bottle	bowl	cup	plate	mug	sugarbowl	teapot	all
baseline CNN (15)	72.81	73.36	73.26	66.92	72.23	70.45	66.13	71.42
our DGGN (15)	78.83	79.32	77.60	68.88	78.25	76.09	73.69	76.55
baseline CNN (45)	71.02	74.16	73.50	63.31	74.23	72.70	64.19	71.32
our DGGN (45)	78.77	80.63	78.06	70.13	79.29	77.52	72.88	77.25

Table 6.1: Grasping Outcome prediction accuracy from seen elevation angles.

Method / Category	bottle	bowl	cup	plate	mug	sugarbowl	teapot	all
baseline CNN (30)	71.15	72.98	71.65	61.90	71.01	70.06	61.88	69.50
DGGN (30)	79.17	77.71	77.23	67.00	75.95	75.06	70.66	75.27
baseline CNN (60)	68.45	73.05	72.50	61.27	74.40	71.30	63.25	70.18
DGGN (60)	77.40	78.52	76.24	68.13	79.39	76.15	70.34	75.76

Table 6.2: Grasping Outcome prediction accuracy from novel elevation angles.

to the network. As shown in Figure 6.4(a), our shape generation model is able to generate a detailed 3D occupancy grid from single-view input without 3D supervision during training. As shown in Figure 6.4(b), our model demonstrates reasonable generalization quality even on novel object instances.

Analysis: local geometry inference via projection. One advantage of our shape generation component is that we can obtain additional local geometry information (see the red-dashed box in Figure 6.2(c)) from our geometry-aware representation. This is the key difference between our work and the related work that require additional camera from the gripper. With 3D geometry as part of the intermediate representation, we hallucinate the local geometry by running a projection from the gripper’s perspective (i.e., simply treat the gripper as another virtual camera). To further understand the advantages of our shape generation component, we visualized the intermediate local geometry projected from generated 3D occupancy grid. As shown in Figure 6.4(c), our shape generation component provides accurate local geometry estimation that is useful for grasping outcome prediction.

Method / Category	bottle	bowl	cup	plate	mug	sugarbowl	teapot	all
baseline CNN + CEM	48.60	64.28	55.44	45.99	61.00	53.97	63.08	55.85
our DGGN + CEM	56.73	68.84	60.31	50.09	67.21	59.87	69.22	61.46
rel. improvement (%)	16.72	7.09	8.77	8.92	10.18	10.92	9.73	10.03

Table 6.3: Grasping planning on novel objects: success rate by optimizing for up to 20 steps.

6.4.4 Model evaluation: Grasping outcome prediction

To evaluate the actual advantages in grasping outcome prediction from our modeling, we computed the average classification accuracy over 30K demonstrations from novel object instances (from testing set) with diverse observation viewpoints. For each human demonstration, we generated 100 synthetic grasps through perturbation (among which 50% of them are success grasps) and computed the average accuracy on 100 grasps (i.e., random guess achieves 50% accuracy). To investigate the model performance due to viewpoint changes, we repeat the evaluation experiment for four different elevation angles (e.g, 15, 30, 45, and 60 degrees). We use parallel computing resources (500 machines) during evaluation and the entire evaluation took about 1 day. The results are summarized in Table 6.1 and Table 6.2. Overall, the deep geometry-aware model consistently outperforms the deep CNN baseline in grasping outcome classification. As we can see, “teapot” and “plate” are comparatively more challenging categories for outcome prediction, since “teapot” has irregular shape parts (e.g., tip and handle) and “plate” has a fairly flat shape. When it comes to novel elevation angles (e.g., compare Table 6.1 and Table 6.2), our deep geometry-aware model is less affected, especially in categories such as “teapot” and “plate” where viewpoint-invariant shape understanding is crucial.

6.4.5 Application: Analysis-by-synthesis grasping planning.

As we improve the classification accuracy over the grasping outcome, a natural question is whether this improvement can be used to guide better grasping planning.

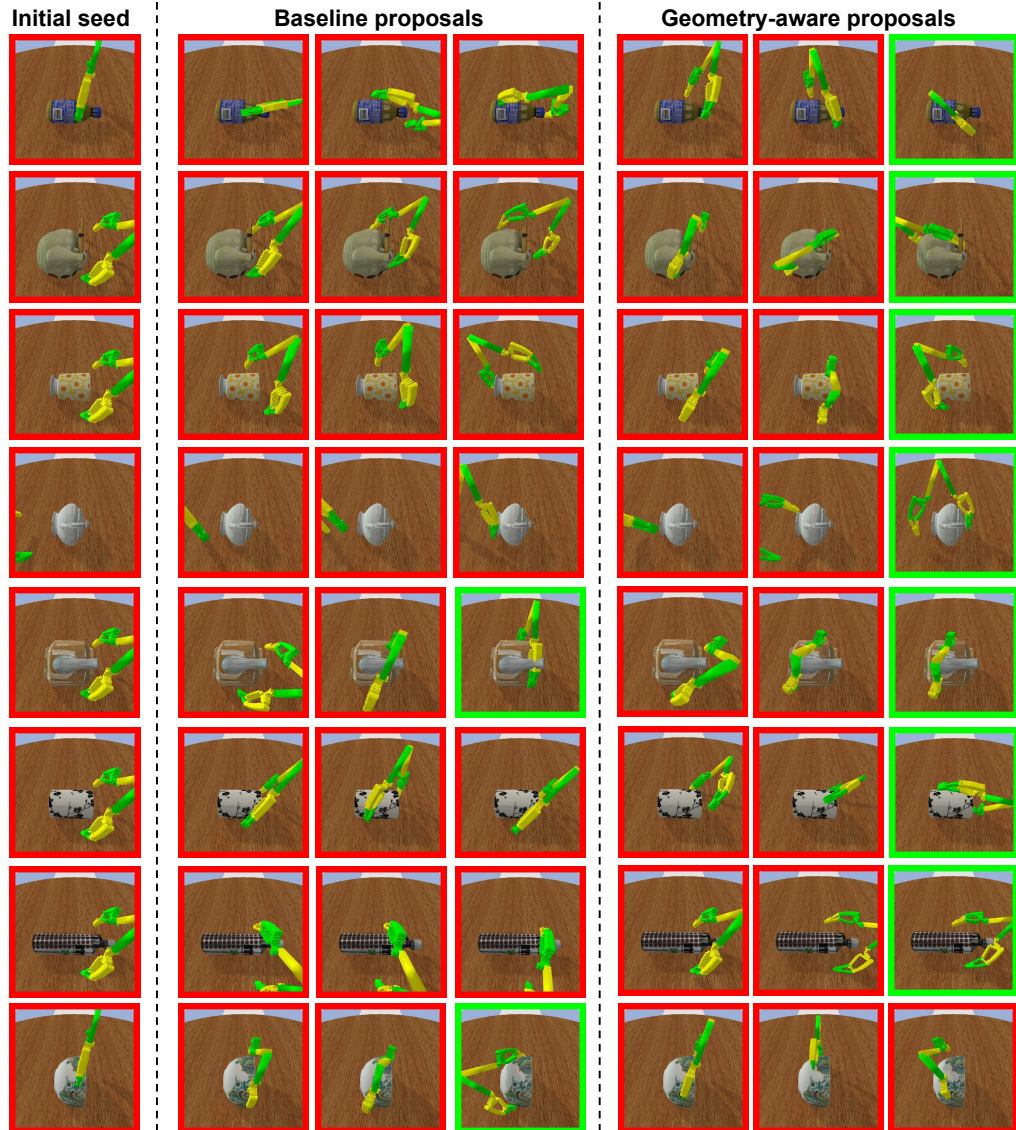


Figure 6.5: Visualization: grasping optimization with CEM based on the grasping prediction output. In each row, we selected three representative steps in grasping optimization (in sequential order from left to right). Red box represents a failure grasp while green box represents a successful grasp.

Given a grasping proposal (defined as target gripper pose) seed, we conducted grasping planning by sequentially adjusting the grasping pose guided by our deep grasping network until a grasp success. In each optimization step, we performed cross-entropy method (CEM) (*Rubinstein and Kroese, 2004; Levine et al.*) as follows. (1) We initialized with a failure grasp in order to force the model to find better grasping

pose. (2) To obtain the gradient direction in the 6D space, we sample 10 random directions and selected the top one based on the score returned by the neural network (output of outcome predictor). We repeat the iterations until success (we set an upper bound of 20 steps). We conducted the same grasping explore evaluation for both the baseline CNN and our deep geometry-aware model. To account for the variations in observation viewpoints and initial seeds, we repeat the evaluation for eight times per testing demonstration in our dataset and reported the average success rate after 20 iterations (marked as failure only if there is no success in 20 steps). As shown in Table 6.3, CEM guided our geometry-aware model performance consistently better than the baseline CNN model. We believe the improved performance comes from the explicit modeling of the 3D geometry as intermediate representation in our deep geometry-aware model. Our model achieved the most significant improvement in the “bottle” category, since a bottle shape is relatively easy to reconstruct. Our improvement in the “bowl” category is less significant, partly due to the difficulty of predicting its concave shape in novel object instances. Figure 6.5 demonstrates example grasping planning trajectories on different objects. The baseline CNN is less robust compared to our deep geometry-aware model, which is more likely to transit from one side of the object to the other side with a clear notion of 3D geometry.

6.5 Discussions

In this work, we studied the problem of learning the grasping interaction with deep geometry-aware representation. We proposed a deep geometry-aware network that performs shape generation as well as grasping outcome prediction with a learning-free physical projection layer. Compared to the CNN baseline, experimental results demonstrated improved performance in outcome prediction thanks to generative shape modeling. Guided by the geometry-aware representation, we obtained better planning via analysis-by-synthesis grasping optimization.

We believe the proposed deep geometry-aware grasping framework has many potentials in advancing robot learning in general. One interesting future direction is to apply the learned geometry-aware representation to perform tasks using other types of hands (e.g., hands with very different kinematics). In addition, we would like to explore some alternative model designs (e.g., learn to grasp without the auxiliary state encoder) such that the learned geometry-aware representation might be easily adapted to other domains (e.g., real robot setup).

CHAPTER VII

Future work

For future directions, I am going to continue the research on controllable image generation and generative structure prediction. As most of the thesis has been focused on learning object-centric representations with specific data domain, I plan to investigate the possible extensions and applications to multi-modal data and generative relational modeling on object-to-object, object-to-scene . For example, one limitation of semantic image manipulation work (mentioned in Chapter VI) is that the semantic bounding boxes are generated by data-driven heuristics or provided by human users. To fully automate the process, I plan to investigate the problem of semantic scene generation using deep neural networks with an emphasis on object-to-object relational modeling. To facilitate generative relational modeling with semantic structures, we consider (1) learning an intermediate 3D representations, as images are 2D projections from 3D world and (2) discovering object relational structures from both image and text. In another example, the human motion prediction work (mentioned in Chapter IV) assumes a single-actor setting without taking environmental factors into consideration. For future work, I plan to investigate the problem of generating human actors in the movie with multi-modal constraints (e.g., objects in the scene, interactions with other actors, or transcripts). Similarly, the motion generation framework is applicable to predicting movements of cars and pedestrian in a modern

city. This can be very useful for improving the motion prediction performance with diverse motion trajectories. I believe learning controllable and structured representations in an online environment (e.g., simulators or interactive platforms), as opposed to learning using offline-generated data can potentially become an interesting research topic in machine learning and AI in the near future.

Finally, adversarial learning becomes a very popular topic in machine learning and privacy. Currently, most adversarial examples are generated by adding pixel-wise perturbations or semantically transforming the image patches. I plan to investigate possible research directions that combine adversarial learning and deep generative models. For example, I plan to aim to explore the impact of semantic manipulation on DNNs by manipulating semantic attributes of images and generate unrestricted adversarial examples. Such semantic based perturbation is more practical and structured compared with pixel-wise manipulation. Such structured adversarial examples with controlled semantic manipulation can shed light on further understanding about vulnerabilities of Deep Neural Networks (DNNs) as well as potential defensive approaches.

APPENDIX

APPENDIX A

A.1 Derivation of disCVAE Objective

We provide a detailed derivation of the objective function for disentangling CVAE (disCVAE). Similarly to the vanilla CVAE, we have x and x_F as input image (full, foreground), g as foreground mask, y as attribute labels, and $z = [z_F, z_B]$ as latent variables (z_F for foreground and z_B for background).

The joint conditional log-likelihood of x , x_F and g given y can be written as follows:

$$\begin{aligned}
 & \log p_\theta(x_F, g, x|y) && \text{(A.1)} \\
 = & \mathbb{E}_{q_\phi(z_F, z_B|x_F, g, x, y)} [\log p_\theta(x_F, g, x|y)] \\
 = & \mathbb{E}_{q_\phi(z_F, z_B|x_F, g, x, y)} [\log p_\theta(x_F, g, x, z_F, z_B|y) - \log p_\theta(z_F, z_B|x_F, g, x, y)] \\
 = & KL(q_\phi(z_F, z_B|x_F, g, x, y) || p_\theta(z_F, z_B|x_F, g, x, y)) \\
 & + \underbrace{\mathbb{E}_{q_\phi(z_F, z_B|x_F, g, x, y)} [\log p_\theta(x_F, g, x, z_F, z_B|y) - \log q_\phi(z_F, z_B|x_F, g, x, y)]}_{\triangleq \mathcal{L}_{\text{disCVAE}}(x_F, g, x, y; \theta, \phi)},
 \end{aligned}$$

Based on the disentangling assumptions, we write the generation model by

$$p_\theta(x_F, g, x, z_F, z_B|y) = p_\theta(x|z_F, z_B, y)p_\theta(x_F, g|z_F, y)p_\theta(z_F)p_\theta(z_B), \quad (\text{A.2})$$

the recognition model by

$$q_\phi(z_F, z_B|x_F, g, x, y) = q_\phi(z_B|z_F, x_F, g, x, y)q_\phi(z_F|x_F, g, y) \quad (\text{A.3})$$

and thus the variational lower bound $\mathcal{L}_{\text{disCVAE}}(x_F, g, x, y; \theta, \phi)$ is given by

$$\begin{aligned} & \mathcal{L}_{\text{disCVAE}}(x_F, g, x, y; \theta, \phi) \\ &= -KL(q_\phi(z_F|x_F, g, y)||p_\theta(z_F)) - \mathbb{E}_{q_\phi(z_F|x_F, g, y)} [KL(q_\phi(z_B|z_F, x_F, g, x, y)||p_\theta(z_B))] \\ & \quad + \mathbb{E}_{q_\phi(z_F|x_F, g, y)} [\log p_\theta(x_F, g|z_F, y)] + \mathbb{E}_{q_\phi(z_F, z_B|x_F, g, x, y)} [\log p_\theta(x|z_F, z_B, y)] \\ &= -KL(q_\phi(z_F|x_F, g, y)||p_\theta(z_F)) - \mathbb{E}_{q_\phi(z_F|x_F, g, y)} [KL(q_\phi(z_B|z_F, x_F, g, x, y)||p_\theta(z_B))] \\ & \quad - \mathbb{E}_{q_\phi(z_F|x_F, g, y)} [L(\mu_{\theta_F}(y, z_F), x_F) + \lambda_g L(s_{\theta_g}(y, z_f), g)] \\ & \quad - \mathbb{E}_{q_\phi(z_F, z_B|x_F, g, x, y)} L(\mu_\theta(y, z_F, z_B), x) \end{aligned} \quad (\text{A.4})$$

In the last step, we assumed that $\log p_\theta(x_F, g|z_F, y) = \log p_\theta(x_F|z_F, y) + \lambda_g \log p_\theta(g|z_F, y)$, where λ_g is a hyperparameter when decomposing the probability $p_\theta(x_F, g|z_F, y)$. Here, the third and fourth terms are rewritten as expectations involving reconstruction loss (e.g., ℓ_2 loss) or cross entropy.

A.2 disCVAE Network Architecture

As we visualize in Figure A.1, disCVAE consists of four convolutional neural networks (one for foreground and the other for background for both recognition and generation networks).

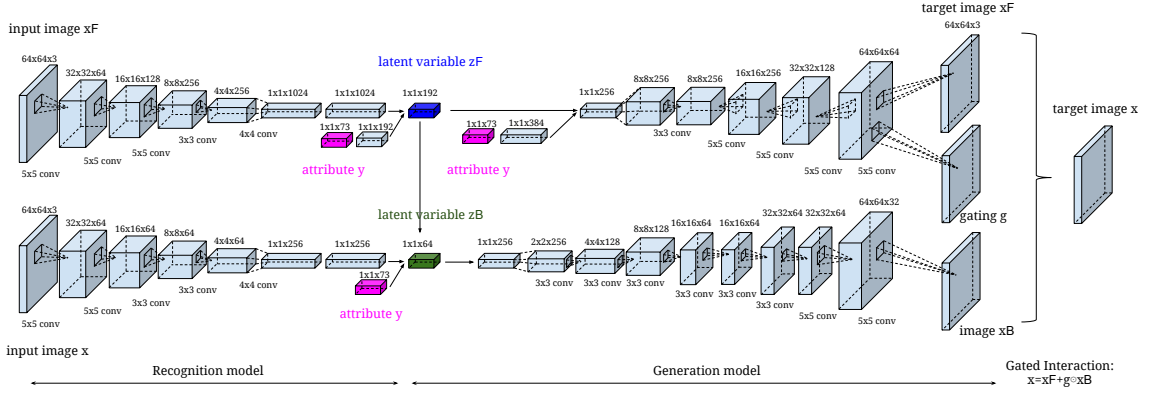


Figure A.1: Network Architecture for disentangling CVAE

The foreground encoder network consists of 5 convolution layers, followed by 2 fully-connected layers (convolution layers have 64, 128, 256, 256 and 1024 channels with filter size of 5×5 , 5×5 , 3×3 , 3×3 and 4×4 , respectively; the two fully-connected layers have 1024 and 192 neurons). The attribute stream is merged with image stream at the end of the recognition network. The foreground decoder network consists of 2 fully-connected layers, followed by 5 convolution layers with 2-by-2 upsampling (fully-connected layers have 256 and $8 \times 8 \times 256$ neurons; the convolution layers have 256, 256, 128, 64 and 3 channels with filter size of 3×3 , 5×5 , 5×5 , 5×5 and 5×5). The foreground prediction stream and gating prediction stream are separated at the last convolution layer.

We adopt the same encoder/decoder architecture for background networks but with fewer number of channels. For better modeling on the background latent variable z_B , we introduce attribute y and foreground latent variable z_F into the background encoder network, which also agrees with the assumption made in the derivation ($q_\phi(z_B|z_F, x_F, g, x, y)$). Here, the connection from foreground latent variable z_F to background latent variable z_B only exists in the recognition model.

Note that encoder networks are only used during the training stage. Once trained, we can generate images using decoder networks only.

A.3 Attribute-conditioned Image Progression

The attribute vector enables interpolation along each attribute dimension. For example, a face image with “smiling” attribute of value 1 is assumed to look more like a smiling face than that of value 0.5. To better analyze the proposed model, we generate images with interpolated attributes by gradually increasing or decreasing the values along each attribute dimension. We regard this process as *attribute-conditioned image progression*. Specifically, for each attribute vector, we modify the value of one attribute dimension by interpolating between the minimum and maximum attribute value. Then, we generate images with $p_\theta(x|y, z)$ by interpolating the value of y between the two attribute vectors while keeping latent variable z fixed. For visualization, we use the attribute vector from testing set.

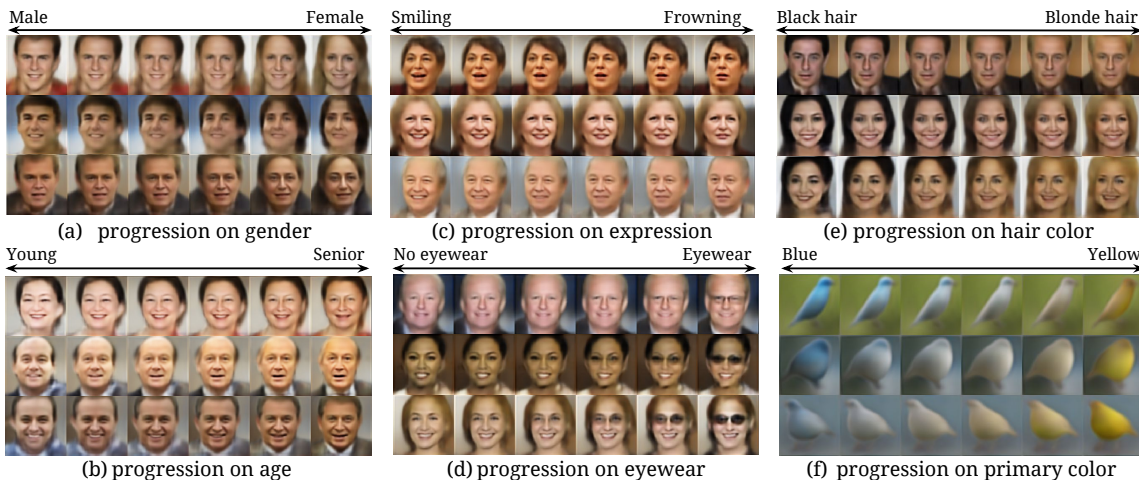


Figure A.2: Attribute-conditioned Image Progression. The visualization is organized into eight attribute groups (e.g., “gender”, “age”, “race”, “eyewear”, “facial expression”, “hair color”, “primary color (blue vs. yellow)”, and “primary color (black vs. white)”). Within each group, the images are generated from $p_\theta(x|y, z)$ with $z \sim \mathcal{N}(0, I)$ and $y = [y_\alpha, y_{rest}]$, where $y_\alpha = (1 - \alpha) \cdot y_{min} + \alpha \cdot y_{max}$. Here, y_{min} and y_{max} stands for the minimum and maximum attribute value respectively in the dataset along the corresponding dimension.

As we can see in Figure A.2, samples generated by progression are visually consistent with attribute description. For face images, by changing attributes like “gender”, “age” and “race”, the identity-related visual appearance is changed accordingly but

the viewpoint, background color, and facial expression are well preserved; on the other hand, by changing attributes like “eyewear”, “facial expression” and “hair color”, the global appearance is well preserved but the difference appears in the local region. For bird images, by changing the primary color from one to the other, the global shape and background color are well preserved. These observations demonstrated that the generation process of our model is well controlled by the input attributes.

A.4 Datasets for Motion Generation.

The evaluation is conducted on the datasets involving two representative human motion modeling tasks: Affect-in-the-wild (Aff-Wild) (*Zafeiriou et al.*) for facial motions and Human3.6M *Ionescu et al.* (2014) for full body motions. The Aff-Wild dataset contains more than 400 video clips (2,000 minutes in total) collected from Youtube with natural facial expression and head motion patterns. To better focus on face motion modeling (e.g., expressions and head movements), we leveraged the 3D morphable face model (*Paysan et al.*, 2009; *Blanz and Vetter*, 1999) (e.g., face identity, face expression, and pose) in our experiments. We fitted 198-dim identity coefficients, 29-dim expression coefficients, and 6-dim pose parameters to each frame with a pre-trained 3DMM-CNN (*Tran et al.*, 2017) model, followed by a face fitting algorithm (*Zhu et al.*, 2016b) based on optimization. This disentangled representation allows us to study face motion modeling without being distracted by unrelated factors such as facial identity, background scene, and illumination of the environment. We trained our model with 80% of the data on the expression and pose parameters since these are the main factors that change over time.

Human3.6M is a large-scale database containing more than 800 human motion sequences captured by 11 professional actors (3.6 million frames in total) in an indoor environment. For experiments on Human3.6M, we used the raw 2D trajectories of 32 keypoints and further normalized the data into coordinates within the range $[-1, 1]$.

We used subjects number 1, 5, 6, 7, and 8, for training and tested on subjects 9 and 11.

A.5 MT-VAE Network Architecture.

Our MT-VAE model consists of four components: sequence encoder network, sequence decoder network, latent encoder network, and latent decoder network. We build our sequence encoder and decoder using Long Short-term Memory units (LSTMs) (*Hochreiter and Schmidhuber, 1997*). We used 1-layer LSTM with 1,024 hidden units for both networks. For experiments on Aff-Wild dataset, the input to our sequence encoder is the 35-dimensional expression-pose representation (29 expression and 6 pose parameters) per timestep and we recursively predict the future parameters using our sequence decoder. For experiments on Human3.6M dataset, we used the 64-dimensional xy-coordinate representation (32 joints with 2 coordinates each joint) instead. Given past and future motion features extracted from our sequence encoder network, we build three fully-connected layers with skip connections within our latent encoding network. We adopted a similar architecture (three fully-connected layers with skip connections) for our latent decoder network. For all the models (including baselines), we fixed the bottleneck latent dimension to be 512 and found this configuration is sufficient to generate both face and full-body motions.

A.6 MT-VAE Implementation Details

We used ADAM (*Kingma and Ba, 2015*) for optimization in all experiments. For training, we used a mini-batch size of 256 and learning rate of 0.0001 with default ADAM settings (e.g., $\beta_1 = 0.9, \beta_2 = 0.999$). For experiments on Aff-Wild, we trained models to predict 32 steps in the future given a varying number of observed frames between 8 and 16. For experiments on Human3.6M, we trained models to predict 64

steps in the future given a varying number of observed frames between 10 and 20. To stabilize the training, we applied layer normalization (Ba et al., 2016) in both LSTMs and fully-connected layers. To encourage our latent variable to capture motion patterns, we applied the KL annealing technique (Bowman et al., 2015) during training, in which we gradually increased the weight of KL term from 0 to 1. For experiments on Aff-Wild only, we applied dropout of ratio 0.8 to both sequence encoder and decoder networks to learn more robust features.

We used Prediction LSTM (Villegas et al., 2017b) as a deterministic baseline. Similar model has been used in previous work for learning dynamics of human motion (Fragkiadaki et al., 2015; Chao et al., 2017). We implemented the vanilla VAE model (Babaeizadeh et al., 2018) as our stochastic baseline. Similar model has been utilized in Xue et al. (2016); Walker et al. (2016, 2017) for stochastic flow prediction from a single image. During training, we used \mathcal{L}_1 distance as the reconstruction term. We conducted extensive hyper-parameter search for vanilla VAE and our MT-VAE variants by enumerating smoothing window $K \in [0, 4, 8, 12, 16]$, motion ratio $\lambda_{\text{motion}} \in [0, 1, 5, 10, 20]$, cycle loss ratio $\lambda_{\text{cycle}} \in [0, 1, 5, 10, 20]$. All models achieve the best performance with $K = 8$ and $\lambda_{\text{cycle}} = 5$. Specifically, the best-performing MT-VAE (add) takes the hyper-parameter $\lambda_{\text{motion}} = 5$, while all other models take the hyper-parameter $\lambda_{\text{motion}} = 20$.

A.7 Details regarding Perspective Transformer Network

As defined in Chapter V, 2D silhouette $S^{(k)}$ is obtained via perspective transformation given input 3D volume \mathbf{V} and specific camera viewpoint $\alpha^{(k)}$.

Perspective Projection. In this work, we implement the perspective projection (see Figure A.3) with a 4-by-4 transformation matrix $\Theta_{4 \times 4}$, where \mathbf{K} is camera cali-

bration matrix and (\mathbf{R}, \mathbf{t}) is extrinsic parameters.

$$\Theta_{4 \times 4} = \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \quad (\text{A.5})$$

For each point $\mathbf{p}_i^s = (x_i^s, y_i^s, z_i^s, 1)$ in 3D world coordinates, we compute the corresponding point $\mathbf{p}_i^t = (x_i^t, y_i^t, 1, d_i^t)$ in screen coordinates (plus disparity d_i^t) using the perspective transformation: $\mathbf{p}_i^s \sim \Theta_{4 \times 4} \mathbf{p}_i^t$.

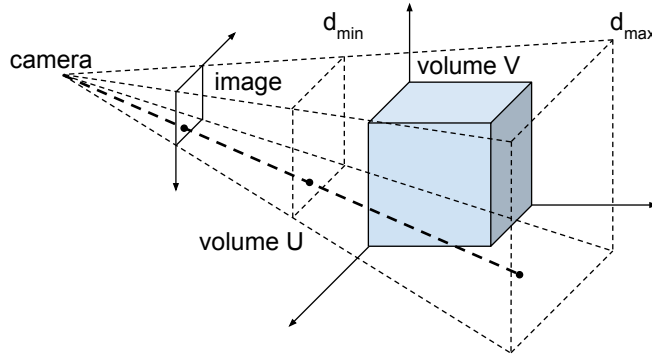


Figure A.3: Illustration of perspective projection. The minimum and maximum disparity in the screen coordinates are denoted as d_{min} and d_{max}

Similar to the spatial transformer network introduced in *Jaderberg et al. (2015)*, we propose a 2-step procedure: (1) performing dense sampling from input volume (in 3D world coordinates) to output volume (in screen coordinates), and (2) flattening the 3D spatial output across disparity dimension. In the experiment, we assume that transformation matrix is always given as input, parametrized by the viewpoint α . Again, the 3D point (x_i^s, y_i^s, z_i^s) in input volume $\mathbf{V} \in \mathbb{R}^{H \times W \times D}$ and corresponding point (x_i^t, y_i^t, d_i^t) in output volume $\mathbf{U} \in \mathbb{R}^{H' \times W' \times D'}$ is linked by perspective transformation matrix $\Theta_{4 \times 4}$. Here, (W, H, D) and (W', H', D') are the width, height and depth of input and output volume, respectively.

$$\begin{pmatrix} x_i^s \\ y_i^s \\ z_i^s \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} & \theta_{14} \\ \theta_{21} & \theta_{22} & \theta_{23} & \theta_{24} \\ \theta_{31} & \theta_{32} & \theta_{33} & \theta_{34} \\ \theta_{41} & \theta_{42} & \theta_{43} & \theta_{44} \end{bmatrix} \begin{pmatrix} \tilde{x}_i^t \\ \tilde{y}_i^t \\ \tilde{z}_i^t \\ 1 \end{pmatrix} \quad (\text{A.6})$$

In addition, we compute the normalized coordinates by $x_i^t = \frac{\tilde{x}_i^t}{\tilde{z}_i^t}$, $y_i^t = \frac{\tilde{y}_i^t}{\tilde{z}_i^t}$ and $d_i^t = \frac{1}{\tilde{z}_i^t}$, where d_i is the disparity.

Differentiable Volume Sampling. To perform transformation from input volume to output volume, we adopt the similar sampling strategy as proposed in *Jaderberg et al. (2015)*. That is, each point (x_i^s, y_i^s, z_i^s) defines a spatial location where a sampling kernel $k(\cdot)$ is applied to get the value at a particular voxel in the output volume U .

$$U_i = \sum_n^H \sum_m^W \sum_l^D V_{nml} k(x_i^s - m; \Phi_x) k(y_i^s - n; \Phi_y) k(z_i^s - l; \Phi_z) \quad \forall i \in \{1, \dots, H'W'D'\} \quad (\text{A.7})$$

Here, Φ_x , Φ_y and Φ_z are parameters of a generic sampling kernel $k(\cdot)$ which defines the interpolation method. We implement bilinear sampling kernel $k(x) = \max(0, 1 - |x|)$ in this work.

Finally, we summarize the dense sampling step and channel-wise flattening step as follows.

$$U_i = \sum_n^H \sum_m^W \sum_l^D V_{nml} \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|) \max(0, 1 - |z_i^s - l|)$$

$$S_{n'm'} = \max_{l'} U_{n'm'l'} \quad (\text{A.8})$$

Note that we use the max operator for projection instead of summation along one dimension since the volume is represented as a binary cube where the solid voxels have value 1 and empty voxels have value 0. Intuitively, we have the following two

observations: (1) each empty voxel will not contribute to the foreground pixel of S from any viewpoint; (2) each solid voxel can contribute to the foreground pixel of S only if it is visible from specific viewpoint.

BIBLIOGRAPHY

BIBLIOGRAPHY

- Averbuch-Elor, H., D. Cohen-Or, J. Kopf, and M. F. Cohen (2017), Bringing portraits to life, *ACM Transactions on Graphics (Proceeding of SIGGRAPH Asia 2017)*, 36(6), 196.
- Ba, J. L., J. R. Kiros, and G. E. Hinton (2016), Layer normalization, *arXiv preprint arXiv:1607.06450*.
- Babaeizadeh, M., C. Finn, D. Erhan, R. H. Campbell, and S. Levine (2018), Stochastic variational video prediction, in *ICLR*.
- Barnes, C., E. Shechtman, A. Finkelstein, and D. B. Goldman (2009), Patchmatch: A randomized correspondence algorithm for structural image editing, *ACM Transactions on Graphics*, 28(3), 24.
- Beeler, T., F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R. W. Sumner, and M. Gross (2011), High-quality passive facial performance capture using anchor frames, *ACM Trans. Graph.*, 30(4), 75:1–75:10.
- Bengio, Y., E. Thibodeau-Laufer, G. Alain, and J. Yosinski (2013), Deep generative stochastic networks trainable by backprop, *arXiv preprint arXiv:1306.1091*.
- Blanz, V., and T. Vetter (1999), A morphable model for the synthesis of 3D faces, in *SIGGRAPH*.
- Bohg, J., and D. Kragic (2010), Learning grasping points with shape context, *Robotics and Autonomous Systems*, 58(4), 362–377.
- Bowman, S. R., L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio (2015), Generating sentences from a continuous space, *arXiv preprint arXiv:1511.06349*.
- Bregler, C. (1997), Learning and recognizing human dynamics in video sequences, in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 568–574, IEEE.
- Chang, A. X., et al. (2015), Shapenet: An information-rich 3d model repository, *arXiv preprint arXiv:1512.03012*.
- Chao, Y.-W., J. Yang, B. Price, S. Cohen, and J. Deng (2017), Forecasting human dynamics from static images, in *CVPR*.

- Chen, L.-C., Y. Zhu, G. Papandreou, F. Schroff, and H. Adam (2018), Encoder-decoder with atrous separable convolution for semantic image segmentation, *arXiv:1802.02611*.
- Chen, Q., and V. Koltun (2017), Photographic image synthesis with cascaded refinement networks, in *ICCV*.
- Choy, C. B., D. Xu, J. Gwak, K. Chen, and S. Savarese (2016), 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction, in *ECCV*.
- Collobert, R., K. Kavukcuoglu, and C. Farabet (2011), Torch7: A matlab-like environment for machine learning, in *BigLearn, NIPS Workshop*.
- Cordts, M., M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele (2016), The cityscapes dataset for semantic urban scene understanding, in *CVPR*.
- Coumans, E., Y. Bai, and J. Hsu (), Pybullet physics engine, <http://pybullet.org>.
- Dang, H., and P. K. Allen (2014), Semantic grasping: planning task-specific stable robotic grasps, *Autonomous Robots*, 37(3), 301–316.
- de Aguiar, E., C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun (2008), Performance capture from sparse multi-view video, *ACM Trans. Graph.*, 27(3), 98:1–98:10.
- Denton, E., and R. Fergus (2018), Stochastic video generation with a learned prior, in *ICML*.
- Denton, E., S. Chintala, A. Szlam, and R. Fergus (2015), Deep generative image models using a Laplacian pyramid of adversarial networks, in *NIPS*.
- Denton, E., S. Gross, and R. Fergus (2016), Semi-supervised learning with context-conditional generative adversarial networks, *arXiv preprint arXiv:1611.06430*.
- Denton, E. L., and V. Birodkar (2017), Unsupervised learning of disentangled representations from video, in *Advances in Neural Information Processing Systems*, pp. 4417–4426.
- Dosovitskiy, A., and V. Koltun (2016), Learning to act by predicting the future, *arxiv preprint: 1611.01779*.
- Dosovitskiy, A., J. T. Springenberg, and T. Brox (2015), Learning to generate chairs with convolutional neural networks, in *CVPR*.
- Efros, A. A., A. C. Berg, G. Mori, and J. Malik (2003), Recognizing action at a distance, in *null*, p. 726, IEEE.
- Eigen, D., C. Puhrsch, and R. Fergus (2014), Depth map prediction from a single image using a multi-scale deep network, in *NIPS*.

- Fan, H., H. Su, and L. Guibas (2017), A point set generation network for 3d object reconstruction from a single image, in *CVPR*.
- Finn, C., I. Goodfellow, and S. Levine (2016a), Unsupervised learning for physical interaction through video prediction, in *Advances in Neural Information Processing Systems*, pp. 64–72.
- Finn, C., I. J. Goodfellow, and S. Levine (2016b), Unsupervised learning for physical interaction through video prediction, in *NIPS*.
- Fischer, P., A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox (2015), Flownet: Learning optical flow with convolutional networks, *arXiv preprint arXiv:1504.06852*.
- Fragkiadaki, K., S. Levine, P. Felsen, and J. Malik (2015), Recurrent network models for human dynamics, in *Computer Vision (ICCV), 2015 IEEE International Conference on*, pp. 4346–4354, IEEE.
- Gadelha, M., S. Maji, and R. Wang (2016), 3d shape induction from 2d views of multiple objects, *arXiv preprint arXiv:1612.05872*.
- Gatys, L. A., A. S. Ecker, and M. Bethge (2015), Texture synthesis using convolutional neural networks, in *NIPS*.
- Gauthier, J. (2015), Conditional generative adversarial nets for convolutional face generation, *Tech. rep.*
- Girdhar, R., D. F. Fouhey, M. Rodriguez, and A. Gupta (2016), Learning a predictable and generative vector representation for objects, *arXiv preprint arXiv:1603.08637*.
- Godard, C., O. Mac Aodha, and G. J. Brostow (2016), Unsupervised monocular depth estimation with left-right consistency, in *CVPR*.
- Goldfeder, C., M. Ciocarlie, H. Dang, and P. K. Allen (2009), The columbia grasp database, in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 1710–1716, IEEE.
- Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014), Generative adversarial nets, in *NIPS*, pp. 2672–2680.
- Gorelick, L., M. Blank, E. Shechtman, M. Irani, and R. Basri (2007), Actions as space-time shapes, *IEEE transactions on pattern analysis and machine intelligence*, 29(12), 2247–2253.
- Gregor, K., I. Danihelka, A. Graves, and D. Wierstra (2015), DRAW: A recurrent neural network for image generation, in *ICML*.

- Gualtieri, M., A. ten Pas, K. Saenko, and R. Platt (2016), High precision grasp pose detection in dense clutter, in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 598–605, IEEE.
- Gupta, A., A. A. Efros, and M. Hebert (2010), Blocks world revisited: Image understanding using qualitative geometry and mechanics, in *ECCV*.
- Ha, D., and D. Eck (2018), A neural representation of sketch drawings, in *ICLR*.
- Hinton, G. E., A. Krizhevsky, and S. D. Wang (2011), Transforming auto-encoders, in *ICANN*, Springer.
- Hochreiter, S., and J. Schmidhuber (1997), Long short-term memory, *Neural computation*, 9(8), 1735–1780.
- Hoiem, D., A. A. Efros, and M. Hebert (2005), Geometric context from a single image, in *ICCV*.
- Hoiem, D., A. A. Efros, and M. Hebert (2008), Putting objects in perspective, *International Journal of Computer Vision*, 80(1), 3–15.
- Hong, S., D. Yang, J. Choi, and H. Lee (2018), Inferring semantic layout for hierarchical text-to-image synthesis, in *CVPR*.
- Hu, Z., Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing (2017), Controllable text generation, *arXiv preprint arXiv:1703.00955*.
- Huang, G. B., M. Ramesh, and T. Berg (2007), Labeled faces in the wild: A database for studying, *month*, (07-49).
- Ionescu, C., D. Papava, V. Olaru, and C. Sminchisescu (2014), Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments, *IEEE transactions on pattern analysis and machine intelligence*, 36(7), 1325–1339.
- Isola, P., and C. Liu (2013), Scene collaging: Analysis and synthesis of natural images with semantic layers, in *ICCV*.
- Isola, P., J.-Y. Zhu, T. Zhou, and A. A. Efros (2017), Image-to-image translation with conditional adversarial networks, in *CVPR*.
- Jaderberg, M., K. Simonyan, A. Zisserman, et al. (2015), Spatial transformer networks, in *NIPS*.
- Johns, E., S. Leutenegger, and A. J. Davison (2016), Deep learning a grasp function for grasping under gripper pose uncertainty, in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 4461–4468, IEEE.

- Kalchbrenner, N., A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu (2016), Video pixel networks, *arXiv preprint arXiv:1610.00527*.
- Katz, D., A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz (2014), Perceiving, learning, and exploiting object affordances for autonomous pile manipulation, *Autonomous Robots*, 37(4), 369–382.
- Kemelmacher-Shlizerman, I., S. Suwajanakorn, and S. M. Seitz (2014), Illumination-aware age progression, in *CVPR*.
- Kingma, D., and J. Ba (2015), ADAM: A method for stochastic optimization, in *ICLR*.
- Kingma, D. P., and M. Welling (2014), Auto-encoding variational Bayes, in *ICLR*.
- Kingma, D. P., S. Mohamed, D. J. Rezende, and M. Welling (2014), Semi-supervised learning with deep generative models, in *NIPS*.
- Kopicki, M., R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt (2016), One-shot learning and generation of dexterous grasps for novel objects, *The International Journal of Robotics Research*, 35(8), 959–976.
- Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012), Imagenet classification with deep convolutional neural networks, in *NIPS*.
- Kulkarni, T. D., W. F. Whitney, P. Kohli, and J. Tenenbaum (2015), Deep convolutional inverse graphics network, in *Advances in Neural Information Processing Systems*, pp. 2539–2547.
- Kumar, N., A. C. Berg, P. N. Belhumeur, and S. K. Nayar (2009), Attribute and simile classifiers for face verification, in *ICCV*.
- Kutulakos, K. N., and S. M. Seitz (2000), A theory of shape by space carving, *International Journal of Computer Vision*, 38(3), 199–218.
- Lan, T., T.-C. Chen, and S. Savarese (2014), A hierarchical representation for future action prediction, in *European Conference on Computer Vision*, pp. 689–704, Springer.
- Laptev, I. (2005), On space-time interest points, *International journal of computer vision*, 64(2-3), 107–123.
- Laput, G. P., M. Dontcheva, G. Wilensky, W. Chang, A. Agarwala, J. Linder, and E. Adar (2013), Pixeltone: a multimodal interface for image editing, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.
- Le Roux, N., N. Heess, J. Shotton, and J. Winn (2011), Learning a generative model of images by factoring appearance and shape, *Neural Computation*, 23(3), 593–650.

- Lee, H., R. Grosse, R. Ranganath, and A. Y. Ng (2009), Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in *ICML*.
- Lenz, I., H. Lee, and A. Saxena (2015), Deep learning for detecting robotic grasps, *The International Journal of Robotics Research*, 34(4-5), 705–724.
- León, B., et al. (), Opengrasp: A toolkit for robot grasping simulation.
- Levine, S., P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen (), Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, *The International Journal of Robotics Research*, p. 0278364917710318.
- Li, M., K. Hang, D. Kragic, and A. Billard (2016), Dexterous grasping under shape uncertainty, *Robotics and Autonomous Systems*, 75, 352–364.
- Li, Y., S. Liu, J. Yang, and M.-H. Yang (2017), Generative face completion, in *CVPR*.
- Li, Z., Y. Zhou, S. Xiao, C. He, Z. Huang, and H. Li (2018), Auto-conditioned recurrent networks for extended complex human motion synthesis, in *ICLR*.
- Liu, C., J. Yuen, and A. Torralba (2016), Sift flow: Dense correspondence across scenes and its applications, in *Dense Image Correspondences for Computer Vision*, pp. 15–49.
- Liu, M.-Y., T. Breuel, and J. Kautz (2017), Unsupervised image-to-image translation networks, in *NIPS*.
- Mahler, J., J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg (2017), Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics, *arxiv preprint: 1703.09312*.
- Mahler, J., et al. (2016), Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards, in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1957–1964, IEEE.
- Mathieu, M., C. Couprie, and Y. LeCun (2016), Deep multi-scale video prediction beyond mean square error, in *ICLR*.
- Maturana, D., and S. Scherer (2015), Voxnet: A 3d convolutional neural network for real-time object recognition, in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 922–928, IEEE.
- Memisevic, R., and G. Hinton (2007), Unsupervised learning of image transformations, in *CVPR*.
- Michalski, V., R. Memisevic, and K. Konda (2014), Modeling deep temporal dependencies with recurrent grammar cells”, in *NIPS*.

- Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013), Distributed representations of words and phrases and their compositionality, in *Advances in neural information processing systems*, pp. 3111–3119.
- Montesano, L., and M. Lopes (2012), Active learning of visual descriptors for grasping using non-parametric smoothed beta distributions, *Robotics and Autonomous Systems*, 60(3), 452–462.
- Ngiam, J., A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng (2011), Multimodal deep learning, in *ICML*.
- Nikandrova, E., and V. Kyrki (2015), Category-based task specific grasping, *Robotics and Autonomous Systems*, 70, 25–35.
- Nitzberg, M., and D. Mumford (1990), The 2.1-d sketch, in *ICCV*.
- Oh, J., X. Guo, H. Lee, R. L. Lewis, and S. Singh (2015), Action-conditional video prediction using deep networks in atari games, in *NIPS*.
- Osa, T., J. Peters, and G. Neumann (2016), Experiments with hierarchical reinforcement learning of multiple grasping policies, in *International Symposium on Experimental Robotics*, pp. 160–172, Springer.
- Pathak, D., P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros (2016), Context encoders: Feature learning by inpainting, in *CVPR*.
- Paysan, P., R. Knothe, B. Amberg, S. Romdhani, and T. Vetter (2009), A 3d face model for pose and illumination invariant face recognition, IEEE, Genova, Italy.
- Pinto, L., and A. Gupta (2016), Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours, in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 3406–3413, IEEE.
- Pinto, L., D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta (2016), The curious robot: Learning visual representations via physical interactions, in *European Conference on Computer Vision*, pp. 3–18, Springer.
- Porter, T., and T. Duff (1984), Compositing digital images, in *ACM Siggraph Computer Graphics*, vol. 18, pp. 253–259, ACM.
- Qi, C. R., H. Su, M. Niessner, A. Dai, M. Yan, and L. J. Guibas (2016), Volumetric and multi-view cnns for object classification on 3d data, in *CVPR*.
- Radford, A., L. Metz, and S. Chintala (2015), Unsupervised representation learning with deep convolutional generative adversarial networks, *arXiv preprint arXiv:1511.06434*.
- Ranzato, M., V. Mnih, and G. E. Hinton (2010), Generating more realistic images using gated mrfs, in *NIPS*.

- Reed, S., K. Sohn, Y. Zhang, and H. Lee (2014), Learning to disentangle factors of variation with manifold interaction, in *ICML*.
- Reed, S. E., Y. Zhang, Y. Zhang, and H. Lee (2015), Deep visual analogy-making, in *NIPS*.
- Reed, S. E., Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee (2016), Learning what and where to draw, in *NIPS*.
- Rezende, D. J., S. Mohamed, and D. Wierstra (2014), Stochastic backpropagation and approximate inference in deep generative models, in *ICML*.
- Rezende, D. J., S. Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess (2016), Unsupervised learning of 3d structure from images, in *NIPS*.
- Rose, C., B. Guenter, B. Bodenheimer, and M. F. Cohen (1996), Efficient generation of motion transitions using spacetime constraints, in *SIGGRAPH*.
- Rubinstein, R., and D. Kroese (2004), The cross-entropy method: A unified approach to combinatorial optimization, monte-carlo simulation, and machine learning.
- Sangkloy, P., J. Lu, C. Fang, F. Yu, and J. Hays (2017), Scribbler: Controlling deep image synthesis with sketch and color, in *CVPR*.
- Saxena, A., J. Driemeyer, and A. Y. Ng (2008), Robotic grasping of novel objects using vision, *The International Journal of Robotics Research*, 27(2), 157–173.
- Sermanet, P., C. Lynch, J. Hsu, and S. Levine (2017), Time-contrastive networks: Self-supervised learning from multi-view observation, *arXiv preprint arXiv:1704.06888*.
- Simonyan, K., and A. Zisserman (2014), Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556*.
- Smith, K. A., and E. Vul (2013), Sources of uncertainty in intuitive physics, *Topics in cognitive science*, 5(1), 185–199.
- Sohn, K., W. Shang, and H. Lee (2014), Improved multimodal deep learning with variation of information, in *NIPS*.
- Sohn, K., X. Yan, and H. Lee (2015), Learning structured output representation using deep conditional generative models, in *NIPS*.
- Srivastava, A., A. B. Lee, E. P. Simoncelli, and S.-C. Zhu (2003), On advances in statistical modeling of natural images, *Journal of mathematical imaging and vision*, 18(1), 17–33.
- Srivastava, N., and R. R. Salakhutdinov (2012), Multimodal learning with deep Boltzmann machines, in *NIPS*.

- Srivastava, N., E. Mansimov, and R. Salakhudinov (2015), Unsupervised learning of video representations using lstms, in *International conference on machine learning*, pp. 843–852.
- Su, H., S. Maji, E. Kalogerakis, and E. Learned-Miller (2015), Multi-view convolutional neural networks for 3d shape recognition, in *ICCV*.
- Suwajanakorn, S., S. M. Seitz, and I. Kemelmacher-Shlizerman (2015), What makes tom hanks look like tom hanks, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3952–3960.
- Suwajanakorn, S., S. M. Seitz, and I. Kemelmacher-Shlizerman (2017), Synthesizing obama: learning lip sync from audio, *ACM Transactions on Graphics (TOG)*, 36(4), 95.
- Szegedy, C., et al. (2015), Going deeper with convolutions, *CVPR*.
- Szeliski, R. (2010), *Computer vision: algorithms and applications*, Springer Science & Business Media.
- Tang, Y., and R. Salakhutdinov (2013), Learning stochastic feedforward neural networks, in *NIPS*.
- Tang, Y., R. Salakhutdinov, and G. Hinton (2012), Robust boltzmann machines for recognition and denoising, in *CVPR*.
- Tatarchenko, M., A. Dosovitskiy, and T. Brox (2016), Single-view to multi-view: Reconstructing unseen views with a convolutional network, in *ECCV*.
- Tewari, A., M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt (2017), Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction, in *The IEEE International Conference on Computer Vision (ICCV)*, vol. 2.
- Thies, J., M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner (2016), Face2face: Real-time face capture and reenactment of rgb videos, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2387–2395.
- Tran, A. T., T. Hassner, I. Masi, and G. Medioni (2017), Regressing robust and discriminative 3d morphable models with a very deep neural network, in *Computer Vision and Pattern Recognition (CVPR)*.
- Tu, Z. (2007), Learning generative models via discriminative approaches, in *CVPR*.
- Tulsiani, S., T. Zhou, A. A. Efros, and J. Malik (2017), Multi-view supervision for single-view reconstruction via differentiable ray consistency, in *CVPR*.
- Tulyakov, S., M.-Y. Liu, X. Yang, and J. Kautz (2017), Mocogan: Decomposing motion and content for video generation, *arXiv preprint arXiv:1707.04993*.

- Tung, H.-Y., H.-W. Tung, E. Yumer, and K. Fragkiadaki (2017), Self-supervised learning of motion capture, in *Advances in Neural Information Processing Systems*, pp. 5242–5252.
- Vahrenkamp, N., L. Westkamp, N. Yamanobe, E. E. Aksoy, and T. Asfour (2016), Part-based grasp planning for familiar objects, in *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pp. 919–925, IEEE.
- Varley, J., C. DeChant, A. Richardson, A. Nair, J. Ruales, and P. Allen (2016), Shape completion enabled robotic grasping, *arxiv preprint: 1609.08546*.
- Villegas, R., J. Yang, S. Hong, X. Lin, and H. Lee (2017a), Decomposing motion and content for natural video sequence prediction, *ICLR*, 1(2), 7.
- Villegas, R., J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee (2017b), Learning to generate long-term future via hierarchical prediction, in *ICML*.
- Vondrick, C., H. Pirsiavash, and A. Torralba (2016), Generating videos with scene dynamics, in *NIPS*, pp. 613–621.
- Wah, C., S. Branson, P. Welinder, P. Perona, and S. Belongie (2011), The Caltech-UCSD birds-200-2011 dataset.
- Walker, J., A. Gupta, and M. Hebert (2015), Dense optical flow prediction from a static image, in *Computer Vision (ICCV), 2015 IEEE International Conference on*, pp. 2443–2451, IEEE.
- Walker, J., C. Doersch, A. Gupta, and M. Hebert (2016), An uncertain future: Forecasting from static images using variational autoencoders, in *ECCV*.
- Walker, J., K. Marino, A. Gupta, and M. Hebert (2017), The pose knows: Video forecasting by generating pose futures, in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3352–3361, IEEE.
- Wang, H., A. Kläser, C. Schmid, and C.-L. Liu (2011), Action recognition by dense trajectories, in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 3169–3176, IEEE.
- Wang, J., Z. Liu, Y. Wu, and J. Yuan (2012), Mining actionlet ensemble for action recognition with depth cameras, in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 1290–1297, IEEE.
- Wang, J. Y., and E. H. Adelson (1994), Representing moving images with layers, *Image Processing, IEEE Transactions on*, 3(5), 625–638.
- Wang, T.-C., M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro (2017), High-resolution image synthesis and semantic manipulation with conditional gans, in *ICCV*.

- Wang, X., A. Farhadi, and A. Gupta (2016), Actions~ transformations, in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2658–2667.
- Wang, Z., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli (2004), Image quality assessment: from error visibility to structural similarity, *IEEE transactions on Image Processing*, 13(4), 600–612.
- Wichers, N., R. Villegas, D. Erhan, and H. Lee (), Hierarchical long-term video prediction without supervision, in *ICML*.
- Williams, C. K., and M. K. Titsias (2004), Greedy learning of multiple objects in images using robust statistics and factorial learning, *Neural Computation*, 16(5), 1039–1062.
- Wu, J., T. Xue, J. J. Lim, Y. Tian, J. B. Tenenbaum, A. Torralba, and W. T. Freeman (2016a), Single image 3d interpreter network, in *ECCV*.
- Wu, J., C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum (2016b), Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling, in *Advances in Neural Information Processing Systems*, pp. 82–90.
- Wu, J., Y. Wang, T. Xue, X. Sun, B. Freeman, and J. Tenenbaum (2017), Marrnet: 3d shape reconstruction via 2.5 d sketches, in *Advances In Neural Information Processing Systems*, pp. 540–550.
- Wu, Z., S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao (2015), 3d shapenets: A deep representation for volumetric shapes, in *CVPR*.
- Xian, W., P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays (2018), Texturegan: Controlling deep image synthesis with texture patches, in *CVPR*.
- Xue, T., J. Wu, K. Bouman, and B. Freeman (2016), Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks, in *NIPS*, pp. 91–99.
- Yan, X., J. Yang, K. Sohn, and H. Lee (2016a), Attribute2image: Conditional image generation from visual attributes, in *ECCV*.
- Yan, X., J. Yang, E. Yumer, Y. Guo, and H. Lee (2016b), Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision, in *Advances in Neural Information Processing Systems*, pp. 1696–1704.
- Yang, F., J. Wang, E. Shechtman, L. Bourdev, and D. Metaxas (2011a), Expression flow for 3D-aware face component transfer, in *SIGGRAPH*.
- Yang, F., J. Wang, E. Shechtman, L. Bourdev, and D. Metaxas (2011b), Expression flow for 3d-aware face component transfer, in *ACM Transactions on Graphics (TOG)*, vol. 30, p. 60, ACM.

- Yang, F., L. Bourdev, E. Shechtman, J. Wang, and D. Metaxas (2012a), Facial expression editing in video using a temporally-smooth factorization, in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 861–868, IEEE.
- Yang, J., S. E. Reed, M.-H. Yang, and H. Lee (2015), Weakly-supervised disentangling with recurrent transformations for 3d view synthesis, in *NIPS*.
- Yang, J., A. Kannan, D. Batra, and D. Parikh (2017), Lr-gan: Layered recursive generative adversarial networks for image generation, in *ICLR*.
- Yang, Y., S. Hallman, D. Ramanan, and C. C. Fowlkes (2012b), Layered object models for image segmentation, *PAMI*, *34*(9), 1731–1743.
- Yeh, R. A., C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do (2017), Semantic image inpainting with deep generative models, in *CVPR*, pp. 5485–5493.
- Yosinski, J., J. Clune, A. Nguyen, T. Fuchs, and H. Lipson (2015), Understanding neural networks through deep visualization, *arXiv preprint arXiv:1506.06579*.
- Yumer, E., and N. J. Mitra (2016), Learning semantic deformation flows with 3d convolutional networks, in *ECCV*.
- Zafeiriou, S., D. Kollias, M. A. Nicolaou, A. Papaioannou, G. Zhao, and I. Kotsia (), Aff-wild: Valence and arousal in-the-wild challenge.
- Zhou, B., H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba (2017a), Scene parsing through ade20k dataset, in *CVPR*.
- Zhou, T., M. Brown, N. Snavely, and D. G. Lowe (2017b), Unsupervised learning of depth and ego-motion from video, in *CVPR*.
- Zhou, Y., and T. L. Berg (2016), Learning temporal transformations from time-lapse videos, in *European Conference on Computer Vision*, pp. 262–277, Springer.
- Zhu, J.-Y., P. Krähenbühl, E. Shechtman, and A. A. Efros (2016a), Generative visual manipulation on the natural image manifold, in *ECCV*.
- Zhu, J.-Y., T. Park, P. Isola, and A. A. Efros (2017a), Unpaired image-to-image translation using cycle-consistent adversarial networks, in *ICCV*.
- Zhu, J.-Y., R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman (2017b), Toward multimodal image-to-image translation, in *Advances in Neural Information Processing Systems*, pp. 465–476.
- Zhu, S., C. Li, C. C. Loy, and X. Tang (2014), Transferring landmark annotations for cross-dataset face alignment, *arXiv preprint arXiv:1409.0602*.
- Zhu, X., Z. Lei, X. Liu, H. Shi, and S. Z. Li (2016b), Face alignment across large poses: A 3d solution, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 146–155.