Learning Geometry from Vision for Robotic Manipulation

by

Jun Jin

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

This thesis studies how to enable a real-world robot to efficiently learn a new task by watching human demonstration videos. We propose to introduce a geometric task structure as an interpretable inductive bias to the learning problem. We aim to learn a representation that geometrically encodes "what the task is" from offline human demonstration videos and then transfer the learned representation to a robot controller using uncalibrated visual servoing (UVS). Specifically, we propose *Visual Geometric Skill Imitation Learning (VGS-IL)*, which uses a graph-structured task function to learn a task representation under structural constraints in the form of a predefined graph priori related to a geometric constraint type. The task function is optimized by Incremental Maximum Entropy Inverse Reinforcement Learning (InMaxEnt-IRL) based on "temporal-frame-orders" in human demonstration videos.

We show that the learned representation selects task-relevant image features to compose projective invariant geometric constraints, thus forming an efficient and interpretable representation. Secondly, the learned representation selects out the equivalent geometric constraints in the robot scene with adjoint geometric errors used in visual servoing controllers, thus removing the need for

extra robot training when mapping the task representation to robot actions. Lastly, by building task specification correspondence, we show that the learned task function selects task-relevant geometric constraints on categorical objects with the same task functionality, thus achieving task generalization. This is proposed as *Categorical Object Generalizable VGS-IL (CoVGS-IL)*. Various real-world experiments were conducted to verify our proposed method's ability regarding sample efficiency and task generalization.

Preface

This thesis is partly based on our previous publications. Details are as follows:

- Chapter 3 is based on our previous publication as [Jin, J., Petrich, L., Dehghan, M., Zhang, Z., and Jagersand, M. (2019, May). "Robot eye-hand coordination learning by watching human demonstrations: a task function approximation approach." In *2019 International Conference on Robotics and Automation (ICRA)* (pp. 6624-6630). IEEE.]. Changes have been made including correcting several typos, adding more descriptions on the method and experiment details in the evaluation.

- Chapter 5 is based on our previous two publications as [Jin, J., Petrich, L., Zhang, Z., Dehghan, M., and Jagersand, M. (2020, May). "Visual geometric skill inference by watching human demonstration." In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 8985-8991). IEEE.] and [Jin, J., Petrich, L., Dehghan, M., and Jagersand, M. (2020). "A geometric perspective on visual imitation learning." In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5194-5200). IEEE.]. Changes have been made including combining the two papers systematically, deleting the overlapping contents, adding details about the geometric constraint's graph structure design, and adding more details in the experiments.

The authors' contributions in the three mentioned publications are stated as below:

- Jun Jin's contribution was recognized as methodology proposal and implementation, experiments, data preparation and draft writing.
- Laura Pestrich implemented the baselines, helped with the experiments, prepared the data and participated in draft writing. The other co-authors helped with paper revision.

The remaining chapters have been submitted for publications. Details are as below:

- Chapter 4 "Task specification capability of geometric constraints" was submitted to *2022 IEEE International Conference on Robotics and Automation (ICRA)* titled with "How many features do we need: an empirical study on the capability of visual task specification using geometric features".
- Chapter 6 "Generalizable task representation learning" was submitted to *2022 IEEE International Conference on Robotics and Automation (ICRA)* titled with "Generalizable task representation learning from human demonstration videos: a geometric approach".

To myself of 2003,

For the everlasting dreams that brought me here.

Learn how to see. Realize that everything connects to everything else.

– Leonardo da Vinci.

Acknowledgements

I would like to express my gratitude to my supervisor Prof. Martin Jagersand. Perhaps there are many different mentorship styles among university professors, but what Martin has is unique. He shows enormous mentoring flexibility while being devoted to each of his students. He told me to define problems from practice, not from imagination. He provides me with many opportunities, like designing live demos in top robotic conferences and participating in the KUKA real-world challenges. This helped my research training a lot because real-world applications largely drive robotic research itself. This rewarding experience also helped me establish critical thinking and become an independent researcher.

During my PhD. study, I owe a lot of gratitude to all current and previous members of the robotics and vision lab. Dr. Camilo Perez Quintero gave me valuable guidance using ROS and the WAM robot. Dr. Masood Dehghan helped a lot in the development of my research projects, in the basics of control theory, and a lot of snacks and jokes after lunchtime. I would also like to thank Dr. Xuebin Qin and Abhineet Singh for the hands-on experience of using trackers. Lastly, I would like to thank people in our robotics lab, Vincent Zhang, Chen Jiang, Michael Przystupa and Laura Petrich, for the numerous valuable discussions and happy times.

Also, I am grateful to my supervisory committee member, Prof. A. Rupam Mahmood and Dr. Jun Luo, from whom I learned a lot about reinforcement learning and cognitive science in robotics research. Though I don't have a chance to include such research projects in my thesis, I am looking forward to exciting research collaborations soon. Besides, I would like to express my special thanks to Prof. Alan Lynch for the discussions and his introduction to the Autonomous Systems Initiative (ASI) in Alberta that broadens my vision of the real-world impacts of robotics. Meanwhile, I would like to thank Huawei Technologies Canada Co., Ltd for the financial support through internship and research grants that supported my PhD. study.

Finally, I would like to express my gratitude to my family members, especially my wife Zhou Cai, who sacrificed a lot for my five years' PhD. study. Furthermore, to my two kids Cassidy and Casper, my parents-in-law Yanjuan and Shixun, and my parents Yusheng and Chengfeng, for helping me fulfill this everlasting dream.

Contents

List of Tables

List of Figures

List of Symbols

T	Task fuction
$\mathcal{G}$	A graph-structured prior
E	Deep feature graph
g	A graph neural network
U	Graph selector
ϕ	An image patch encoder
$\mathbf{X}$	Graph node features
b_j	Task-relevance factor of a graph instance j
g_j	Normalized b_j
$\mathcal{E}_t$	Geometric error output of a geometric constraint at time t
o_t	Image observation at time t
z_t	Task embedding at time t
Δz_t	Task embedding difference between time t+1 and t
r_t	Score of transition $o_t \rightarrow o_{t+1}$
R	Score function
$\mathbf{r_{max}}$	Maximum scores
$\mathcal{Z}$	Partition function
$\dot{q}$	Joint velocites
$\tilde{\mathbf{J}}$	Estimated Jacobian matrix
$\mathbf{z}^i$	Task embedding vector of object i
$\mathcal{L}_{sim}$	Loss function of task similarity
$\{f_i\}$	Image features

(u, v)	Image pixel coordinates
(x, y)	Normalized image coordinates
$\mathcal{F}$	Admissible space constructed by combination of geometric features
η	The coverage of tasks that can be specified using geometric constraints

List of Common Acronyms

BC	Behavior Cloning
RL	Reinfocement Learning
IL	Imitation Learning
IRL	Inverse Reinforcement Learning
LfD	Learning from Demonstrations
PbD	Programming by Demonstration
GNN	Graph Neural Network
UVS	Uncalibrated Visual Servoing
VS	Visual Servoing
HRI	Human Robot Interface
AR	Augmented Reality
VR	Virtual Reality
MPC	Model Predictive Control

List of Publications

1.Jun Jin, Laura Petrich, Masood Dehghan, and Martin Jagersand. "A Geometric Perspective on Visual Imitation Learning." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2655–2662.

2.Jun Jin, Laura Petrich, Zichen Zhang, Masood Dehghan, and Martin Jagersand. "Visual geometric skill inference by watching human demonstration." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8985–8991.

3.Jun Jin, Laura Petrich, Masood Dehghan, Zichen Zhang, and Martin Jagersand. "Robot eye-hand coordination learning by watching human demonstrations: a task function approximation approach." In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6624–6630.

4.Jun Jin, Nhat M Nguyen, Nazmus Sakib, Daniel Graves, Hengshuai Yao, and Martin Jagersand. "Mapless Navigation among Dynamics with Social-safety-awareness: a reinforcement learning approach from 2D laser scans." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6979–6985,

5.Bowen Xie, Mingjie Han, Jun Jin, Martin Barczyk, and Martin Jagersand. "A Generative Model-Based Predictive Display for Robotic Teleoperation." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021,

6.Daniel Graves, Nhat M Nguyen, Kimia Hassanzadeh, Jun Jin, and Jun Luo. "Learning robust driving policies without online exploration." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021,

7.M. Dehghan, Z. Zhang, M. Siam, J. Jin, L. Petrich, and M. Jagersand. "Online Object and Task Learning via Human Robot Interaction." In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 2132–2138,

8.Xuebin Qin, Shida He, Zichen Zhang, Masood Dehghan, Jun Jin, and Martin Jagersand. "Real-Time Edge Template Tracking via Homography Estimation." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 607–612

CHAPTER 1

Introduction

The demand for new ways of robotic task specification: Robotic task specification has been the core problem for the industry since the first robot was deployed in the automobile production line in the 1970s. With nearly fifty years of technological development, as summarized in Fig. 1.1, though industrial solutions of robotic task specification have been improved significantly, the requirement for robotic experts to hand specify every task on-site has not been changed fundamentally. For example, the fact that for every task, on-site professionals are required to hand design each perception pipeline and each robot motion trajectory still remains, which impedes the large-scale deployment of robots. Therefore, new robotic task specification solutions that remove the requirements of robotic experts and generally apply to a wide range of manipulation tasks are still demanding.

Human demonstrations provide an intuitive interface for robotic task specification. Various methods have been proposed during the past decade including trajectory design based LfD (Learning from Demonstration [7]) and PbD (Programming by Demonstration [27]), reward function based IRL (inverse reinforcement learning [6]) and many recent approaches [40, 129, 140]. One major limitation of the above methods is that they typically require collecting massive

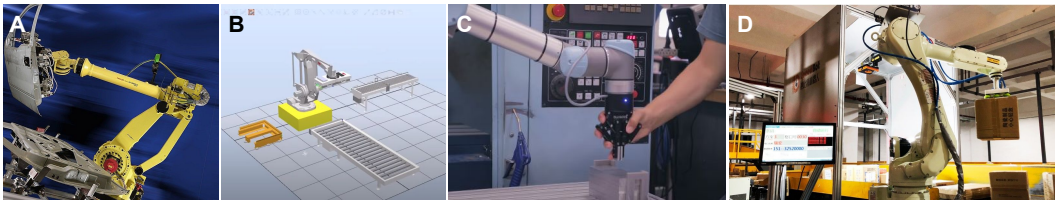


Figure 1.1: Task specification solutions in the industry still require robotic experts to hand design every task. **A**: Designing a car assembly task by PLC programming in the 1980s. **B**: Design a palletizing robotic system using ABB RobotStudio [1] which was firstly realsed in 1998 [24]. **C**: In the era of collaborative robots (2010s), task waypoints are specified by kinesthetic teaching [120] using UR10 robot from Universal Robots A/S . **D:** In the era of artificial intelligence powered robotic solutions, perception pipelines such as object detection/pose estimation combined with conventional trajectory designing tools are used to specify the task. The figure shows a parcel sorting robot released in 2021, and applied in the logistics industry from Speedbot Inc. [139], a robotic startup in China.

robot-involved samples either via human kinethetic teaching/teleoperation [37, 40, 131], or via robot-environment interactions [6, 37, 60]. Therefore, from the perspective of time cost, hardware wear-out and human resource cost, applying such methods in the real world does not manifest significant advantages over traditional methods that rely on calibrated instrumentation and hand-designed trajectories. These challenges impede the large-scale acceptance of methods using robot learning from human demonstrations in real-world tasks.

The demand for interpretable models: During the past decade, machine learning in robotics, compared to its influences in computer vision and natural language processing (NLP), has not been widely applied in real-world robotic applications. To better understand the industrial concerns, we conduct investigations by consulting from leading robotic AI firms, including Viwistar [146] in China, YPC in Canada [163], and a robotic system integration company located in the manufacturing cluster of Yangtze River Delta. We find that one major concern is the model's interpretability. Unlike computer vision and NLP tasks, robotic applications care more about safety and diagnosability, which relate to a model's interpretability. We typically demand a method that can be trusted when deployed in the real world and diagnosed when an unexpected failure occurs.

A recent research trend of interpretable machine learning is to introduce

strong inductive bias in learning, for example, using object relations to implicitly extract task-relevant information [11, 165], and using task hierarchies to explicitly model complex robot motions [114]. In these methods, a model's interpretability is enabled by structural representations. Compared to learning without any inductive bias, recent works [11, 165] also report using a structural representation will be more sample efficient and have a better generalization performance. Inspired by the above methods, we aim to research on what interpretable inductive bias can be introduced to robot learning that enables sample efficiency and task generalization.

Our approach: We address the above demands by introducing a *geometric task structure* (Fig. 1.3A) as an interpretable inductive bias to the problem of robot learning from human demonstrations. Specifically, we propose a method that learns geometric task specifications from offline human demonstration videos, as shown in Fig. 1.2A and B. We formulate the problem as learning a task represenation with structural constraints in the form of predefined graph structures that relate to geometric constraint types (Fig. 1.2C and D). We show that combining the approach of robot learning from human demonstrations and geometric task structure will remove the need for hand-selected features in traditional visual servoing methods [48], and will make the learning from human demonstrations sample efficient and task generalizable.

In summary, our method views the robotic task specification problem from a geometric perspective that can be learned using only human demonstration videos, which provides a practical way for real-world robot learning. Fig. 1.2 explains our method using a hammering task. The above claims are explained in the following sections.

The remainder of this chapter is as follows. In section 1.1, we review research on geometric task specification and robot learning from human demonstrations. Then we explain why our method will overcome each method's limitations. Sections 1.2, 1.3, and 1.4 introduce how our proposed method enables sample efficiency and task generalization in learning a new task. In the remaining sections, we summarize our contributions, describe outlines of each chapter and our publication notes.

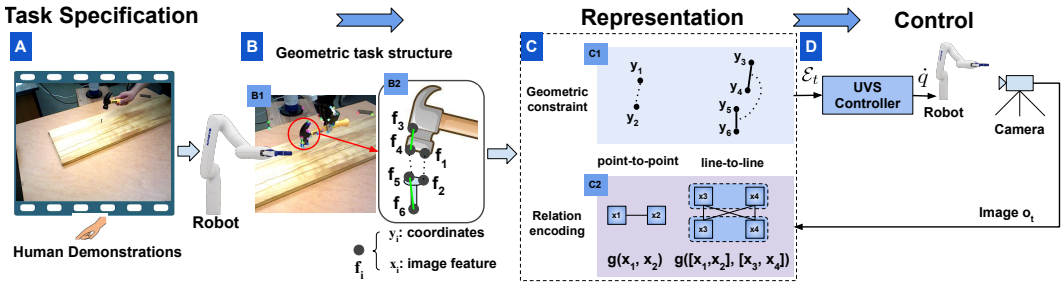


Figure 1.2: Overview of our method explained using a hammering task. Given human demonstration videos (Fig. A), we aim to learn a task representation that defines the task (Fig. B) by point-to-point (f_1 and f_2) and line-to-line (f_3, f_4, f_5 and f_6) constraints. This is achieved by constraining the representation space using predefined graph structures, where each structure defines a geometric constraint type. A task representation has two parts. (1) The geometric constraint part (Fig. C1) outputs geometric errors $\mathcal{E}_t$ that links to an uncalibrated visual servoing (UVS) controller [68] so that mapping the learned representation to robot actions is achieved without extra robot training. (2) The relation encoding part (Fig. C2) uses a graph neural network g to encode the association relationship between image features, thus representing point-to-point and line-to-line constraints. During testing, given an image observation o_t, the learned representation selects image features to fit in the graph nodes and outputs geometric errors $\mathcal{E}_t$ for robot control (Fig. D). In this example, $\mathcal{E}_t$ is computed from the coordinate difference between two points, $[y_2 - y_1]$ and the parallelism metric between two lines, $[(y_3 \times y_4) \times (y_5 \times y_6)]$. Details of how $\mathcal{E}_t$ is used in an UVS controller can be found in Chapter 7, Section 7.4.

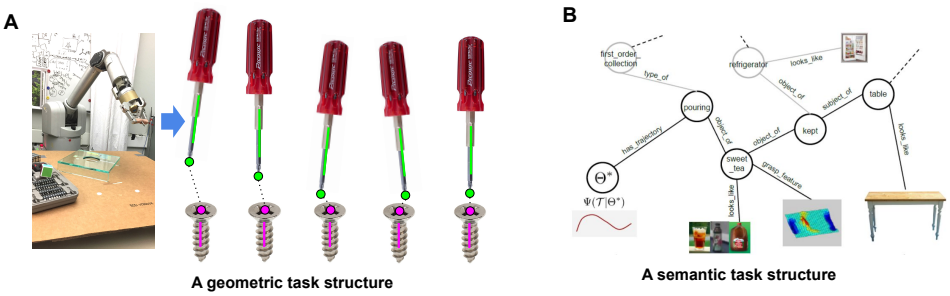


Figure 1.3: Two types of task structures: (1) a geometric task structure A) that defines tasks by geometric constraints; (2) a semantic task structure [126] (B) that extract task semantic meanings in the forms of knowledge graphs [110], grammar trees [161], behaviour trees [112] or the planning domain definition language (PDDL) [88]. We choose a geometric task structure as the inductive bias since its adjoint outputs—geometric errors, compared to a semantic task structure's output, are more friendly to robot controllers as shown in the previous Fig. 1.2E.

1.1Background

1.1.1Geometric task structure in robotics

Broadly speaking, there are two types of task structures: (1) a geometric task structure (Fig. 1.3A) and (2) a semantic task structure (Fig. 1.3B).

Humans have been using points and lines to describe structural concepts for about 5,000 years. Many tasks in our everyday life can be described using geometric features (as shown in Fig. 1.4 A, B). Likewise, in robotics, composing a task from image or point cloud data by the association, combinations and sequential linkage of geometric features is studied in the visual servoing literature, including the theoretical frameworks [20, 54, 59], system [29] and applications [49]. Examples of robotic task specification using geometric constraints are shown in Fig. 1.4 C and D.

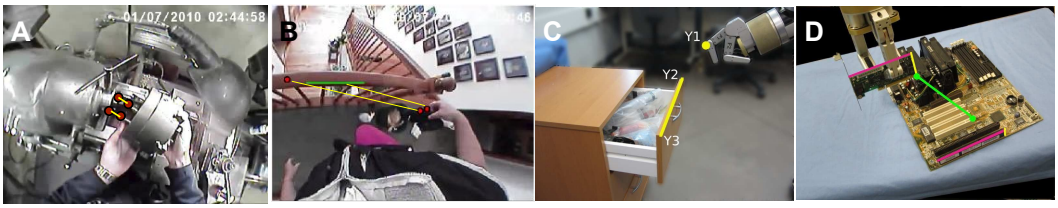


Figure 1.4: Examples of using geometric constraints to specify tasks, A and B are human task examples while C and D are robotic task examples. **A:** a tool insertion tasks [14] which is specified by two point coincidences. **B:** a wiping stairs handle task [14] which is specified by a point-to-line and point-to-point constraint. The point-to-line constraint means a point on the cloth stays contact with the stair handle. The point-to-point constraint means a point on the cloth should approach a point to the far end of the stair handle. **C:** an open drawer task [48] which is specified by a point-to-line constriant. **D:** a circuit board RAM insertion task [48] which is specified by a line-to-line alignment constraint and a point-to-point constraint.

In addition to traditional methods, geometric task structures, can be used as an intermediate representation in robot learning to improve sample efficiency and task generalization. For example, S. Levine et al. 2015 [89] and their following works [37] report sample efficiency using the spatial-softmax operator (Fig. 1.5A) to enforce the neural network extracting task-relevant feature point structures. Qin et al. 2019 [116] show task generalization in reinforcement learning by representing the task using keypoint structures (Fig.

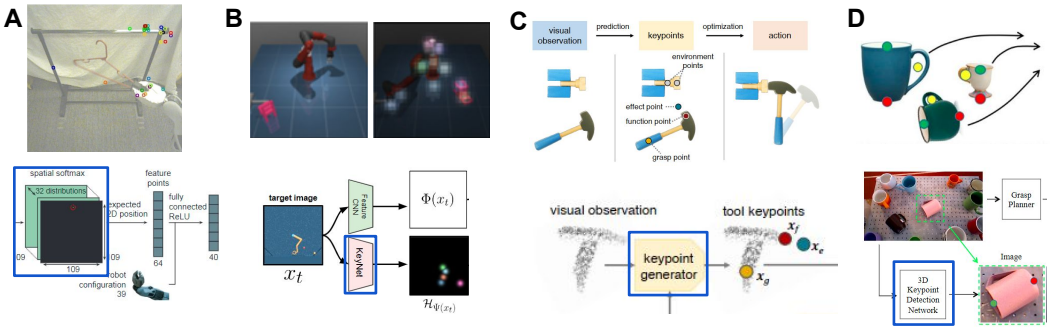


Figure 1.5: Example of current researches using geometric task structures in robot learning to improve sample efficiency (A, B) and task generalization performance (C, D). The top row shows the method, geometry structures are extracted by modules marked with a blue block. **A:** S. Levin et al. 2016 [89], the geometry module is "spatial softmax". **B:** Transporters, Kulkarni et al. 2019 [84], the geometry module is "KeyNet". **C:** KETO, Qin et al. 2019 [116], the geometry module is "keypoint generator". **D:** kPAM, Manuelli et al. 2019 [99], the geometry module is "3D keypoint detection network".

1.5C). Other examples can be found in Fig. 1.5 B and D.

Although there are the above benefits of a geometric task structure, the below limitations make it hard to be used in real-world tasks. For example, in visual servoing, a geometric task structure needs to be hand specified and relies on robust trackers [22]. On the other hand, in robot learning methods, a geometric task structure resides in the intermediate layers of neural networks that are jointly optimized with the policy. Since there are no supervisory clues that can be used to learn a representation with geometric task structures, massive training samples are typically required in the above approaches.

Our work aims to overcome the above limitations by using offline human demonstration videos to learn geometric task specifications, thus removes the need for hand-selected features and robust trackers in visual servoing. We use human demonstrations to guide the learning of a representation with geometric task structures. As shown in Fig. 1.6, Compared to methods that jointly learn a representation [89, 116], our method uses offline human demonstrations to guide the learning, thus can better extract task-relevant geometric features with fewer robot-involved samples.

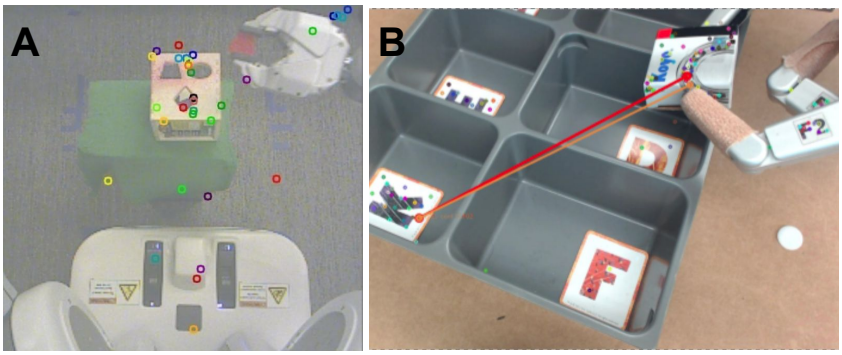


Figure 1.6: Comparison of our method to other feature-point representation based works [89]. **A**: the representation learned in [89] are visually salient, but not necessarily task-salient. For example, in A, the insertion task, no feature points are extracted on the to-be-inserted block. **B**: As a comparison, our method selects features by observing their saliency in describing task movement and alignments. Furthermore, since our learned features are specific to the task, task generalization can be achieved by selecting features on different objects but are specific to the same task (as described in Chapter 6).

1.1.2Robot learning from human demonstrations

Robot learning from human demonstration approaches include the early pioneers of robot learning by watching human demonstrations [1], robot learning from demonstration (LfD [7]) or robot programming by demonstration (PbD [27]), behavior cloning [7] , inverse reinforcement learning (IRL) or apprenticeship learning [6], and some recent approaches [37, 40, 60, 140]. A common assumption shared by different methods is that human demonstration data contains task-relevant information, which helps to boost up robot learning in two folds. (1) It facilitates the task specification problem. For example, IRL [6] removes the need for tedious reward engineering by approximating a reward function using the *expert assumption* in human demonstrations. For another example, in the third-view visual imitation learning [140], we can learn a task goal generator from human demonstrations since they define the task. (2) Some approaches collect state and robot action as sample pairs, which provide supervisory signals in learning a controller. Such guided state-action mapping is used as a learning clue in approaches including LfD [7], behaviour

[1]The earliest work can be traced back to Ikeuchi et al. [66] and Kuniyoshi et al. [85] in 1994.

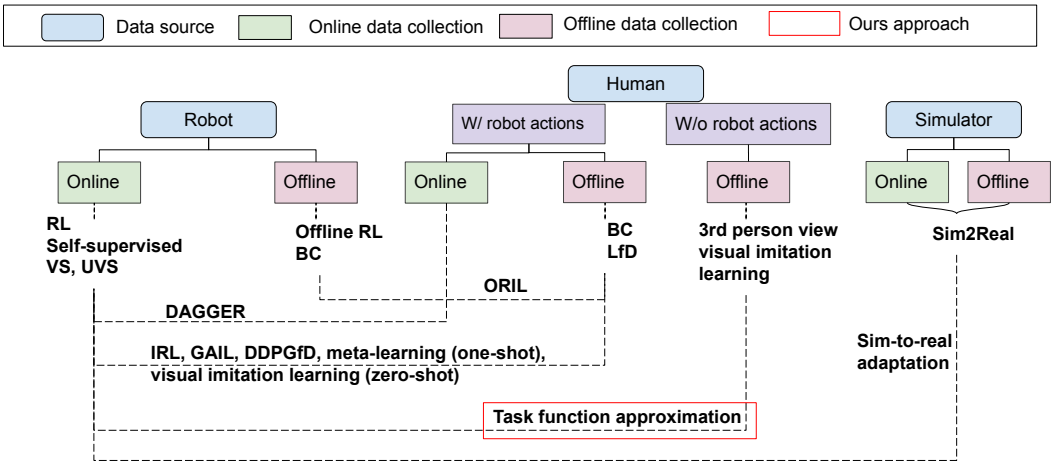


Figure 1.7: Summary of existing methods from the perspective of training data and the position of our approach marked with red sqaure. A list of all the methods metioned above with citations is as below : RL [89], self-supervised [70, 131], VS [22], UVS [68], offline RL [90], Behavior cloning (BC) [63], DAGGER [121], IRL [6, 37], GAIL [60], DDPGfD [145], meta-learning (one-shot) [40], visual imitation learning (zero-shot) [108], ORIL [173], LfD [7], 3rd person view visual imitation learning [130, 134], Sim2real [78]

cloning [7] and one-shot visual imitation learning [40]. Fig. 1.7 shows the characteristics of learning from human demonstration approaches that differ from other robot learning methods from the perspective of training data.

One major limitation of the above mentioned methods is that , as shown in Fig. 1.7, most robot learning from human demonstration approaches require a massive amount of robot-involved samples since the state-action space is large to be covered properly either by collecting massive data from human kinesthetic teaching/teleoperation (BC [7], one-shot [40]) or by robot-environment interactions (IRL [6], GAIL[60]). The requirement of massive robot-involved sample collection impedes the acceptance of learning from demonstration methods in real-world applications.

In order to overcome the above limitations, one approach is to decouple learning *what the task is* to *how to do the task*. Such a decoupled approach views human demonstration data as the information source for only task specification purpose. The key insight of this approach is to rethink the *correspondence problem* [7] in learning from demonstration (LfD) that the

actuation mechanisms of a robot and a human are fundamentally different. Therefore, the only shared component is the mapping from observation to task definitions—*task representation*. In this learning paradigm, human only needs to show the robot the task without tedious kinesthetic teaching or teleoperation.

This learning paradigm is named as learning by watching [66, 94], or third view visual imitation learning [134, 140]. Since there are no robot actions in the training samples, there is no link between the learned task definition to a specific robot's actions. Thus it is challenging to train a robot controller. For example, in [140], the task specification is represented by a neural network parameterized goal generator, which requires massive robot-environment interactions when using the goal generator to control the robot. This limitation contradicts the benefit of the decoupled learning paradigm. Is there a better way to represent the task specification that enables effcient robot control?

Our work uses a geometric task structure as a priori to construct a representation space that is shared by the human demonstrator and robot imitator. Given a task T, the insight is that geometric constraints-based task specification are the same regardless of the task executor is human or robot. For example, a hammering task is specified by geometric constraints on features from the hammer and the nail regardless of whether a human or a robot holds the hammer (Fig. 1.2A, B). In addition to form a shared representation space, a geometric task structure outputs geometric errors (Fig. 1.2D) that can be directly used in visual servoing controllers without extra robot training (Fig. 1.2E).

Therefore, our approach constructs a geometry-structured representation space to learn task specification from human demonstration videos, enabling human-friendly task teaching and solving the sample efficiency problem when mapping task specification to robot actions. We show that using a geometric task structure as a learning priori also brings in task generalization. The above claims are elaborated in Sections 1.2, 1.3 and 1.4.

1.2Research questions

As summarized above, our purpose is to introduce a geometric task structure as an inductive bias to construct a representation space that enables the robot to efficiently learn a new task from offline human demonstration videos. Therefore, our method enables teaching the robot a new task by simply showing it. Our introduced geometric task structure extracts task-relevant geometric constraints from an image state, which provides model interpretability. In order to implement such a learning system, we have to answer the following questions.

(1) *How to make geometric task specification learnable from data?* Current methods either require humans to select geometric constraints as the task specification [22, 48] or learning task-relevant geometric features without considering their associations that define a task [84, 116]. We propose to view learning geometric task specification as a representation learning problem that uses a graph-structured priori (Fig. 1.2D) to define each geometric constraint type. our method forms the representation of geometric constraints by selecting task-relevant image features into the graph nodes. We call our approach geometric task representation learning, which was published in ICRA2020 [75] and elaborated in Section 1.3 and Chapter 5.

(2) *How to learn geometric task specification from unlabeled human demonstration videos?* Current approaches commonly require tedious human annotation [99, 140] on each video frame. Inspired by the *expert assumption* in the IRL (inverse reinforcement learning) literature [6] that the human demonstration video itself actually defines what the task is, we propose to use the *temporal-frame-orders*, a time clue commonly used in human video understanding [18, 91, 148, 171], to learn such a representation from human demonstration videos. Our proposed method was published in ICRA 2019 [72] and is elaborated in Chapter 3.

(3) *How to efficiently map the learned task specification to robot actions?* We propose to use the learned representation in an uncalibrated visual servoing (UVS) controller, which online estimates the linear mapping between the learned representation to robot actions. This method was proposed as a

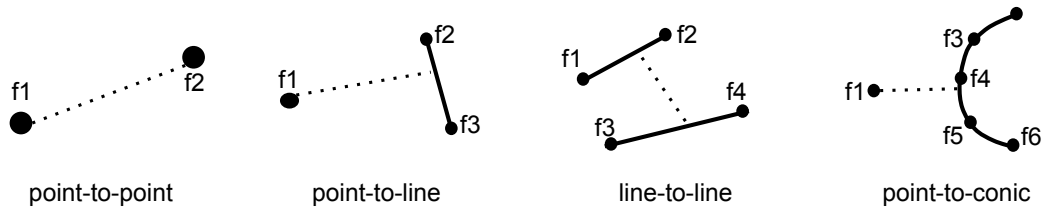


Figure 1.8: An arbitrary geometric constraint is a multi-entity relationship between a set of feature points, wherein the low level, the inner-connections (solid line) between feature points define a complex geometric feature and in the high level, the outer-connection (dashed line) between geometric features defines different geometric constraints.

conference paper in ICRA 2019 [72] and elaborated mainly in Chapters 3 and 7.

(4) *How to learn generalizable task representation?* Additionally, we are interested in generalizable task representation learning using human demonstration videos under different task settings. Our intuition is geometric constraints that specify a task stay the same under different task settings, such as in the hammering task using different hammers. We propose a method that uses the *task specification correspondence* (explained below in Section 1.3) to learn generalizable task representations from several human demonstration videos using different objects. Our proposed method is introduced in Chapters 6 and 7.

To summarize, the keywords of the thesis are *robotic task specification*, *representation learning*, *sample-efficient learning* and *task generalization*.

1.3Concept of geometric task representation

A key contribution of this research is our proposed geometric task representation. We explain the details as below.

Representation: Generally, an arbitrary geometric constraint is a binary relationship between two geometric primitives (point, line, conic and plane) observed in image space or point clouds. Considering the difficulty of representing complex geometric primitives using neural networks and that complex geometric features can be further decomposed to discrete points, then a ge-

ometric constraint can be represented as a multi-entity relationship between feature points (Fig. 1.8). This relationship can be encoded using an undirected graph $\mathcal{G} = \{V, E\}$, with each graph node $v \in V$ representing an image feature. The connections E between graph nodes define (1) how complex geometric features are constructed from feature points; (2) how geometric features are associated as a geometric constraint. This way, we use a graph to encode the relationship between image features to represent a geometric constraint. We call this approach *geometric task representation*, as shown in Fig. 1.8. The graph structure that encodes the multi-entity relationship to represent each geometric constraint is derived in Chapter 5.

Generalization: We explain how geometric task representation enables task generalization. The key is the *task specification correspondence* which means given the same task, geometric constraints from different task domains should be the same. For example, given task T, the task-relevant image features on categorical objects compose similar graphs that define the same geometric constraints. Thus, generalizable task representation learning can be fulfilled. This part is elaborated in Section 6.

Scalability: Geometric task representation has the scale up ability to represent complex tasks since different types of geometric constraints can be further combined or chained. Moreover, since they are encoded using graphs, extending them will scale up to define more complex tasks.

Linkage to controllers: Geometric task representation enables efficient robot controller design since each geometric constraint has its corresponding geometric errors (Fig. 1.2E), which can be used in visual servoing, uncalibrated visual servoing, or alternatively a policy training that maps geometric errors to robot actions. Using a geometric task representation in control is more efficient than encoding an image as task-irrelevant features by VAE [80] or forward prediction-based methods [107].

1.4Methodology

Learning geometric task specifications: We formulate the problem of learning geometric task specification from data as a representation learning problem that can be solved by approximating a graph-structured task function T. Formally, T is defined as a mapping from image observation o_t to a d-dimensional latent vector z_t which encodes the task definition, $T : o_t \rightarrow z_t$, where $o_t \in \mathbb{R}^{w \times h \times c}$, $z_t \in \mathbb{R}^d$, and w, h, c are image width, height, and channels respectively.

Introducing a geometric task structure into the task definition z_t is done by adding a graph structured priori $\mathcal{G}$ to the task function T. Specifically, let $\mathcal{G} = \{V, E\}$, which is an undirected graph that defines a geometric constraint type with nodes V relating to image features and edges E relating to their inner-connection and outer-connections (Fig. 1.8). Therefore, a task function T is defined as:

$$z_t = T(o_t|\mathcal{G}) \tag{1.1}$$

, which means T selects image features to fit the graph nodes V to construct a geometric constraint. T is called *task function* because it specifies the task by geometric constraints. So its output z_t is the compact vector representing the graph that describes the geometric constraint. z_t is computed by a graph neural network given a graph structure defined by $\mathcal{G}$ and image features selected by T as the graph nodes. Chapter 5 discusses the design of the graph-structured task function for each geometric constraint.

Meanwhile, applying the local coordinates of image features to the graph defined geometric constraint will output geometric errors $\mathcal{E}_t \in \mathbb{R}^k$, where k is the degree of freedom that a geometric constraint contributes. Examples of computing $\mathcal{E}_t$ for each geometric constraint type are included in Chapter 7, Section 7.2.

Learning from human demonstration videos: To optimize the task function cost-effectively, we apply unsupervised learning from human demon-

stration videos. Our learning approach does not need to run sample collection on the robot, which avoids robot hard wear-out and the tedious human kinesthetic teaching or teleoperation. The unsupervised learning clue is the *temporal-frame-orders* in human demonstration videos, similar to the *expert assumption* in inverse reinforcement learning (IRL) [6]. It makes unsupervised learning of the task function from human demonstration videos possible. Our method is formulated as *Incremental Maximum Entropy-Inverse Reinforcement Learning (InMaxEnt-IRL)* in Chapter 3 and Visual Geometric Skill-Imitation Learning (VGS-IL) in Chapter 5.

Generalizable task representation: The graph-structured task function facilitates task generalization by extracting task-relevant geometric constraints on different objects with the same task functionality. This is called *task specification correspondence* since the geometric constraints from different task domains define the same task. We propose *Categorical Object generalizable VGS-IL (CoVGS-IL)* to build *task specification correspondence* between categorical objects. CoVGS-IL learns generalizable task functions using different human demonstration videos with categorical objects, thus fulfils generalizable task representation learning.

System: Lastly, we combine each module together as a geometric task learning system, which is described from system architecture, data collection, training and real-world deployment, forming a conceptual prototype of a real-world geometric task learning system.

1.5Contributions

Our research contributions are as below:

- We propose Incremental Maximum Entropy-Inverse Reinforcement Learning (InMaxEnt-IRL, Chapter 3), an unsupervised learning method that optimize a task function from human demonstration videos using the *emporal-frame-orders* between frame transitions.

- We propose Visual Geometric Skill-Imitation Learning (VGS-IL, Chapter 5), an imitation learning method that learns geometric task structures from human demonstration videos. Specifically, VGS-IL uses a graph-structured task function to represent geometric constraints, and learn to select out task-relevant geometric constraints from human demonstration videos using our proposed InMaxEnt-IRL.

- We propose Categorical Object Generalizable VGS-IL (CoVGS-IL, Chapter 6), which learns a generalizable task representation by building the *task specification correspondence* (as defined in Section 1.3) between categorical objects. Specifically, CoVGS-IL uses a graph-structured task function to select out task-relevant image features on categorical objects with the same task functionality, thus fulfilling task generalization.

Other contributions include: (1) this is the first work that empirically studies the task specification capability of geometric features on recently proposed real-life task datasets (Chapter 4). (2) We provide a detailed description of how to deploy a geometric task learning system on a Kinova Gen3 robot (Chapter 7), which takes an input of human demonstration videos, learns the geometric task structure and outputs joint velocity commands that guide the robot to fulfil the task.

1.6Thesis outline

The remaining chapters are organized as follows.

- Chapter 2 reviews different robotic task specification approaches and geometric task specifications in robotics. Then we highlight our appproach among the current methods.

- Chapter 3 introduces InMaxEnt-IRL, an algorithm that uses *temporal-frame-orders* in human demonstration videos to optimize a task function. We give an elementary example by using a basic convolutional neural

network to parameterize the task function. Limitations of such parameterization give clues to the design of a graph-structured task function in Chapters 5 and 6.

- Chapter 4 studies the task specification capability of geometric constraints using real-life task datasets. We find using only points and lines has the same task speciation capability as using general and complex (conics, planes) geometric features. Meanwhile, we answer the question about how many geometric constraints are sufficient to define a task. These findings guide our neural network design of the task function in Chapters 5 and 6.

- Chapter 5 introduces VGS-IL, a method that learns task-relevant geometric constraints from human demonstration videos.

- Chapter 6 introduces CoVGS-IL, a method that learns generalizable task representation by building *task specification correspondence* between categorical objects.

- Chapter 7 assembles each proposed module, described from system architecture, data collection, training and real-world deployment, introduce a real-world geometric task learning system.

- Chapter 8 concludes the thesis, discusses our method's limitations, and describes the future direction.

1.7Publication note

This thesis is based on our previous publications. Details are as follows.

Chapter 3 is based on our previous publication as [72]. Changes have been made include correcting several typos, adding more descriptions on the method and experiment details in the evaluation. Chapter 5 is based on our previous two publications as [75] and [73]. Changes have been made include combining the two papers systematically, deleting overlapping contents, adding details about the geometric constraint's graph structure design, and adding more

details in the experiments. All the other chapters are our new contributions and are planned for publication (details are listed in the Preface page).

CHAPTER 2

Literature review

2.1A survey of robotic task specification

2.1.1Overview of methods

Based on how a task is specified, current robotic task specification methods can be categorized into four types: (1) methods that rely on human hand specification; (2) methods that require human design a reward/cost function; (3) methods that require trajectory level design of tasks[1]; (4) methods that use human demonstration to specify tasks.

Alternatively, based on the observation space dimensionality, current methods can also be divided into two categories, (1) methods that work for high dimensional observations, such as images or point clouds, and (2) methods that only work for low dimensional observation, such as object poses and robot states.

Furthermore, based on the learning level of robot motions, current methods can also be divided into two types, learning low-level motion skills and learning

[1]The trajectory level design category actually shares many commonalities with the other three, but it is listed separately since many impactful works [12, 65] have been proposed under this category.

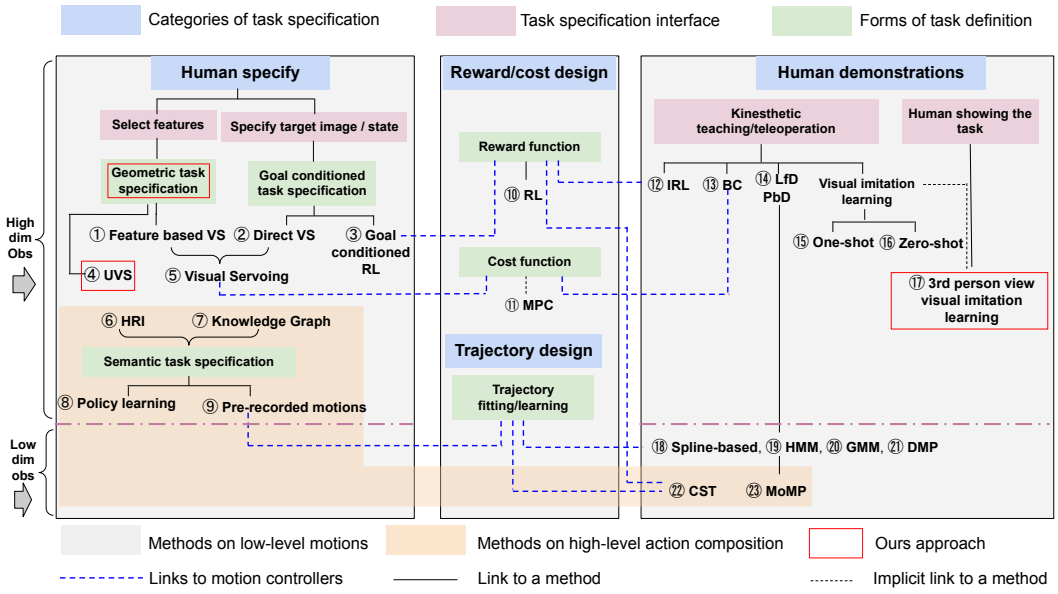


Figure 2.1: A survey of robotic task specification research. Current methods can be categorized based how a task is specified (shadowed in light blue color), whether a method can work with high-dimensional observations (marked with grey arrows), and if a method learns the low-level motions (shadowed in light grey color) or the high-level action compositions (shadowed in light orange color). A total number of 21 methods are summarized. Table 2.1 gives a description and the citation of all the methods mentioned above. Chapter 2, Section 2.1 reviews the details of each method.

high-level action compositions, such as complex tasks that require sequential linking of motion primitives.

A summary of method categories is shown in Fig. 2.1. Different colours mark different category types and the characteristics of the method. Table 2.1 summarises the methods mentioned in Fig. 2.1. Detailed analyses are as below.

2.1.2Human involved task specification

This category of methods requires humans to hand specify tasks. Based on the form to represent the task definition, approaches in this category can have three types: (1) representing task definition using geometric task specification [20, 54]; (2) representing task definition using semantic task specification [110, 161]; (3) representing task definition using a goal conditioned task specification [22,

#	Metehods	Comment
1	Hutchinson et al. 1996 [64]	Feature based visual servoing
2	Chaumette et al. 2006 [22]	Direct visual servoing
3	Schual et al. 2015 [127]	Goal conditioned RL
4	Jagersand et al. 1997 [68]	Uncalibrated visual servoing
5	Chaumette et al. 2006 [22]	Visual seroving
6	Guerin et al. 2016 [79], Paxton et al. 2017 [112]	HRI approaches, use AR/VR/Touch Screen to construct robot tasks
7	Paulius et al. 2016 [111]	Knowledge graph based appraoches, use knowledge graph to generate task plans
8	Saxena et al. 2014 [126]	Map the semantic task specification to robot actions by an extra policy learning module
9	Leidner et al. 2012 [88]	Map the semantic task specification to robot actions by pre-recorded trajectories
10	Sutton et al. 2018 [141]	Reinforcement learning, task specified by the reward function
11	Camacho et al. 2013 [17]	Model predictive control, task specified by cost function
12	Abbeel et al. 2004 [6]	IRL, Inverse reinforcement learning
13	Argall et al. 2009 [7]	BC, Behavior cloning, supervised learning
14	Billard et a. 2004 [12]	LfD, Learning from demonstration, PbD, Programming by demonstration
15	Finn et al. 2017 [40]	One-shot visual imitation learning
16	Pathak et al. 2018 [108]	Zero-shot visual imitation learning
17	Stadie et al. 2019 [140]	3rd person view visual imitation learning, learning by watching human demonstration videos
18	Miyamoto et al. 1996 [103]	Spline-based, using spline to cast motion characteristics
19	Williams et al. 2008 [152]	Using HMM to learn motion pritimives
20	Calinon et al. 2007 [16]	Using GMM to cast motion characteristics
21	Ijspeert et al. 2013 [65]	DMPs, dynamical movement primitives to represent robot motions
22	Konidaris et al. 2010 [81]	CST, constructing skill tree, an options based method for skill chaining
23	Mulling et al. 2013 [105]	MoMP, mixture of motor primitives. A method to select motion basis and use them to compose new motions.

Table 2.1: A list of methods mentioned in Fig. 2.1.

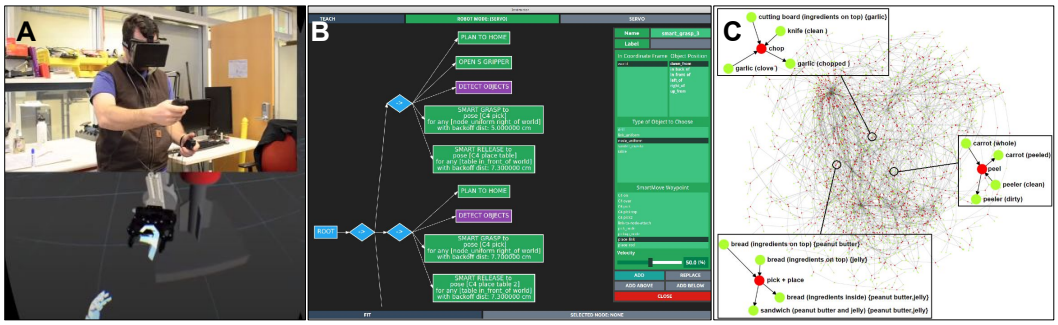


Figure 2.2: Task specification examples in research. **A**: A VR/AR robotic task teaching system from JHU [79]. **B**: A task tree based task specification system, coSTAR from JHU [112]. **C**: A knowledge graph constructed by annotated human demontration videos [111].

127].

Ordered by how general these methods can specify a task, semantic task specification is the most general form to represent a task, however, it is the most difficult one since its hierarchical structure commonly requires human-engineered sub-modules [88] or additional policy training when map the semantic task definition to robot actions [126]. Geometric task specification is the second general method, but it is controller-friendly since its output geometric errors are compact and task-relevant. Goal conditioned task specification has the lowest generality, it only works by assuming a task can be specified by a target image or state does not generally apply in many tasks.

We review each category type as follows.

Semantic task specification approaches: Approaches using semantic task specification can generate the semantic task plan either from an HRI (human-robot interface) device, such as AR/VR [79] and touch screens [112] or from a pre-obtained knowledge graph or behavior tree. For example in HRI research, using an AR/VR device (Fig. 2.2B) enables humans to specify a task by defining each motion, each gripper action visually [79]. Using a touchscreen-based task tree (coSTAR) (Fig. 2.2C) [112], humans can drag and construct tasks by the pre-defined modules.

In approaches that build robot knowledge graphs (Fig. 1.3 A), researchers

use massive data annotation from images or point clouds to extract task semantic graphs [111] by the method of ontology that views object instances as graph entities and object affordance as node properties. Then a new task can be planned using the pre-obtained knowledge graph. The semantic approaches commonly face difficulties when mapping the semantic task meaning to robot actions. There are two ways that map semantic task specification to robot actions. (1) Training a standalone policy but requires additional samples for each task [126]. (2) Using pre-recorded robot motions but has the trajectory generalization issue when initial task settings are different from the pre-recording conditions [88]. Further combing LfD (learning from demonstration) methods will address this issue.

Geometric task specification approaches: In approaches that use geometric task specification (Fig. 1.3 B), human needs to hand-select task features and specify types of geometric constraints. This form of task specification is closely related to geometric vision-based controllers (visual servoing [22]) that virtual linkage maps observed feature motions to robot actions. Visual servoing that takes in this form of task specification is called feature-based visual servoing. Besides, in the uncalibrated camera scenario, geometric task specification can also be used in uncalibrated visual sevoing [50]. *Our work* applies a similar approach in the thesis.

Goal conditioned task specification approaches: In approaches that use goal conditioned task speciation, human is required to specify a target image or target state. The corresponding controllers are *DVS* (direct visual seroving [22]) and *goal conditioned RL* [127]. Direct visual servoing drives robot motions by computing the relative pose (homography) between the target image and the current image observed by an eye-in-hand camera, Therefore, it requires the target and current image to be similar, which means their relative distance should be under a threshold given a distance metric. As a comparison, goal conditioned RL augments the state with goals and specifies the task via a reward function. It does not need to limit the distance between

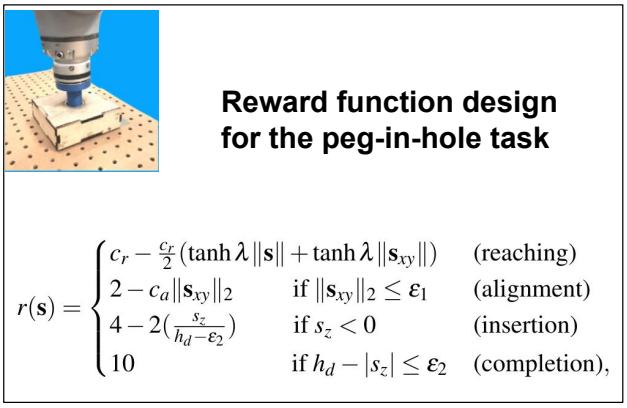


Figure 2.3: Reward function design example of a peg-in-hole task [87]. Each hyperparameters in the reward function determines if the learnd policy can finish the task successfully.

the current and target state. However, the training itself needs massive data and challenges to be done on real-world robots.

2.1.3Task specification via reward/cost function design

Methods fall in this category specify a task in a more general way than methods fall in the first category. However, as the "no free lunch" theorem, indicates, it brings extra challenges when applied in robot learning. For example, the reward shaping problem in reinforcement learning (RL) [141], does not only matter sample efficiency, policy generalization, but also maters the fundamental part—how should the agent learn the correct behaviour that we specified. In some literature, this phenomenon is called reward hacking. For another example, in model predictive control (MPC [17]), the design of the cost function also affects if the robot will finish the task as we expected. Tedious tuning efforts are required in design an MPC controller.

2.1.4Task specification via trajectory design

Methods that fall in this category include the early approach in industrial robots that require point, line and arc parameters to define the robot's 3D trajectory and semantic task specification approaches that use pre-recorded robot trajectories to map the semantic task meaning to robot actions.

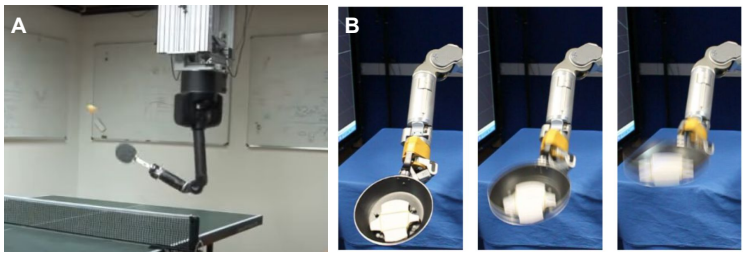


Figure 2.4: Task specification examples in research. **A**: Robot learns striking motion by explict modeling the motion primitives using MoMP (mixture of motion primitives [81, 105]). **B**: Robot learns flipping pan skills [83] by encoding human demonstrated motions as DMPs (dynamic movement primitives [65])

Moreover, there are various approaches in learning from demonstrations (*LfD* [12]) or learning by programming (*PbD*) that learn the characteristics of human demonstrated trajectory. Methods include spline-based [103], HMM (hidden Markov model)-based [152], GMM (Gaussian mixture model)-based [16] and DMP (dynamical system motor primitives)-based [65] approaches. These methods focus on the trajectory level, take efforts to build different models (*spline, HMM, GMM, DMP*) in order to cast the characteristics of human demonstrated motions. However, these methods only work on low-dimensional space that requires known object pose and robot states. It still remains challenging to directly learn trajectory properties from raw image observations.

2.1.5Task specification via human demonstration

Methods that fall in this category have a basic assumption that human demonstrators are experts and their demonstrations define the task itself. This aligns with our approach in the thesis.

Generally, methods can be divided into two types, (1) demonstrations that include robot actions via kinesthetic teaching or teleoperation; (2) demonstrations that do not have any robot actions and are collected merely by humans showing the task. The first type has the most research focus and proposed works.

Data collected via kinesthetic teaching or teleoperation: The first type can have many tricks to play since robot actions are included.

In *IRL* (Inverse Reinforcement Learning [6]), via the "expert assumption", we can learn a reward function from human demonstration, and jointly optimize the reward function and the policy. Therefore, IRL is typically not sample-efficient since it involves not only optimizing the reward but also the policy.

In *BC* (behaviour cloning [8]), we use supervised learning that maps observation to robot actions by learning a regressor. Behaviour cloning commonly will overfit to the demonstrated samples.

In addition to IRL and BC, when methods learn from video frames in human demonstrations, we call this approach *visual imitation learning*. Methods include one-shot visual imitation learning [40], which uses thousands of human kinesthetic teaching data to learn the shared mechanism of state to action mapping between different tasks, —meta-learning. Zero-shot visual imitation learning [108] follows a decoupled "what" and "how" learning paradigm and represent the task using a goal conditioned neural network,

The last category that falls in this data type are methods in *LfD* (learning from demonstration) and *PbD* (programming by demonstration) [12], which have fruitful works proposed. This type of learning includes learning can be divided into learning low-level individual motions and learning the high-level action compositions. The first types of learning focus on understanding the task from the trajectory level. Methods including spline-based [152], GMM-based [16] and DMP-based [65] approaches, which are introduced previously. The second type uses pre-learned motions to compose higher-level and complex actions. For example, *MoMP* (mixture of motor primitives) composes a new motion by a linear combination of motion basis in a pre-trained motion library. *CTS* (constructing skill trees) detects the changing point in human demonstrated motion trajectories and uses them to chain the motion trajectories, thus builds a skill tree by constructing options automatically.

Data collected via human showing the task: The second type, also known as 3rd-person-view visual imitation learning [140], commonly suffers from the "correspondence problem" [7] that humans and robots essentially have different kinematic mechanisms. In theory, it is impossible for the robot to imitate the mapping from observation to actions. Therefore, a common approach is a decoupled "what" and "how" learning paradigm that assumes the task definition can be learned and can be transferred from human demonstrator to robot imitator. Then using a pre-defined controller, or a conventional controller like UVS [68], or learning a policy, to map the task definition to robot actions. *Our work* applies a similar approach. The difference is that we introduce geometry as a structural prior when learning the task definitions.

2.2Task specification using geometric constraints

In robotics, representing a task using geometric constraints was firstly studied in visual servoing [22]. Research started with two-point coincidence and soon expanded to point-to-line, line-to-line, more geometric features (planes, conics, cylinders) and hierarchical structure linking [29] of the geometric constraints. Fig. 2.3 shows tasks that are defined using geometric constraints.

Assumptions: Task specification by geometric constraints has the following two assumptions. (1) A task can be refined by the association of geometric features on objects, tools, and robot end effectors. (2) All the features can be simultaneously observed in the same camera system. A quantitative study of the task specification capability using geometric constraints is included in Chapter 3.

2.2.1Geometric constraints:

Let's define the robot workspace as $\mathcal{W} \in SE(3)$ and camera view space as $\mathcal{V}$ which are basically features observed in the image plane. A task is defined by a list of geometric features $\{f_1, f_2, \ldots, f_n\}$ and their geometric constraints. For example, f_i can be a point, line, plane or conics. Their geometric constraints

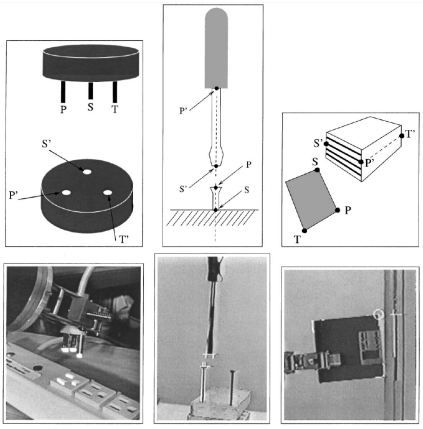


Figure 2.5: Tasks defined using geometric constraints [59]. Top row shows schematic examples and bottom row shows real-world image observations. Left: a socket plugin task is defined by three point-coincidences. Middle: a screwdriver task is defined by a four-point-collinearity. Right: a disk insertion task is defined by P, S, T, P' being coplanar and P, S coincidence with P', S'.

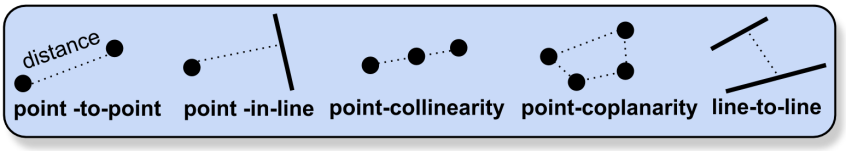


Figure 2.6: Examples of geometric constraints.

can be point-to-point coincidence, point-in-line, three-point-collinearity, four-point-coplanarity, line-to-line parallelism and perpendicularity, point-in-plane and etc., as shown in Fig. 2.6.

For simplicity, Let's define feature list $\{f_1, f_2, \ldots, f_n\}$ as $\mathbf{f}$, and $\mathcal{F}$ as non empty subset of $\mathcal{V}^n$ which means the Cartesian product of $\mathcal{V}$ for n times, and $\mathbf{f} \in \mathcal{F}$. We call $\mathcal{F}$ as the admissible space. For example, $\mathcal{F}$ for point-to-point constraint consists of all possible two-point combinations of the feature points observed in the image plane $\mathcal{V}$. A geometric constraint is a feature list $\mathbf{f} \in \mathcal{F}$ combined with a feature relationship structure, such as point-to-point is a two feature connection relationship.

2.2.2Task function:

A task function is defined as $T : \mathbf{f} \mapsto [0, 1], \mathbf{f} \in \mathcal{F}$. A geometric task function T maps a set of features in admissible space to 0 or 1 values.

A task specified by T is the equation $T(\mathbf{f}) = 0$, if which holds, we say a task is accomplished at $\mathbf{f}$. Note here 0 means task accomplishment since this is measured by minimizing the geometric loss error to 0 as shown in visual servoing literature [20, 64, 68]. Let's give some examples.

Let T_{p2p} designates two point coincidence task, denote $\mathcal{F}_{p2p} \stackrel{\text{def}}{=} \mathcal{V} \times \mathcal{V}, \mathbf{f} \in \mathcal{F}$. Let $\mathbf{f} = \{f_1, f_2\}$, where f_1, f_2 are two feature points observed in image. Now $n = 2$, by the rule which maps:

$$\mathbf{f} \mapsto \begin{cases} 0 & \text{the corresponding 3D points in } \mathcal{W} \text{ of } f_1 \text{ and } f_2 \text{ are the same.} \\ 1 & \text{otherwise} \end{cases} \tag{2.1}$$

T_{l2l} is a geometric task function defined on $\mathcal{F}_{l2l} \stackrel{\text{def}}{=} \mathcal{V} \times \mathcal{V}$. $\mathbf{f} = \{f_1, f_2\}$ is a set of two line features that are mapped according to:

$$\mathbf{f} \mapsto \begin{cases} 0 & \text{the corresponding 3D lines in } \mathcal{W} \text{ of } f_1 \text{ and } f_2 \text{ are parallel.} \\ 1 & \text{otherwise} \end{cases} \tag{2.2}$$

T_{col} is a geometric task function defined on $\mathcal{F}_{col} \stackrel{\text{def}}{=} \mathcal{V} \times \mathcal{V} \times \mathcal{V}$ that maps $\mathbf{f} = \{f_1, f_2, f_3\}$ according to the rule:

$$\mathbf{f} \mapsto \begin{cases} 0 & \text{the corresponding 3D points in } \mathcal{W} \text{ of } f_i \text{ are colinear.} \\ 1 & \text{otherwise} \end{cases} \tag{2.3}$$

Fig. 2.5 shows three examples of how a task can be defined using geometric constraints.

Therefore, the above task function plays the role of selecting geometric features to fit in the geometric constraint and output geometric errors for the controller. But it does not have an encoder that represents the selected geometric features and their relationships. This thesis extends the task function to a graph-structured neural network that selects out task-relevant geometric features and encodes the whole graph as the representation of geometric constraints.

2.2.3Task construction:

More complex tasks can be constructed by task operators and task changing.

Following [54], we define the following task operators:

- Complement: $\neg k(\mathbf{f}) \stackrel{\text{def}}{=} 1 - k(\mathbf{f})$, perform the opposite of task $k(\mathbf{f})$.
- Disjunction: $(k_1 \vee k_2)(\mathbf{f}) \stackrel{\text{def}}{=} k_1(\mathbf{f})|k_1(\mathbf{f})$, perform task $k_1(\mathbf{f})$ or $k_1(\mathbf{f})$.
- Conjunction: $(k_1 \wedge k_2)(\mathbf{f}) \stackrel{\text{def}}{=} k_1(\mathbf{f})|k_1(\mathbf{f})$, perform task $k_1(\mathbf{f})$ and $k_1(\mathbf{f})$.

Following [29], we can also construct hierarchical tasks by a task chaining link:$A = (\mathbf{E}^{init}(.), M, \mathbf{E}^{final}(.))$ [29], where $\mathbf{E}_{init}(.)$ and $\mathbf{E}_{final}(.)$ represent perceptual conditions and final configurations of a task $T(\mathbf{f})$.

Here $\mathbf{E}$ is image-based sub-task encoding using basic geometric constraints defined above. $\mathbf{M}$ describes a maintaining constraint, e.g., maintaining grasping, avoiding joint limits/singularity. $\mathbf{M}$ is not limited to projective geometric constraints, but any equational constraints that can drive control, for example, a "up" motion defined in robot base coordinate frame. As a result, a sequence $\{A_0, A_1, A_2, ...\}$ represent a chained task. Likewise, we can generate a tree-structured task hierarchy [29].

In this thesis, we only consider conjunction operators on geometric constraints. We leave other task construction methods as future directions to explore.

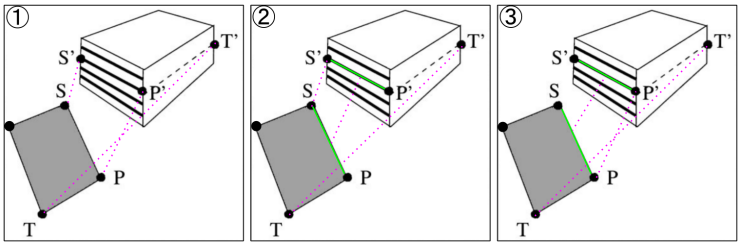


Figure 2.7: Examples of task function surjective mapping [59], which means multiple sets of geometric constraints can define the same task. For example, the same disk insertion task can be defined as shown in Fig. 2.5. Alternatively, it can be defined as three point-coincidences in (1), or two point-coincidences with a line-to-line parallelism in (2) and (3).

Task surjective mapping: Task specification using geometric features is a surjective mapping from $\mathcal{F}$ to the task definition, which means multiple sets of geometric features can specify the same task definition. As shown in Fig. 2.7, the alignment task can be defined by four point-coincidences or three point-coincidences, or two point-coincidences combined with two-line parallelism.

Projective ambiguity: For a given camera system, not all tasks are decidable because of projective ambiguity. This is because the task execution happens in the 3D robot's workspace $\mathcal{W}$ but a task function uses point and line features in $\mathcal{V}$ which are on the 2D image plane. For example, as shown in Fig. 2.8, we use image coordinates to decide if two points coincide, but there are situations where two different 3D points in $\mathcal{W}$ are observed as coincidence because of the camera's projective transformation. [59] gives an elaborate analysis of the projective decidability problem. Since adding multiple view camera systems will easily solve the problem, in this thesis we don't consider projective ambiguity that when a task is visually achieved from the image observations, we mark the trial as success.

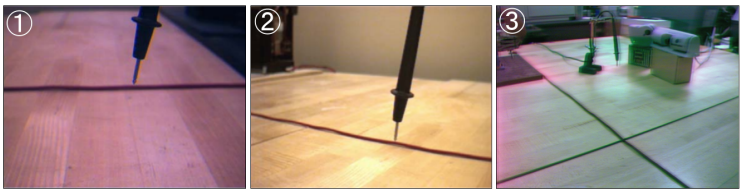


Figure 2.8: Example of projective ambiguity and its related decidability [69]. The pen tip falls in line as observed from (1) and (2) cameras, however it's far from the lines when observed in a slightly top-down view. This examples also shows projective ambiguity related task decidability. Designing a more multiple-view-camera system will solve the problem.

2.2.4The virtual linkage to robot actions:

The virtual linkage (Fig. 2.9) maps the motion of task-relevant features observed in the 2D image space to robot actions using the camera's geometry model — the perspective projection model. This mapping is called the virtual linkage in visual servoing [21, 22], which is explicitly expressed using an

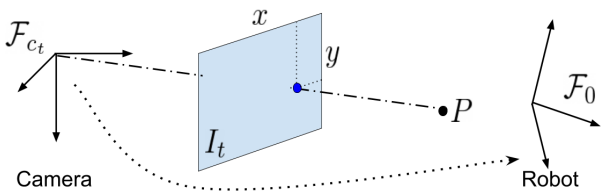


Figure 2.9: The virtual linkage that maps task-relevant features observed in the 2D image space to robot actions using the camera's geometry model [21].

interaction matrix.

2.3Summary: how this research stands out

On the learning paradigm level, this thesis employes a decoupled "what" and "how" approach that learns a task function from human demonstration videos and enables an efficient controller design by the graph-structured task function. This learning paradigm significantly reduces the training cost of real-world robot learning.

On the technique level, this thesis focuses on geometry-based task specifications. We introduce graph structures to the task function design that enables learning task-relevant geometric constraints from human demonstration videos. The geometric constraints are represented as image feature-based graphs. Compared to a direct approach learning from image pixels, such a graph-structured task function makes more sample efficient training and brings in generalizable task representation since the structure encodes the task definition which can be used to build "task specification correspondence" for task generalization.

CHAPTER 3

Task function approximation using human demonstrations

TL;DR:

> "Task function, which encodes the task definition, is learned from raw video frames by assuming the temporal-frame-orders in human demonstration videos."

We present incremental maximum entropy inverse reinforcement learning (InMaxEnt-IRL) that directly learns visual task specifications by watching human demonstrations. Task specification is represented as a task function, which is learned by maximizing the entropy distribution of observed state transitions in the expert trajectory. The learned task function's output is then used as continuous feedbacks in an uncalibrated visual servoing (UVS) controller designed for the execution phase.

This chapter will also give the formation of task function that is used to encode task definition from an image state and learned from human demonstration videos. For simplicity, we gives a case study that directly parameterizes the task function using convolutional neural networks without any structural

inductive bias. Analysis in Section 3.5.5 will show its drawbacks which motivate us to further improve it by introducing geometric task structure as an inductive bias to the task function design—a graph-structured task function, which is elaborated in Chapter 5.

3.1Introduction

As discussed in Chapter 1, Section 1.2, a learning paradigm that uses only human demonstration videos has a zero training cost of robot hardware wear-out but suffers the "correspondence" problem when mapping the observations to actions. Motivated by "peer learning" or "observational learning" [15] in human experience that humans can quickly grasp the task definition—"what is the task"—by merely watching others, and then practice the task individually through multiple success or failure attempts. We propose first to learn a task function, which encodes the task specification by measuring how good or bad the observed state transition is. Then, let the robot try the learned task function online that approximates the Jacobian, which maps the task function's output to robot joint actions.

The proposed method decouples learning "what is the task" from human demonstration videos and "how to do the task" from online robot trials, significantly reducing the number of training samples needed from real-world robot-environment interactions. Fig. 3.1 shows an overview of the learning paradigm.

Lastly, a case study that uses a convolutional neural network to parameterize the task function is given and evaluated under four different tasks under various task/ environmental setting changes.

3.2Related works

Since our approach is unsupervised learning from unlabeled human demonstration videos, it is worth firstly reviewing the unsupervised learning signals used in the state-of-art human video understanding research in computer vision. The answer is the time clues in human demonstration videos.

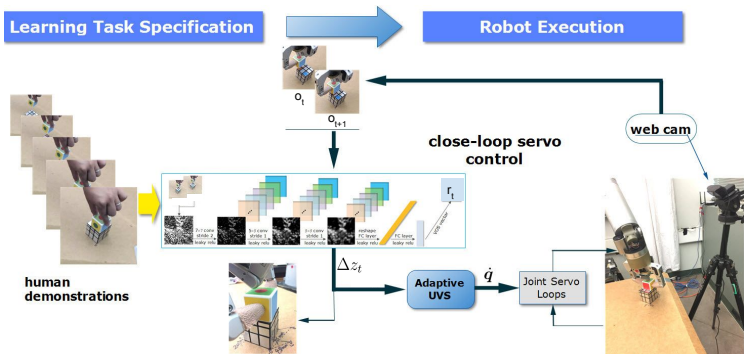


Figure 3.1: Our proposed method firstly learns a task function from human demonstration videos. The learned task function receives real-time video streams and outputs a task-relevance score vector Δz_t, which is used as continuous feedbacks in a close-loop UVS controller to guide robot motions in the execution phase.

Time clues in human demonstration videos: Using different time clues in human videos is not new in computer vision tasks like human activity classification and human video understanding. Overall, current approaches can be divided into two categories: using time clues in the local frame transitions and the global frame transitions.

For example, in the case of local frame transitions, the time clue of temporal-frame-orders are used to characterize human motions in video frames. Wei et al. 2018 [149], used optical flow and supervised learning applied on the locally selected consecutive T frames for the human activity analysis. Carreira et al 2020 [19], stack the local T frames as 3D-conv-nets that encodes the temporal-frame-order for human activity classification tasks.

Other works see the importance of characterizing the overall frame transitions—global transitions. For example, Dimitri et al. 2020 [171] propose "global-informative-score" of the whole video sequence as a clue for a video classification task. Cao et al. 2020 [18] propose a whole sequence matching technique for few-shot-temporal-alignment. Materzynska et al. 2020 [101] proposes "something-else", which analyze human-object interactions in the whole frame sequence that encode task goal by modelling geometric relationships between objects. This approach directly models the task goal as the characteristic of a human activity video, is similar to our approach that model task definitions

by geometric feature relationships in Chapter 5 and 6.

Task specification in visual servoing: This work was inspired by research advances in visual servoing, including defining a task function using geometric constraints [26, 30, 49, 55, 117], direct visual servoing (DVS [136]) method to remove the tracking challenge, and increasing generalization ability in visual servoing [10] using deep neural networks. DVS is very similar to our approach. However, it relies on a planar assumption and an inconvenient task specification process.

Inverse reinforcement learning (IRL) IRL seeks to derive a *reward function* from observable actions and is closely related to learning from demonstration (also known as imitation learning or apprenticeship learning) [6]. This learned reward function is synonymous with the visual servoing error function, notwithstanding the scalar output of the reward function. In contrast, the error function outputs a vector with dimensionality determined by task DOF. Maximum entropy IRL was proposed to manage the problem of suboptimal demonstrations [172], and Wulfmeier et al. 2015 represented the reward function using neural networks to handle non-linearity [155]. A challenge still exists in estimating the partition function Z, since it must solve the entire Markov Decision Process (MDP); this can be computationally expensive and unfeasible to generalize in a large action space or under unknown system dynamics. Finn et al. proposed an iterative solution using importance sampling to approximate a soft optimal policy[38] and recent works revealed the connection between IRL and generative adversarial networks (GAN) [39, 60]. Limitations of learning-based approaches have been detailed in Section1.

Robot learning by watching human demonstrations This approach, commonly known as visual imitation learning [6], has been recently gaining interest. Sermanet et al. presented TCN [131] to learn from contrastive positive and negative frame changes over time. Yu et al. proposed a meta-learning-based method [164] to encode prior knowledge from a few thousand

human/robot demonstrations, then learned a new task from one demonstration. End-to-end learning approaches lack interpretability. Our approach is similar to TCN but different in the controller design that we TCN separately trains a controller on the robot. In contrast, our approach maps the learned task encoding to robot actions using uncalibrated visual servoing.

In addition to the recently proposed end-to-end approaches, hierarchical methods that extract the semantic task definition from human demonstration videos have nearly two decades of research. Such approaches try to generate human-readable symbolic representations at a semantic level [27, 158, 161] to provide high-level task planning, which is important for generalizability. Ikeuchi et al. presented a general framework [66] that relies on object/task/-grasp recognition to generate assembly plans from observation. Modern approaches use a grammar parser [161], causal inference [158], and neural task programming [160]. Konidaris et al. proposed constructing skill trees [82] at the trajectory level to acquire skills from human demonstration using hierarchical reinforcement learning (RL) with options. This work presents a general framework to learn a tree-level structured task. However, such works require hard-coded recognition submodules or lack generality in various tasks.

3.3InMaxEnt-IRL: learning task function from human demonstration videos

We propose Incremental Maximum Entropy inverse reinforcement learning (*InMaxEnt-IRL*) to learn the task function from demonstrations. We utilize the same entropy maximization principle of the "expert assumption" in inverse reinforcement learning (IRL).

Unlike most IRLs that optimize on the whole trajectory level, our method learns from state observation transitions [43, 94, 129] which has the potential of bringing a better generalization ability [43]. Since it learns on the state observation level, changes between successive observations can be incrementally stacked with increased human demonstrations. For simplicity, we use one demonstration in this section to derive the basics.

3.3.1Terms and definitions

Firstly, we give the terms and definitions as below:

Task function: Let a task function T maps an arbitrary high dimensional observation o_i to a latent vector $z_i \in \mathbb{R}^d$, where d is the dimension of the latent vector. We denote a task function as: $z_i = T(o_i)$, where z_i is the task embedding since it describes the task. The latent vector z_i will be linked to the task by a score function as stated below.

Score function: Given two arbitrary state observations o_i and o_j, and their corresponding task embeddings z_i and z_j, we define a score function R that maps the observation change tuple (z_i, z_j) to a scalar score $r_{ij} \in \mathbb{R}$, where r_{ij} measures the task relevance weighting score: $R : (z_i, z_j) \rightarrow r_{ij}$. Linking the score function with the task function, we have: $r_{ij} = R(T(o_j), T(o_i))$. For computation convenience, we bound the r_{ij} into the range of $[-1, 1]$. The higher r_{ij}, the more task relevance that the change from o_i to o_j contributes.

Since the final learning signal is r_{ij}, in practice, we link the task function T and score function R as one function ϕ with parameter weights θ. So, we have:

$$r_{ij} = \phi(o_j, o_i|\boldsymbol{\theta}) \tag{3.1}$$

Apparently, without any supervision signal, the above function could encode any information. In order to ensure T and R extract task-definition related information, let us plugin human demonstrations.

Temporal-frame-orders assumption: Given an expert demonstration τ_i. Let o_t denotes the raw image observation sampled at time t, and $\tau_i = \{o_t\}$. Because of the "expert assumption" in the human demonstrated trajectory, an observed state transition from o_t to $_{t+1}$ is a positive change pair $(o_t \rightarrow o_{t+1})$. Its corresponding task relevance score, $r_t^* = \phi(o_t, o_{t+1})$ represents expert changes that will contribute to a high task relevance score. This is the *temporal-frame-orders* time clue that we used to optimize the task function.

Generic actions: A generic action a_{tj} is defined as any actions that cause state transition from o_t to o_j, regardless of where they are from, human or

robot. So a generic action is independent of a specific robot or human, it generally indicates a state transition. The mapping from generic actions to particular actions (e.g., joint velocity, torques) is done by a traditional controller in our method. Policy learning methods can also be used (e.g., RL[113]).

3.3.2Boltzmann Distribution with Human Factor

Next, we model the oberved state change behaviour in human demonstrations as a Boltzmann distribution that measures expert's preference over the selection of trajectory states. A Boltzmann distribution is commonly used in maximum entropy-based IRL [172] to model a human demonstrator's preference. Unlike IRL [172] approaches that model human preference over the trajecotry leve, we measure the expert's preference in the state transition level in that how a generic action is selected during demonstration that causes state changes from o_t to o_{t+1}.

We assume that in a human demonstration, at state o_t, the probability of transiting to the observed state o_{t+1} follows a Boltzmann distribution:

$$p(o_{t+1}|o_t) = \frac{1}{\mathcal{Z}_t} exp(r_t^*) p(r_t^*), \tag{3.2}$$

where r_t^* is the score of this observed state change, and

$$\mathcal{Z}_t = \mathbb{E}_{r_{tj} \sim p(r_{tj}; r_t^*)}[exp(r_{tj})] \tag{3.3}$$

, is the partition function, r_{tj} measures the score of state transition from o_t to an arbitrary state o_j, and $r_{tj} \sim \mathcal{N}(r_t^*, \sigma_0^2)$ is a truncated normal distribution within range [-1,1]. This means that the expert prefers the action with the highest score among all possible generic actions $\mathcal{A}_t = \{a_{tj}\}$.

σ_0 is a human factor prior since it varies with the human demonstrator's confidence and used to tackle the "imperfect expert demonstrations" as explained below.

3.3.3Imperfect expert demonstrations

The *temporal-frame-orders* assumption does not always hold in practice since perfectness is hard to be guaranteed. Considering imperfect expert demon-

strations, we introduce σ_0) as the human factor prior in the euqations above that refers the demonstrator's confidence α during the demonstration.

The confidence level α will have a resulting variance σ_0, where a high confidence level corresponds to low σ_0, vice versa. For example, if an expert is pretty sure about what actions are optimal at o_t, he is selecting only from a small set of candidate actions. On the other hand, if he/she is unsure what actions are 'good', the expert will have a hard time making decisions since the candidate action pool becomes large. We measure this behaviour using a Boltzmann Distribution with Human Factor as stated above.

The role of human factor σ_0 casts how stronger the expert assumption is. It will determine how much effort we should invest in optimization. Obviously, if an expert is very confident in his/her demonstrations, we should invest greater efforts in optimization since we know it's promising. On the contrary, if an expert is more indecisive in the demonstration, we should invest less effort in optimization since the expert assumption is weak. The effect of using σ_0 is further explained in the below optimization section and in Appendix B1.

3.3.4Optimization

Loss function: Given an expert trajectory τ_i, we formulate the learning problem by maximizing the likelihood of an observed state sequence $\{o_1, ..., o_n\}$, $p(\{o_t\})$. By applying MDP property, we have:

$$\mathcal{L} = \arg\max_{\theta} \sum log[p(o_{t+1}|o_t)] \tag{3.4}$$

With equation 3.2 and removing the last constant, the cost function can be further written as:

$$\mathcal{L} = \arg\max_{\theta} \sum r_t^* - log\mathcal{Z}_t \tag{3.5}$$

To give an intuitive interpretation of the derived loss function, our algorithm maximizes the score of positive state observation transitions to approximate the task function, which is precisely the "temporal-frame-orders" assumption used as a time clue in human video understanding tasks, smiliar to works in human video understanding [19, 149].

Optimization: Let's rewrite $\mathcal{Z}_t$ is a function of r_t^*, which is further represented using a task function T and score function R. Then, we have:

$$\nabla_\theta \mathcal{L} = \sum \nabla_\theta r_t^* - \frac{1}{\mathcal{Z}_t} \nabla_{r_t^*} \mathcal{Z}_t \nabla_\theta r_t^* \tag{3.6}$$

$\nabla_\theta r_t^*$ can be solved by back propagation from eq. (7) to the graph neural network in the skill kernel. $\mathcal{Z}_t$ can be estimated by a Monte Carlo estimator sampling o_1 samples from the truncated normal distribution $p(r_{tj})$:

$$\mathcal{Z}_t \approx \frac{1}{o_1} \sum^{o_1} exp(r_{tj}), \;\; r_{tj} \sim p(r_{tj}) \tag{3.7}$$

$\nabla_\theta \mathcal{Z}_t$ is the derivative of an expectation. By applying the log derivative trick, we have:

$$\begin{aligned} \nabla_{r_t^*} \mathcal{Z}_t &= \nabla_{r_t^*} \mathbb{E}_{p(r_{tj})}[exp(r_{tj})] \\ &= \mathbb{E}_{p(r_{tj})}[exp(r_{tj}) \nabla_{r_t^*} log\, p(r_{tj})] \\ &\approx \frac{1}{o_1} \sum^{o_1} exp(r_{tj}) \nabla_{r_t^*} log\, p(r_{tj}), \quad r_{tj} \sim p(r_{tj}) \end{aligned} \tag{3.8}$$

Since $p(r_{tj})$ is a truncated normal distribution[1] that is tractable, it's trivial to get:

$$\nabla_{r_t^*} log\, p(r_{tj}) = \frac{x_\mu}{\sigma_0} + \frac{exp(-b_\mu^2/2) - exp(-a_\mu^2/2)}{\sqrt{2\pi}\sigma_0[\phi(b_\mu) - \phi(a_\mu)]} \tag{3.9}$$

$x_\mu = (r_{tj} - r_t^*)/\sigma_0$, $a_\mu = (-1 - r_t^*)/\sigma_0$, $b_\mu = (1 - r_t^*)/\sigma_0$. where ϕ is defined in [150]. By combining the above equations, $\nabla_\theta \mathcal{L}$ is solved.

The optimization on ϕ is summarized in Algorithm 1.

3.4Designing choices of task functions

One challenge of using InMaxEnt-IRL is how to ensure the learned task function T's output can have numerical properties that are friendly to the controller used in the task execution phase. Given the example of using an online UVS controller, directly approximate the task function using a neural network may learn a vector z_t that encodes task definition. However, it can not guarantee a

[1]This is because we bound the task relevance score r_{tj} to [-1, 1] for computation convenience, as defined in Eq. 3.1. Note that if $p(r_{tj})$ is a normal distribution without a truncated domain, the loss function will degenerate to a constant. Proofs can be found on Appendix B1.

Algorithm 1: InMaxEnt-IRL

Input: Expert demonstration video frames $\{o_1, ..., o_n\}$, confidence level α
Result: Optimal weights $\boldsymbol{\theta}^*$ of the linked task and score function ϕ
Prepare State Change Samples $\mathcal{D}s$
Compute σ_0 using α; Shuffle $\mathcal{D}s$; Initialize $\boldsymbol{\theta}^0$
for *each iteration* **do**
  for *each observed sample change in* $\mathcal{D}s$ **do**
    Forward pass
    Compute r_t^*
    Compute $\nabla_{r_t^*}\mathcal{L} = \sum 1 - \frac{1}{\mathcal{Z}_t}\nabla_{r_t^*}\mathcal{Z}_t$
    $grad = \phi.backProp(\nabla_{r_t^*}\mathcal{L})$
    Gradient ascent update
    $\boldsymbol{\theta}^{n+1} = updateWeights(\boldsymbol{\theta}^n, grad)$
  end
end

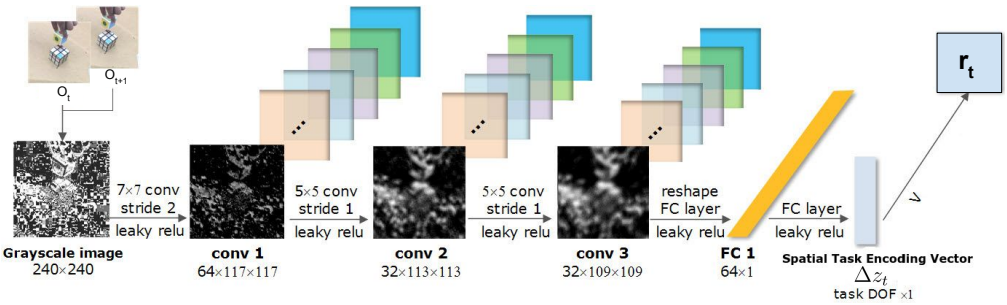


Figure 3.2: Neural network architecture of the task function.

high success rate in the task execution phase. As a result, the designing choice of task function affects the robot controller design.

This chapter gives the example of a task function using neural networks. Readers will find the success rate of such design reaches only 60% even given 11 human demonstrations. This result is equivalent to other visual imitation learning methods [40] that use only human demonstration videos and a controller trained on the robot, where 70% success rate was reported given thousands of human demonstration videos.

It is worth noting that, in the remaining chapters, we improve the success rate significantly by introducing graph structural priors to the task function design. As a result, the example given below can be treated as a case study of different task function designs.

3.4.1Example task function design

Given two successive states observed in human demonstrations, o_t and o_{t+1}, from eq. 4.1, we have $r_t^* = \phi(o_t, o_{t+1})$, which can be directly approximated using a neural network takes input of o_t and o_{t+1} and outputs r_t^*. To process an ordered input pair (o_t, o_{t+1}), one possible solution is to use a Siamese neural network structure for both o_t and o_{t+1}, and encode the positive transition by subtraction. Alternatively, in order to have a minimal parameter size of the neural network, we apply a simple modular subtraction between the two images to encode the state transition change $o_t \rightarrow o_{t+1}$ as Δo_t. So, we have $\Delta z_t = nn(\Delta o_t)$, where $\Delta z_t \in \mathbb{R}^d$ with all elements in the range $[-1, 1]$. Then a scalar score r_t^* is computed by a dot product with $\mathbf{v} = \frac{1}{d}[1, ..., 1]^\top$. Fig. 3.2 shows the network design.

3.4.2Example controller design

Given the above neural network's output Δz_t, uncalibrated visual servoing (UVS [68]) is used to directly approximate an affine mapping from task space to robot joint space. UVS has four steps in a closed-loop control manner: 1) estimate an initial Jacobian by driving random motions and observing changes in task function. This is the only online training process on a robot and costs an average of 4-7 seconds, depending on hardware speed. 2) Calculate action (joint velocity) using this Jacobian and execute this action. 3) Observe new changes in task function. 4) Broyden update the Jacobian.

Since we want an action that results in a maximum score, which can be defined as $\mathbf{r_{max}} = [1, ..., 1]^\top \in \mathbb{R}^d$, where d is the task degree of freedom. A robot action is computed by:

$$\dot{\mathbf{q}} = \hat{J}_t^\dagger \mathbf{r_{max}} \tag{3.10}$$

, where $\hat{J}_t^\dagger$ is the estimated Jacobian.

The biggest challenge is how to handle accumulated drifting error caused by continuous Broyden updates. In practice, we set a threshold vector $\mathbf{r_{thres}}$ to estimate $\hat{J}_{t+1}^\dagger$'s singularity proximity and perform $\hat{J}_0$ re-calibration if necessary.

We name this approach as adaptive UVS control, as shown in Algorithm 2.

Algorithm 2: Adaptive UVS Control

Input: Learned Task Function nn, real-time video stream S, robot joint number m
Result: Task fulfilled
Initial Jacobian Estimation
Estimate by exploratory motions: $\hat{J}_t = init(m)$
while *Task not fulfiled* **do**
  Set reference state: o_t = S.currentImage
  Move robot
  $\dot{q} = \hat{J}_t^{\dagger}\mathbf{r_{max}}$
  moveRobot($\dot{q}$)
  Return Δz_t
  Set current state: o_{t+1} = S.currentImage
  Compute Δo_t using o_t, o_{t+1}
  $\Delta z_t = nn(\Delta o_t)$
  Jacobian update
  if $\Delta z_t \preceq \mathbf{r_{thres}}$ *or havePhysicalConstraints* **then**
    Re-estimate: $\hat{J}_t = init(m)$
  else
    $\hat{J}_{t+1} = BroydenUpdate(\hat{J}_t, \Delta z_t, \dot{q})$
    Update current Jacobian: $\hat{J}_t \leftarrow \hat{J}_{t+1}$
  end
end

3.5Evaluations

3.5.1Objectives and setup

Evaluation objectives: Our experimental objective aims to answer the following three questions: (1) What kind of task is our method capable of, and what task fails? (2) How does generalization differ under variances with tasks and environments? (3) How does performance change with a varying number of demonstrations?

Experimental setup: We consider four tasks shown in Fig. 3.4 left: (1) a 3DOF task with regular geometric shapes "stack blocks"; (2) a 3DOF task with complex backgrounds "plug in a socket on circuit board"; (3) a same "plug in" task with 6DOF. (4) a 3DOF task with small visual signatures "pointing with the tip of a screw driver". All tasks were trained on a moderate PC (one GTX 1080Ti GPU) using 11 human demonstrations.

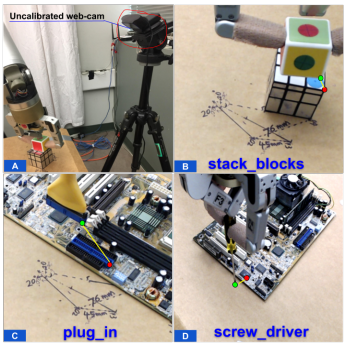


Figure 3.3: (A) experimental setup, a low-cost uncalibrated webcam is used to record human demonstrations and guide robot motions; **(B, C, D)** measurement of error using pixel distance between current position(green dot) to target(red dot). A threshold of 20 pixels ($\approx 7mm$) in a 580×580 image is used to determine a trial's success. Examples of successful and failure trials are shown in Fig. 3.4.

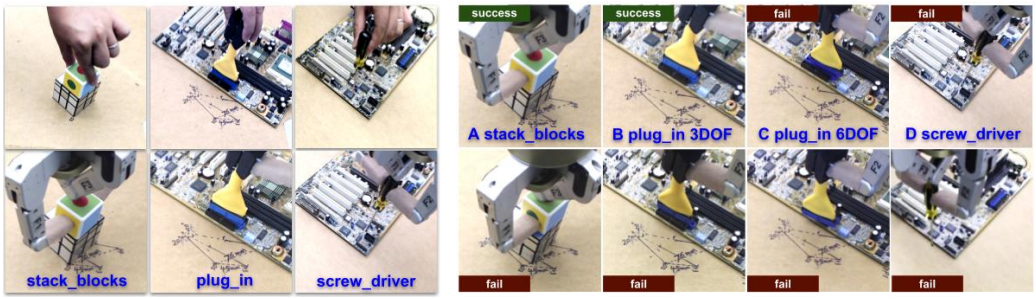


Figure 3.4: Left: Tasks designed. **Right:** success and failure instances.

3.5.2Capability with different tasks

Each task includes ten trials. After each trial, a visual error was manually measured as shown in Fig. 3.3. Results, as shown in Table. 3.1 and Fig. 3.4, indicates, the proposed method performs moderately well in tasks with regular geometric shapes and complex backgrounds, but fails in small visual change conditions and 6DOF tasks. In the screw_driver task, image changes are mostly caused by image noise instead of caused by actual robot motions. This also results in a longer training time. For the 6DOF task, the learned task function is still coarse and the adaptive UVS controller is a local method that's difficult to control with a coarse model.

Task	Training (only images)	Successes	Mean error
stack_blocks	7.6 min	7/10	6.3±2.2
plug_in	9.8 min	6/10	6.3±1.6
screw_driver	13.3 min	0/10	-
plug_in 6DOF	11.84 min	0/10	-

Table 3.1: Evaluation results on different tasks. Training on robots happens only in initial jacobian estimation. avg is 7 seconds which is minimum.

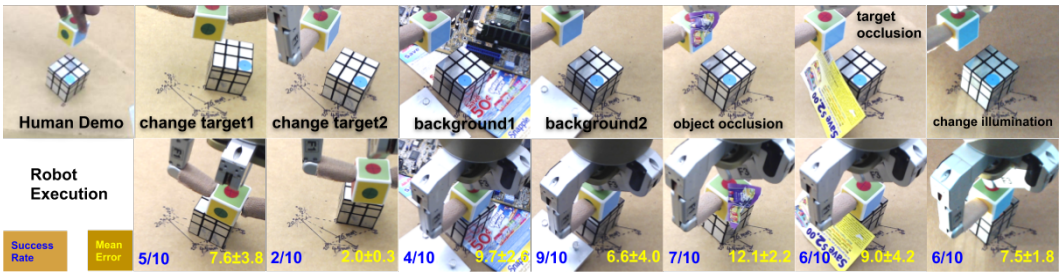


Figure 3.5: Evaluation setup under different task/environment settings. Up row: initial settings; Down row: robot execution results, details are shown in Table 3.2. Results show that it can generalize well under moderately changed target positions and backgrounds, occlusions and illumination changes.

3.5.3Generalization to environmental changes

We tested the following settings (Fig. 3.5): (1) change_target1: The target block is translated 45mm long and rotated with 20°; (2) change_target2: The target block is translated 75mm long and rotated with 40°; (3) background1: Background is changed to a super messy one; (4) background2: Background is changed by adding an extra object; (5) object_occlusion: One face of the small block is occluded; (6) target_occlusion: One face of the target block is occluded; (7) change_illumination: An extra light source is placed opposite to the scene (Fig. 3.5). Each of the seven setting had 10 trials. After each trial, visual error was manually measured following the same rule as stated before.

Results (Fig. 3.5 and Table 3.2) show that it can generalize well under moderately changed target positions and backgrounds, occlusions and illumination changes. Since the state images are further processed using modular subtraction, it's not surprising that environment changes does not affect the performance as much, but the method performs poorly with large target or background changes.

Variant settings	Avg. steps	Successes	Error (Pixel)
(1) change_target1	23	5/10	7.6±3.8
(2) change_target2	22	2/10	2.0±0.3
(3) background1	45	4/10	9.7±2.6
(4) background2	24	9/10	6.6±4.0
(5) object_occlusion	26	7/10	12.1±2.2
(6) target_occlusion	36	6/10	9.0±4.2
(7) change_illumination	30	6/10	7.5±1.8

Table 3.2: Evaluation results of generalization ability.

Demo Num	Training (only images)	Successes	Mean error
1	2.0 min	1/10	10.8±0
5	5.0 min	2/10	16.3±0.5
11	9.8 min	6/10	6.3±1.6

Table 3.3: Evaluation results using different number of human demonstrations. Results show that the performance improves when using more demonstrations.

3.5.4Ablation studies

Performance as Number of Human Demonstrations Increases: We are also interested in performance evaluation with varing numbers of human demonstrations. Using the same task "plug_in", we trained using 1, 5, and 11 human demonstrations, respectively. For each trained model, 10 trials were conducted and visual error was again used as a performance measure. Results are shown in Table 3.3.

Discussion on failures Failure trials mainly come from two aspects: (i) the accuracy of task function, especially in cases with changing target position with patterns unseen in training samples. That's also the reason why increasing human demonstration numbers will improve its performance. (ii) the Jacobian estimation in UVS control since quality of $\hat{\mathbf{J}}_\mathbf{0}$ has a large effect on the control convergence. Jacobian estimation error mostly comes from the strategy of switching between Broyden update and $\hat{J}_0$ re-calibration, which is similar to the *exploration vs. exploitation* problem in Reinforcement Learning.

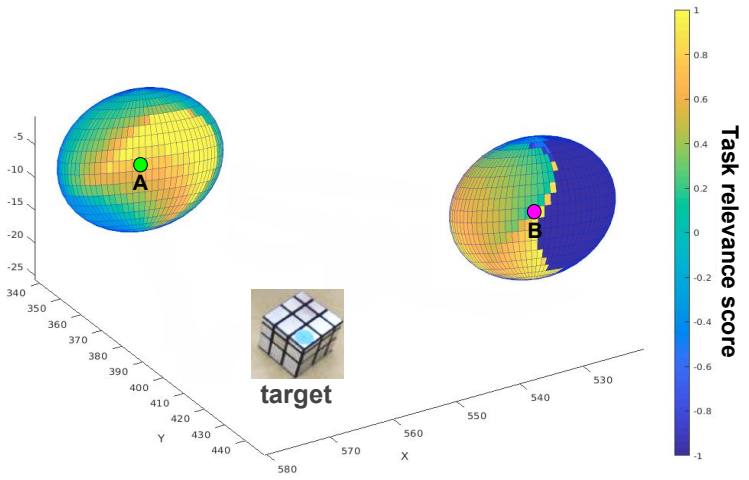


Figure 3.6: Visualization of the learned task function of stacking_blocks task. The colored sphere represents the resulting reward when object moves from the center point (green dot) towards any direction on this sphere surface. The red dot defines the target. Left sphere and right sphere show results on different positions. Results show the region of directions going towards the target will return a **maximum** task relevance score.

3.5.5Visualization and analysis

Visualization of the learned task function: We also visualize the learned task function by collecting samples from a human kinesthetic teaching process. Specifically, we design a virtual sphere as a haptic constraint around a fixed 3D point then human kinethetically move the end effector to collect dense samples inside this sphere. Given the image observation o_t when the end effector locates in the sphere center, we feed the corresponding image o_{t+1} of each sampled end effector positions, then visualize the output task relevance score in a color map. Fig. 3.6 shows the visualization results.

A detailed visualization of Δz_t vector of the *stack_blocks* and *plug_in 3 DoF* tasks are found in Appendix A.1, Fig. A.1.

3.6Summary

This chapter presents InMaxEnt-IRL, an algorithm that uses "temporal-frame-orders" in human demonstration videos to optimize a task function. Our proposed method can directly learn task specifications from raw videos, which

removes the need for hand-engineering. Benefiting from the use of a UVS controller, the training on the real robot only happens at initial Jacobian estimation, which takes an average of 4-7 seconds for a new task. Besides, the learned controller is independent of a particular robot, thus has the potential of fast adapting to other robot platforms. We give an elementary example by using a basic convolutional neural network to parameterize the task function. Various experiments were designed to show that, for a task with certain DOFs, our method can adapt to task/environment changes in target positions, backgrounds, illuminations, and occlusions.

Though some initial success has been achieved, two limitations are worth mentioning. (1) The output vector of the task function has a large variance that deteriorates the success rate of using a UVS controller. (2) Furthermore, with regards to task interpretability, it is still unclear how the output vector of the task function interprets the task definition, especially for high DOF tasks.

Limitations of such a parameterization example give clues to the design of a graph-structured task function in Chapters 5 and 6.

CHAPTER 4

Task specification capability of geometric constraints

TL;DR:

> "Statistical significance test on real-life manipulation dataset is used to study task specification capability of geometric constraints."

In Chapter 3, we've seen a task function based on a conventional image intensity front-end can represent "what is the task" and analyze its drawback as stated in Section 3.6. Then, to further improve the task function's performance, we propose to learn a graph-structured task function that extracts task-relevant geometric constraints since geometric features, compared to raw image pixels, are more salient that give more clues about task-relevant information. Nevertheless, before we go to the algorithm details in Chapters 5, we must first understand the generality of task specification using geometric constraints.

4.1Introduction

In robotics, representing a task using visually observed geometric features (points, lines, conics, planes) has been widely used in traditional visual servoing methods and recently proposed visuomotor policy learning methods. For example, in visual servoing, humans specify a task by selecting what geometric features should be associated to compose geometric constraints. In learning-based methods, using task-relevant key points as the image representation shows advantages regarding sample efficiency and task generalization. However, despite its usage for decades, the task specification capability of geometric features is unclear that in practice, we simply rely on our intuition to decide what and how many features are required for each ad-hoc task. We conclude the problem with three task specification questions. (1) How general that geometric features define a task. (2) Can we use only simple features (points and lines) instead of complex features (conics and planes). (3) How many geometric features are sufficient to define a task. This paper conducts an empirical study on the three task specification questions. By answering the three questions, the generality of task specification using geometric features is validated, and a minimum set of geometric constraints that define a wide range of manipulation tasks is found.

4.1.1Background

Since the dawn of the first robots applied in the automobile production line, researchers have been seeking better ways of robotic task programming [2] towards the ultimate goal of general-purpose robots. The combination of vision, robotics, and artificial intelligence (AI) aims for general-purpose systems [54] instead of ad-hoc systems specifically designed for a particular task. Among the various approaches, using geometric features observed in image planes provides a generative way to construct robotic tasks, which is a possible approach towards general-purpose robotic task programming (Fig. 4.1B).

Task specification using geometric constraints in visual servoing: Research of task specification using geometric constraints spans for nearly 25

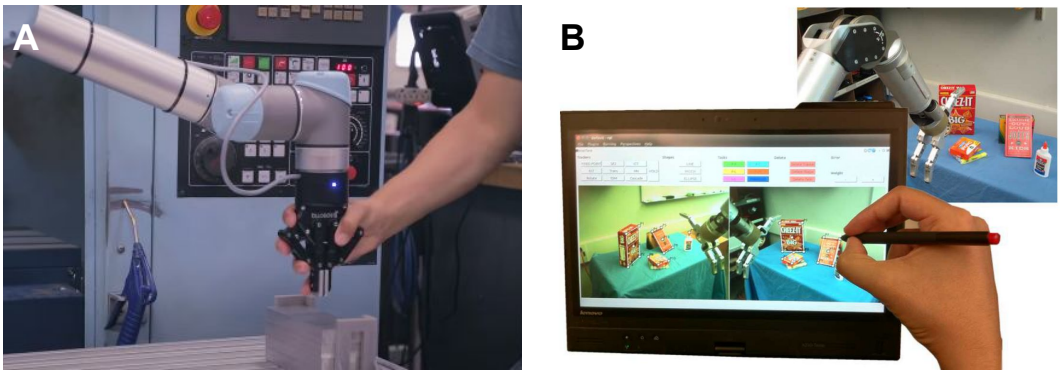


Figure 4.1: Two ways of task programming. **A:** programming a task by coordinates using human kinesthetic teaching, which only works for a simple "reach" motion. **B:**: programming a task by constructing geometric constraints which works for various tasks.

years. Chaumette et al. 1994 [20] propose using basic geometric constraints, such as point coincidence, line alignment and point-to-ellipse alignment, to compose complex tasks by stacking the error vector of each geometric constraint. They furthermore derive the interaction matrix for each type of geometric constraint by assuming a known camera's perspective projection model, which forms an explicitly mapping from observed image features to robot actions, namely the "virtual linkage". Therefore, Chaumette et al. 1994 present a complete system from task programming to controller design in vision-guided robot control, which is named as *visual servoing*. Dodds et al. 1999 [31] and Hager et al. 2000 [54] extend Chaumette's geometry-based task specification to uncalibrated systems. They study how to construct a new task from basic geometric constraints in a more general way. They introduce different task operators [54], including complement, disjunction and conjunction for task composition. Dodds et al. 1999 [29] further extended the generative task construction to a hierarchical approach that enables chaining different geometric constraints under triggering conditions to program more complex tasks.

Task specification using geometric features in robot learning: In recent robot learning approaches, as shown in Fig. 1.5, various methods proposed to use geometric features observed in the image space or point clouds as task-relevant representations. Such approaches do not focus on general-purpose task programming but on learning task-relevant state representations,

for the purpose of sample efficient robot learning and task generalization. The critical insight here is to learn task-relevant keypoints from objects as an efficient and informative image state representation.

For example, in the seminal paper from Levine et al. 2016 [89], using a spatial softmax to extract image points for policy learning is proved a sample-efficient strategy. In KETO [116], using a keypoint generator to extract task-relevant keypoints from point clouds also proves to be a sample efficient and task generalization strategy in reinforcement learning. In dense object descriptors [42], using a camera view-invariant descriptor can help humans specify tasks using point coincidence, point-to-plan to fulfil flexible task specifications. In the following work of kPAM [99], they further extend the descriptor learning to categorical object generalization by the point correspondence built from human annotations. The generalizable descriptor, when further combined with human task specification using point coincide or point-to-plane, kPAM enables task generalization.

A summary of deep learning with keypoints to robotic visual servoing: In summary, although recent advances using keypoint structures in robot learning have achieved significant success regarding sample efficiency and task generalization, however, it is unclear how a task is represented by keypoints or more general geometric features in the observation space. The current research efforts stop at "learning task-relevant keypoints" from the intuition that they should be "good" for visuomotor policy learning but never investigate why these visually observed geometric features can represent a task. In comparison, in traditional visual servoing literature, the study of task specification mechanisms using geometric features is more thorough and in-depth. Such researches, including the theoretical frameworks [20, 54, 59], system [29] and applications [49], are more complete, profound and interpretable that explains clearly such as, how a task is specified using geometric constraints and when such specifications will fail [59].

In all the above methods, however, it is unclear how many tasks can be specified using geometric constraints. Is task specification using geometric constraints a general method or an ad-hoc solution for a specific application?

This is especially important in the era of robot learning that when we apply spatial softmax [89], keypoint bottleneck [84] to extract geometric points or more general geometric features in robot learning since we need to know how many features are sufficient.

4.1.2Three task specification questions

Questions: Specifying robotic tasks using geometric features has long been used in visual servoing [20] and recently proposed learning-based methods [73]. However, three fundamental questions are hanging.

- (1) How many tasks can we specify using geometric features, namely, is task specification based on geometric features a general and scalable method, or just an ad-hoc approach?
- (2) Empirically, we know that using only points and lines can represent more complex geometric features, such as conics, planes. Then does task specification using only points and lines also have the same capability as using general geometric features?
- (3) Assume using only points, and lines can also define a wide range of manipulation tasks, then what is the minimum set of points, lines forming constraints that define a reasonable range of tasks? Such a minimum set gives us an estimation of the complexity upper bound regarding how many and how complex geometric constraints are needed for a task specification.

The significations: why are these questions important?

The first question concerns the generality and scalability of task specification using geometric features.

The second question concerns the possibility of using a neural network to represent the geometric constraints. For example, suppose we can prove that using merely points and lines is sufficient. Thus complex features are not needed. Then, it will facilitate the design of a neural network since the descriptor for complex geometric features in the raw image or point cloud data is still

challenging to get in both hand-engineered and deep learning-based methods. As a result, answering this question will make learning geometry-based task specifications possible, which will be further explored in the following chapters.

The third question concerns the complexity upper bound when designing such a neural network, which helps us estimate how many constraints are sufficient to represent a task. For example, if the number is insufficient, the designed neural network's representation power will be inadequate. Meanwhile, if the number is overly large, then the neural network will have a vast parameter size resulting in extra training difficulty.

The challenges: To answer the questions, one evident approach is to construct a pool of tasks as the test set firstly. Then we can try decreasing or increasing the number and complexity of geometric constraints to get a set $\mathcal{G}$. Successively we can use a visual servoing controller to test the corresponding success rate, which will give us the answers.

Two challenges make it impossible in practice. First, the process of constructing a pool of tasks has lots of subjective factors. For example, the lab bias indicates that researchers tend to build tasks they think should be common in real-life environments. Therefore, we need a pool of more objective tasks. Second, it is an impossible task considering time and labour cost. For example, suppose we are testing N geometric constraints by changing the number of complexity, $N = 1000$. Furthermore, we try them on M tasks, $M = 500$. For each test, we need to run k times to get an average success rate, $k = 10$. Then the total number of trials is $N * M * k$, which are 5.0 million tests that are impossible to finish in a reasonable amount of time. As a result, we apply a similar approach as to [14] that uses real-life datasets to have human raters empirically evaluate the task specification capability of geometric constraints.

4.2Related works

Task specification capability of geometric constraints: There are very few researchers studying the task specification capability of geometric con-

straints. Hespanha et al. 1999 studies the problem of how a task can be visually decidable by observing the geometric constraints considering different camera models, such as injective cameras, weakly calibrated projective cameras and uncalibrated projective cameras. They analyze the relationship between visually decidable task sets of the three camera models. However, they do not study what is the range of tasks that can be specified using geometric constraints. The recently published real-life task dataset enables an empirical study of this problem.

A review of manipulation datasets: Ego-centric video footage of human activities of daily life (ADL) provides useful samples for the study of robotic taxonomy, human attention mechanism during task execution, semantic task representation learning, and benchmarking different robot learning algorithms if the dataset is augmented with robot sensory data and actions. Various datasets have been proposed. Fathi et al. 2012 propose GTEA Gaze+ [36] dataset containing a large number of human egocentric activity videos for the study of human gaze-action coordination mechanisms. Bullock et al. 2015 [14] propose Yale_GRASP showing daily human activities in a household and a machine shop environment. Goyal et al. 2017 [46] propose 20BN-something-something dataset with more general human activity recordings restricted to manipulation task videos. Mandlekar et al. 2019 [98] propose RoboTurk, which is originally an offline dataset for benchmarking robot learning algorithms. In this paper, we use only the robot's videos to study the task specification capability of geometric constraints. Huang et al. 2019 [62] propose USF_DIM dataset, which contains both vision and force/torque sensor readings.

A review of manipulation taxonomy: There are tremendous research efforts to build a proper robotic task taxonomy in the research community. However, building a general robotic task taxonomy involves finding a general way to specify robotic tasks, which is intimidating. As a result, published research mainly focuses on one aspect of robotic tasks. Two common aspects

are taxonomy for robotic grasping and robot motions.

For example, the taxonomy of robotic grasping has attracted a lot of research focus. Works are proposed to build a complete grasping action taxonomy for everyday lives. Therefore, real-life datasets and human empirical evaluation techniques are used. Liu et al. 2014 [93] uses data annotations on pre-recorded video clips about activities of daily livings to study how to build a complete grasp of taxonomy. Bullock et al. 2015 [14] studies the grasping taxonomy by having multiple raters on a pool of tasks that are record using a head-mounted camera. Another two examples are in the taxonomy of robot motions. Zech et al. 2019 [166] introduce a taxonomy for robot motions by a comprehensive literature survey. Paulius et al. 2020 [109] build motion codes for general robotic tasks by annotating datasets collected using sensors mounted on the task objects. The methodology of this research aligns with the works mentioned above in that we use real-life datasets to empirically evaluate the task specification capability of geometric constraints.

4.3Methods

4.3.1Overview

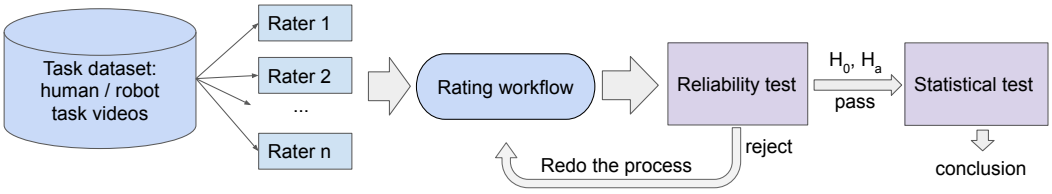


Figure 4.2: Overview of test procedures.

Benefiting from the recently proposed manipulation datasets [14, 62, 98, 101], we construct our pool of tasks coming from real-world environments (household, machine shop, kitchen, lab). Therefore, the test results will reflect the distribution of tasks for real-world demands.

Furthermore, we borrow the same approach from the early research on building a robotic taxonomy [14] by statistical significance tests on different human raters' responses to each task in the pool. Using raters' responses

can significantly reduce the test time cost, makes the test doable. Fig. 4.2 shows the test procedures. Sections below explain each module shown in the diagram.

Figure 4.3: Example tasks in the dataset. **A**: Kitchen environment from USF_DIM [62]. **B**: Hand tool workshop environment from USF_DIM. **C**: Lab environment from RoboTurk[98]. **D**: Machine Shop environment from Yale_GRASP [14]. **E**: Household environment from Yale_GRASP. Full example lists are attached to the thesis. Red dots indicate point-to-point alignment constraints and green lines line alignment constraints.

4.3.2Task pool construction

We use three manipulation datasets to construct a pool with 393 different tasks from various environments, such as kitchen environment, hand tool workshop environment, lab environment, machine shop environment and household environment. The three datasets are:

- USF_DIM [62] was proposed by Huang et al. 2019 with human task videos showing 18 tasks in the kitchen environment and 6 tasks in the hand tool workshop environment.

Environment	Data source	Videos	Duration	Tasks	Av. VPT	Av. TPV
Kitchen	USF_DIM	599	1h 49m 49s	18	33	1
Hand tool	USF_DIM	205	0h 37m 35s	6	34	1
Lab	RoboTurk	30	0h 52m 30s	78	1	2
Machine Shop	Yale_GRASP	67	14h 54m 0s	186	3	14
Household	Yale_GRASP	67	14h 54m 0s	105	3	15

Table 4.1: Statistical analysis of the dataset we used. VPT: number of video clips per task. TPV: number of tasks per video clip. An example of detailed task list in USF_DIM dataset is shown in table C.3

- RoboTurk [98] was proposed by Mandlekar et al. 2019 with robot task videos showing 78 tasks in the lab environment.
- Yale_GRASP [14] by Bullock et al. 2015 with human task videos showing 196 tasks in the machine shop environment and 118 tasks in the household environment.

Table. 4.1 shows a detailed analysis of the three datasets we use to construct our pool of tasks. Fig. 4.3 Shows task examples from each dataset. A full analysis of all the tasks in our major dataset source Yale_GRASP [14] is included in Appendix C, Fig. C.1 and C.2.

4.3.3Participants

We select two raters from diverse backgrounds. Rater A has experience in robotics engineering, hand tool usage and household tasks. Rater B does not have any experience in robotics engineering but with experience in hand tool usage and household tasks. Both rater A and B are trained with machine shop experience by watching the "Yale_GRASP" video footage and their annotations.

4.3.4Response format for raters

Given an evaluation objective, for example, evaluating the task specification capability using general geometric constraints, each rater is required to go through all the tasks in the pool and respond if each task can be specified using a geometric constraint.

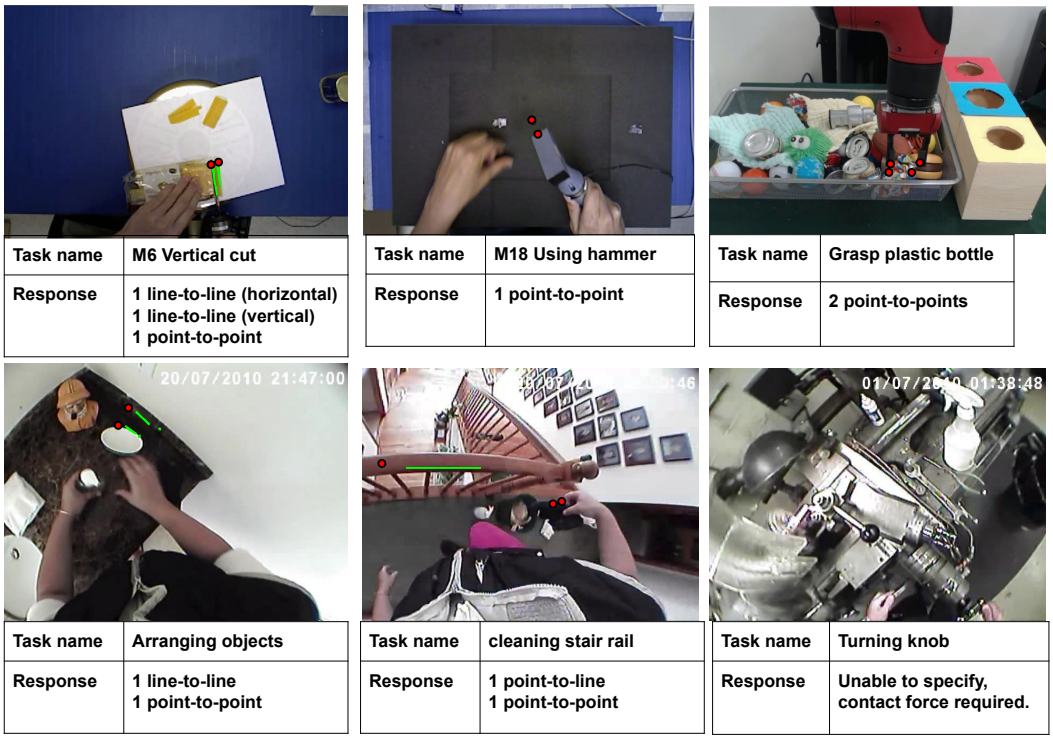


Figure 4.4: Examples of rater B's response in evaluating the general geometric constraints on the household environment from YAL_GRASP dataset. Rater B is more optimistic about the task specification capability of geometric constraints. Using human raters for an empirical evaluation may result in either over-optimistic or under-pessimistic estimation. An ultimate solution is to design tasks in simulation and let a visual servoing controller decide, however, this approach is impractical as stated in Section 4.1.2.

Then the rater is required to annotate the dataset with his/her response by recording the response in a table and drawing what kind of geometric constraints define that task on a video image. If the response is "unable to specify", the rater needs to record reasons in the table. Fig. 4.4 shows examples of the rater's response on tasks from the five environments.

Table 4.2 shows examples of raters' responses in evaluating the general geometric constraints on the household environment from YAL_GRASP dataset.

4.3.5Visual task decidability

How should a rater tell if a task can be specified by a certain type of geometric constraint? Before rating, each rater is trained on the basics of how geometric constraints can define a task.

Video #	Task Name	Description	Response	Comment
156	adjusting faucet	touching the faucet and pull	1	1 point-to-point
141	adjusting object	touching the two points and adjusting	1	2 sets of point-to-point
49	aiming at surface	aiming the nozzle to the surface	1	multiple point-to-points
41	wiping faucet	wiping	1	1 point-to-point, 1 point-to-line
130	arranging	grasping point, placing inside the net	1	2 sets of point to point
177	arranging furniture	arranging the chair in good pose	1	1 point to point, 1 line to line
41	arranging object	arranging object along the counter edge	1	1 point-to-point, 1 line-to-line
43	cleaning floor	picking up the two rugs	0	rugs are hard to be described using points
44	cleaning mop	rising the mop cloth under the faucet	0	it is hard to describe how to brush and make sure it's clean.
44	cleaning stair rail	swiping along the rails	1	1 point-to-point 1 point-to-line

Table 4.2: Example of a rater's response in evaluating the task specification capability of general geometric constraints using the YAL_Grasp dataset [14] with the household environment.

Suppose a task $T \in \mathcal{T}$, where $\mathcal{T}$ is task space (pool of tasks), and a set of geometric constraints $\mathcal{G}$ is given for evaluation. If the rater responds that T can be defined using $\mathcal{G}$ by geometric features from an image, we say T is visually decidable by $\mathcal{G}$. Here we don't consider the visual ambiguity problem and assuming multiple view cameras can be easily set up at an arbitrary location for the task.

4.3.6The rating workflow

Fig. 4.5 shows a rater's workflow. First, a rater receives basic training regarding the definition of geometric constraints and how to mark them on a task image using graphic tools visually. Then the rater needs to go through each geometric constraint test on each task. Note that, in the dataset, each task has multiple video clips. A randomly selected video clip is presented to the rater and lets the rater decide whether a task is doable or not by the given geometric constraints. If the rater provides a positive answer, they need to

explain in texts and draw them on the task video frame. If the answer is negative, the rater needs to provide a reason. Fig. 4.5 shows a rater's rating process.

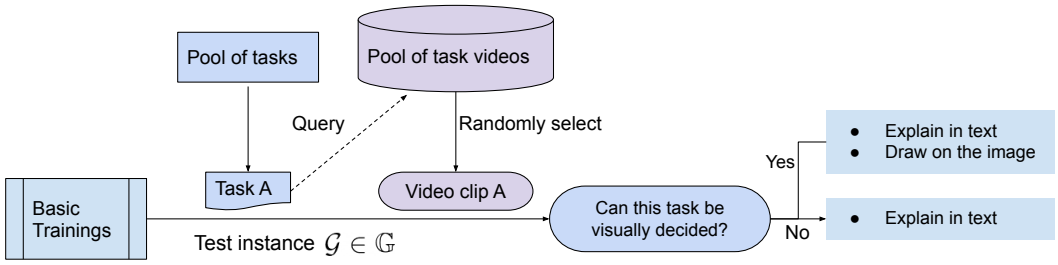


Figure 4.5: A rater's rating workflow.

4.3.7Raters' reliability test

Unlike using a real robot for the test, the human rater may be over-optimistic or pessimistic depending on how much robotic engineering experience they have. So there are subjective factors in the rater's response. To further eliminate such factors, similar to [14], we introduce an additional reliability test. If the reliability test won't pass, the rating workflow needs to be redone from the start.

The reliability test is undertaken by considering raters with different backgrounds and statistically measuring all the raters' decision alignment by a Cohen's kappa coefficient κ, computed from the agreement matrix using each rater's response for all the tasks. If the statistics $\kappa < \beta$, where β is a threshold value, the reliability test will fail. Following [14], we set $\beta = 0.6$ in our tests.

4.4Evaluation results

4.4.1Experiments design

Metrics: The *task coverage* metric is used to evaluate geometric constraints' task specification capability. Specifically, given a pool fo tasks $\mathcal{T}$, a *task coverage* η is defined as $\eta = \frac{n}{N} * 100\%$, where $N = |\mathcal{T}|$ is the size of the task space and n is the number of tasks that are visually specified by the geometric constraints.

Test using general geometric constraints: The evaluation objective is to answer the first question raised in Section 3.1.1—is task specification using geometric constraints a general method or just an ad-hoc method?

For convenience, let's denote set $\mathbb{G}_{all}$ as all the possible combinations or sequential linkage of general geometric constraints. The evaluation process of a rater is defined as:

- For each task $T \in \mathcal{T}$, the rater is required to follow the rating workflow as described in Section 3.3.6.
- The rater records response by deciding if any subset of $\mathbb{G}_{all}$ can be used to describe the task T visually.
- Compute the average task coverage η_{all} which marks the task specification capability using general geometric constraints.

Then, the null hypothesis H_0 in this test is: there is no relationship between any subset of $\mathbb{G}_{all}$ and the task specification of an arbitrary task T. The observation of successful task specifications using a subset from $\mathbb{G}_{all}$ is because of randomness.

Suppose the resulting task coverage using general geometric constraints is η_{all}. Formally, the null hypothesis and alternative hypothesis can be written as a one-tailed tests:

- Null hypothesis H_0^{all}: $\eta_{all} < \tau_{all}$
- Alternative hypothesis H_a^{all}: $\eta_{all} \geq \tau_{all}$

, where the resulting τ_{all} under a significance level threshold is the task coverage using general geometric constraints $\mathbb{G}_{all}$.

Test using points lines constraints: Likewise, the evaluation objective is to answer the second question raised in Section 3.1.1—is task specification using only points and lines also a general and scalable method?

For convenience, let us denote set $\mathbb{G}_{pl}$ as all the possible combinations or sequential linkage of point-to-point, point-to-line, line-to-line constraints. The evaluation process of a rater is defined as:

- For each task $T \in \mathcal{T}$, the rater is required to follow the rating workflow as described in Section 3.3.6.
- The rater records response by deciding if any subset of $\mathbb{G}_{pl}$ can be used to describe the task T visually.
- Compute the average task coverage η_{pl} which marks the task specification capability using general geometric constraints.

Suppose the resulting task coverage using point and line constraints is η_{pl}. Formally, the null hypothesis and alternative hypothesis can be written as a one-tailed tests:

- Null hypothesis H_0^{pl}: $\eta_{pl} < \tau_{pl}$
- Alternative hypothesis H_a^{pl}: $\eta_{pl} \geq \tau_{pl}$

, where the resulting τ_{pl} under a significance level threshold is the task coverage using general geometric constraints $\mathbb{G}_{pl}$.

4.4.2Find the minimum set of geometric constraints:

The evaluation objective is to answer the third question raised Section 3.11 —what is the minimum set of point and line constrain that can define a reasonable range of tasks? This question setting is different from the above two in that we are required to give the specific task coverage for each possible combination using point-to-point, point-to-line and line-to-line constraints.

Test set construction: We consider basic the geometric constraints using only point and line as following. (1) A point-to-point constraint denoted as "PP" that defines two-point coincidence. (2) A point-to-line constraint denoted as "PL" that defines a point fits in a line. (3) A line-to-line constraint denoted as "LL" that defines two lines in parallel. Then a test instance is constructed by different numbers of such constraints, in the form "[i] PP; [j] PL; [k] LL". For example, "1PP;0PL;3LL" defines a task with one point-to-point constraint

#	num of constraints	PP num	PL num	LL num	remark
1	1	1	0	0	1PP;0PL;0LL
2	1	0	1	0	0PP;1PL;0LL
3	1	0	0	1	0PP;0PL;1LL
4	2	2	0	0	2PP;0PL;0LL
5	2	1	1	0	1PP;1PL;0LL
6	2	0	2	0	0PP;2PL;0LL
7	2	1	0	1	1PP;0PL;1LL
...	...	...	...	...	...
342	18	6	6	6	6PP;6PL;6LL

Table 4.3: Test instance examples of $\mathbb{S}$.

and three line-to-line constraints. Obviously, the above test instances have an infinite number of combinations.

For simplicity, we consider each test instance can use a maximum number of 6 for each geometric constraint type, $0 \leq i, j, k \leq 6$. Therefore, the total number of test instances is $7 * 7 * 7 - 1 = 342$. Following this way, we construct our test set $\mathbb{S}$, where $|\mathbb{S}| = 342$. Examples in the test set are given in table 4.3.

Evaluation process: The evaluaton process of a rater is defined as:

- For each task $T \in \mathcal{T}$, the rater is required to follow the rating workflow as described in Section 3.3.6.
- For each geometric constraint $S \in \mathbb{S}$, the rater records response by deciding if using S can sufficiently describe the task T.
- Compute the average task coverage η_S for each $S \in \mathbb{S}$.

Results reporting protocol: In order to efficiently report the 342 test results in a succinct way. We categorize the 342 test instances by their total number of constraints used, as summarized in Appendix C, table C.2 We report only the highest η_S value among all the test instances per category for convenience.

Reduced task pool: Since there are 342 test instances and 393 tasks, the total number of experiments will be $342 * 393 = 134,406$, which is almost im-

possible to finish in a reasonable time. As a compromise solution, we randomly sample 20 tasks from the pool, with five tasks sampled from each dataset[1]. Each geometric constraint will then be evaluated using the select-out 20 tasks.

Test hypothesis: Given a test instance $S \in \mathbb{S}$, we want to know its average task dicidability η_S on the randomly selected 20 tasks. Suppose the resulting task coverage using the test geometric constraint S is η_S. Formally, the null hypothesis and alternative hypothesis can be written as a one-tailed tests:

- Null hypothesis H_0^S: $\eta_S < \tau_S$
- Alternative hypothesis H_a^S: $\eta_S \geq \tau_S$

For each of the 342 test instances, we construct the above H_0^S and H_a^S hypothesis test. To reduce workload, we are only interested in results considering $\tau_S = \tau_{pl}$, which means we are interested in finding the minimum number of constraints used that has an equal task specification capability of using points, lines constraints without number limitations. The intuition is to find the minimum number of geometric constraints that are competent to them.

Therefore, suppose k geometric constraints $\{S_1, ..., S_k\}$ all have a task coverage equal to τ_{all} and τ_{pl}, the minimum number of constraints used among them is the minimum set we want to find.

4.4.3Results

Capability using general geometric constraints: Following the rating process defined in 3.1.2, rater A responds using general geometric constraints can define a task covarege of $\eta_{all} = 79.3\%$, while rater B reports the task coverage of $\eta_{all} = 84.4\%$.

Then we conduct the reliability test. Firstly, we construct the agreement matrix, as shown in Table. 4.4, by listing rater A and B's positive/negative response of each task. Then, we compute the Cohen's kappa coefficient $\kappa = 0.63 > \beta$, which means both raters' reponses are accepcted.

[1]Yale_GRASP dataset is divided into household and machine shop dataset since it is too large.

Rater A \B	Positive	Negative
Positive	299	12
Negative	32	50

Table 4.4: Agreement matrix of rater A and B in testing the task specification capability of $\mathbb{G}_{all}$.

Then we take the average $\eta_{all} = 81.9\%$ and perform Z-test using a sample size of 393, compute the correponding p-values given different τ_{all} ranging from $[0, ..., 100\%]$ with an incremental step size 0.1%. Plots are shown in Fig. 4.6. Considering the significance level $\alpha = 0.05$, we conclude that with $\tau_{all} = 78.3\%$, the corresponding p-value $p\text{-}val < \alpha$, therefore reject the null hypothesis in favor of an alternative.

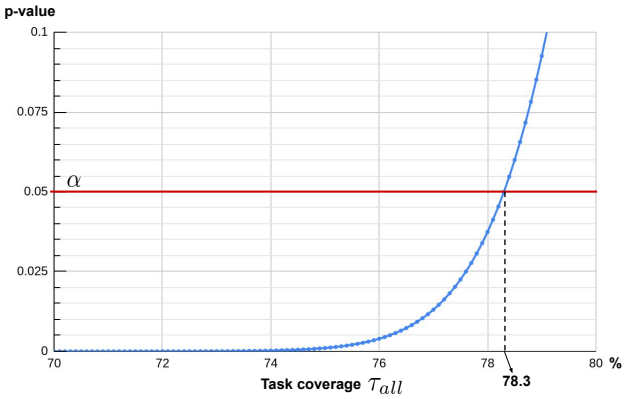


Figure 4.6: Z-test results using general geometric constraints $\mathbb{G}_{all}$ given different τ_{all} as shown in the horizonal axis.

Conclusion: using $\mathbb{G}_{all}$ for task specification covers at least 78.3 % of tasks in the pool with p-value < 0.05.

Capability using point and line constraints: Following the rating process defined in 3.1.2, rater A responds using only point and lines geometric constraints can define $\eta_{pl} = 77.9\%$ tasks in the pool, while rater B reports $\eta_{pl} = 84.4\%$.

Then we conduct the reliability test. Firstly, we construct the agreement matrix, as shown in Table. 4.5, by listing rater A and B's positive/negative response of each task. Then, we compute the Cohen's kappa coefficient

$\kappa = 0.615 > \beta$, which means both raters' responses are accepted. Each rater's detailed response including visual decidability marks on each task failed reasons and visually marked geometric constraints are included as attachments to this thesis.

rater A \B	positive	Negative
Positive	297	12
Negative	34	50

Table 4.5: Agreement matrix of rater A and B in testing the task specification capability of $\mathbb{G}_{pl}$.

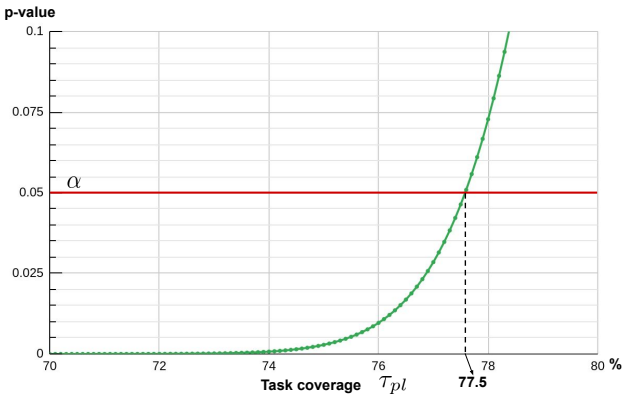


Figure 4.7: Z-test results using point and line constraints $\mathbb{G}_{pl}$ given different τ_{pl} values as shown in the horizonal axis.

Then we take the average $\eta_{pl} = 81.2\%$ and perform Z-test using a sample size of 393, compute the correponding p-values given different τ_{pl} ranging from $[0, ..., 100\%]$ with incremental step size 0.1%. Plots are shown in Fig. 4.6. Considering a significance level of $\alpha = 0.05$, we conclude that with $\tau_{pl} = 77.5\%$, the corresponding p-value $p - val < \alpha$, therefore reject the null hypothesis in favor of an alternative.

Conclusion: we find that using $\mathbb{G}_{pl}$ covers at least 77.5 % of tasks in the pool with p-value < 0.05.

Fiding the minimum set of geometric constraints: Following the rating process defined in 3.1.2, two raters respond results using each of the 342 test

instances S from $\mathbb{S}$ on the 20 tasks and report the average task decidability η_S for each S.

Then we categorize the 342 results by the total number of geometric constraints used in each test instance and report only the best η_S value per category. After that we perform Z-test using a sample size of 20, compute the corresponding p-values given $P_0 = 78.3\%$ and $P_0 = 77.5\%$ respectively. Plots are shown in Fig. 4.8, wherein the horizontal axis represents a total number of geometric constraints used in the test and the vertical axis is the p-value. Table. C.1 in Appendix C shows the number of test instances under each category and detailed results of this test.

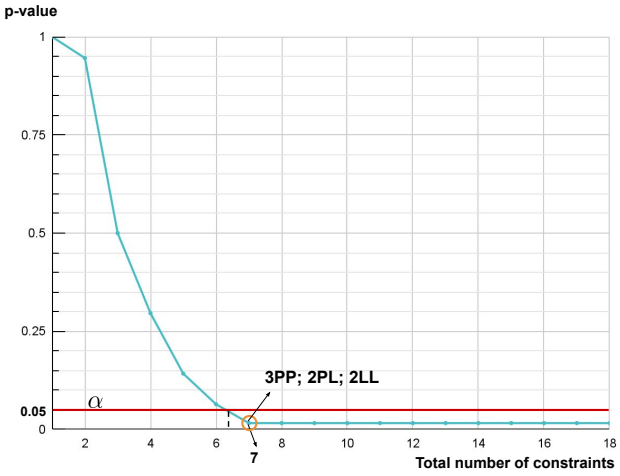


Figure 4.8: P-values different number of used geometric constraints in the test using $\tau_S = \tau_{pl}$ which means we want to find the minimum number of constraints that has the same task specification capability with $\mathbb{G}_{pl}$ that has no number limitations.

Considering the significance level $\alpha = 0.05$, we conclude that using a total number of 7 geometric constraints ("3PP;2PL;2LL") with $\tau_s = \tau_{pl}$, the corresponding p-value $p\text{-}val < \alpha$.

Conclusion: we find that using three point-to-point, two point-to-line, and two line-to-line constraints matches the task specification capability of using $\mathbb{G}_{pl}$ with p-values < 0.05.

4.5Summary

This chapter studies the task specification capability using geometric constraints. Statistical significance test on real-life manipulation dataset is used based on a human rating approach. We find that:

- Using general geometric constraints $\mathbb{G}_{all}$ that use an unlimited number of constraints, and complex geometric features cover 78.3 % of tasks in the pool with p-value < 0.05.
- Using only points-lines constraints $\mathbb{G}_{pl}$ without limiting the constraint number covers 77.5% of tasks in the pool with p-value < 0.05. This means task specification using only point and line is competent with general geometric constraints.
- The minimum geometric constraint set is found, which consists of three point-to-point, two point-to-line and two line-to-line constraints. These few constraints also solve 77.5% of tasks, hence match the task specification capability using $\mathbb{G}_{pl}$ with p-value < 0.05.

These findings will guide the design of a graph-structured task function in Chapters 5 and 6.

Part I

Geometric Task Representation Learning

CHAPTER 5

A geometric perspective on visual imitation learning

TL;DR:

> "Geometry-structured task representation which defines 'what is the task', is parameterized by relational encoders (graph neural network), and learned from human demonstration videos assuming the temporal-frame-orders."

As discussed in Chapter 4, section 4.4, in the visual imitation learning problem following a "what–how" decoupled approach, different choices of the task function design affect the successive robot controller. Therefore, this chapter introduces geometric feature-based task specification from visual servoing to the task function design. Specifically, we propose geometry-structured task representation learning, which introduces a graph structural priori to the task function design, and enables the neural network to extract task-relevant interconnections between geometric features.

5.1Introduction

Compared to traditional robotic task teaching methods, such as kinesthetic teaching and teleoperation, visual imitation learning promises a more intuitive way for general-purpose task programming. Like most other learning methods, the requirement of a large number of robot-environment interactions impedes its deployment in real-world robotic tasks. This is regarded as the sample efficiency problem, moreover, in real-world applications, training sample size obtained on the robot is our critical concern.

Commonly, three strategies are used to tackle such sample efficiency problems. The first one is to increase the number of human demonstrations via kinesthetic teaching or teleoperation. This has been proven effective for supervised learning methods such as behaviour cloning [7]. However, it requires long and tedious human supervision work. The second strategy is to assume access to robot-environment interactions where more samples can be explored via reinforcement learning (RL) methods (e.g. IRL [6], GCL [37], GAIL [60]). Unfortunately, new issues regarding transfer learning and low sample efficiency arise during both simulation and real-world training. The last strategy assumes that shared knowledge can be learned from demonstration samples across multiple but similar tasks; from this shared knowledge, the robot is able to learn a new task when given one more demonstration. This strategy is used in meta-learning-based approaches (e.g. one-shot [40]), however, thousands of human kinesthetic teaching and specially designed ad-hoc robot controllers are still required.

Generally, the aforementioned methods use human demonstration as *state-action* samples to learn a policy mapping from image to action (i.e. they approximate a target state-action distribution). Consequently, in order to cover the whole state-actino space, it is necessary to collect more state-action experiences from either human teaching (supervision) or robot self-exploration (RL training). However, neither approach proves satisfactory. This motivates us to ask the question: *is it possible to learn by watching only a few human demonstrations without the extra effort of interactive training?*

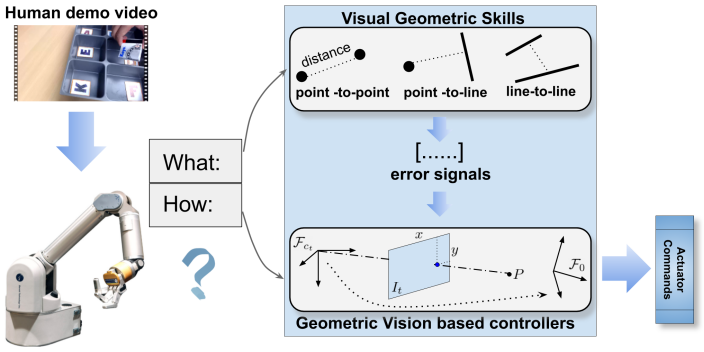


Figure 5.1: Rethinking the classical 'correspondence' problem [8] in imitation learning reveals two essential questions: i) '*what*' information should be transferred from a human demonstrator to a robot imitator; and ii) '*how*' can this information be used to bring about actions (i.e. motor action). This paper presents a geometric perspective to this problem. We find a proper geometric representation of '*what*' that facilitates the training of '*how*'.

Recently, several methods have been proposed to tackle this question. One *key insight* is to rethink the classical 'correspondence' problem [8] which studies the difference between demonstrator and imitator. Such insight changes our view on human demonstration: what if more task concepts, instead of control, were encoded from the demonstrated data? Empirically, this aligns with our cognitive process in peer learning which involves first understanding the task before attempting any motor actions[1]. This hierarchical view decouples learning the 'what' and 'how' (zero-shot [25, 108], see Fig. 5.1). Benefits of this method are immediately observed: i) the promise to sample efficiency since it learns a high-level cognitive concept [71, 72, 134] of the task instead of directly matching state-action distributions; and ii) the promise of reusable low-level policies as basic skills across different tasks [75]. However, two new problems arise: (1) what is the high-level task; and (2) how can we train the low-level controllers without an additional intensive cost.

In this paper, we provide a geometric perspective to derive solutions. We show that, instead of learning from image pixels to actions, learning a geometry-parameterized task concept[2] provides an explainable and invariant

[1]This is studied in observational learning [15] in psychology.

[2]For further reading, task parameterization using geometric constraints (e.g. point-to-point, point-to-line, point-to-conics, etc.) are intensively studied in [29–31, 55].

representation across demonstrator to imitator under various environmental settings. Moreover, it provides *controllability* that can be directly linked to geometric vision-based controllers (e.g. visual servoing).

we propose VGS-IL (visual geometric skill imitation learning), an end-to-end geometry-parameterized task function approximation method, to infer globally consistent geometric feature association rules from human demonstrated video frames. Instead of learning actions from image pixels, we show that learning a geometry-structured task function provides an explainable and invariant representation across demonstrator to imitator under various environmental settings. Moreover, unlike prevalent methods requiring hierarchically training of an additional control policy [40–42, 108, 134], we show that such a geometry-structured task representation provides a direct link with geometric vision-based controllers (e.g. visual servoing), allowing for efficient mapping of high-level task concepts to low-level robot actions.

5.2Related works

Visual imitation learning : The problem defined in visual imitation learning is: given one or several human demonstration videos, how can a new task be learned? Research on this topic dates back to 1994 [66, 85]. With the rise of deep learning and reinforcement learning, more influential works have since been published. While some are reviewed in Section 5.1, which aim to learn a task from visual inputs, it's worth noting another research stream aiming to learn a *semantic knowledge* representation. This method commonly relies on independent pipelines like object detection, action recognition etc. Despite of their method complexity, experiments show they can learn semantic task plans that follow a procedural manner [3, 158, 161].

Hierarchical visual imitation learning : Instead of simultaneously learning task definition and control, hierarchical approaches decouple the two by focusing on learning a shared high-level task representation across human demonstrator and robot imitator. The two core problems are: i) how to rep-

resent the high-level task concept; and ii) how to train the low-level control policy. The first one is more important since representation of the task concept determines the controller training. For example, many pioneer works parameterize the task concept in *pixel level* by using sub-goal output from a neural network [108, 134]. The low-level policy is then sub-goal conditioned and trained following a Hierarchical Reinforcement Learning manner.

More recent works represent a task in the *object level* where object correspondence [41, 42] or graph structure [135] relationships are utilized to parameterize a task. The low-level controller is then trained based on distance errors in the embedded parameterization space. This approach shows success in pushing and placing tasks, however, it lacks the definition resolution required for more complex tasks like insertion.

Geometry approach visual imitation learning : Alternatively, going deeper inside the objects, *geometric feature level* based approaches arise. Early pioneer works from Ahmadzadeh et al. 2015 [3] proposed VSL to learn feature point correspondence based task representation given one human demo video. A similar approach from Qin et al. 2019 [116] presents KETO which utilizes key point relationships to represent a tool manipulation task. In general, their low-level controllers are tediously trained separately without enough study emphasis on how a proper task representation will facilitate the low-level policy training.

Beyond a simple key point correspondence based task concept representation, other basic geometric constraints (point-to-line, line-to-line, etc.) can enrich our toolbox for parameterization of task concepts [50]. Furthermore, by concurrently combining and sequentially linking them [29], we can find a general way to program more complex manipulation tasks that exhibit scalability. To the authors best knowledge, applying such systematic geometry-based task programming in visual imitation learning is rarely studied.

Graph-structured knowledge representation learning in robotics: Previous studies show that, given a 3rd view human demonstration video, we can

extract semantic meaning of the task which is represented as a graph or tree structure [158, 161]. Based on terms from Ontology, each entity in the graph represents an object, the associated motion primitives represents object affordance and the relationship between entities defines a procedural task [110]. Such approaches also have a long history of research ranging from the two pioneering papers on learning by watching human desmontraion [66, 85], to "things are made for what they are" [88], and recent works on *functional object networks*(FON) [111].

These approaches are good at representing a procedure task but is extremely tedious in training, which relies on object detection/segmentation, frame by frame human annotation. Moreover, the mapping from a task plan to robot actions hugely relies on prerecorded motion primitives. How to adapt them to task initial conditions changes remains challenging. While most approaches focus on how to extract semantic meaning of a task, very few works have been done in studying how to **extract the geometric meaning** of a task.

Scene graph representation in computer vision: The closest literature to our proposed modelling geometric feature connections of an image as graph-structured task definitions is the scene graph representation learning in computer vision tasks, such as image captioning [5], visual question answer [76]. Scene graph approaches also view an image as a structured graph where each entity is detected objects [23, 51, 97, 159, 167] and edges representing their relationships (spatial or verbal). The relationships are predefined predicates commonly modelled as message passing in a graph convolutional neural network—and learning based on clustering similar relationship objects together by finding the smallest bounding box between two objects. Scene graph approaches share the same viewpoint as ours — images are more than just a collection of objects and attributes. The relationship of interconnected parts of an image makes sense in visual understanding. However, the relationship types of scene graphs are restricted to predefined predicates [32], as compared to ours, we have more flexible tools to define spatial relationships. Besides, the

bounding box-based approach of scene graphs lacks the description accuracy to define manipulation tasks. In our method, each graph instance encodes the geometric structure of a task and has a geometric loss conjugate, a natural signal that guides robot control.

Neural relational encoders: Commonly, approaches that encodes relational inductive bias can be classified into two categories. (1) Methods that encodes only local dependency, for example, recurrent neural networks (RNNs) and long-short-term-memory (LSTM [61]) that are applied to sequential data encoding. (2) Methods that encodes both local and non-local dependencies, for example, self-attention based methods (transformers [144, 153] and visual transformers [57] in computer vision tasks) and graph neural networks [154]. Battaglia et al. 2018 [11] gave a good summary of the above approaches. We are more interested in methods that are capable of both local and non-local dependency encoding, specifically, we investigate visual transformers and graph neural networks.

Visual transformers use a dot-product based similarity score to measure pair-wise relationships which is quite efficient in computing since dot-product vectorization, however huge parameters introduced since the heavy pair-wise relationship encoding [52]. So visual transformers typically rely on large data corpus to train [162]. Since we want to extend our previous work to encode relationships between pixel-wise vectors in a dense visual descriptor X, pair-wise entity relationship computation will be intimidating. In contrast, graph neural networks directly encode any relationships based on a pre-defined structure[3] The aggregation based node-edge relationship encoding [154], compared to similarity based self-attention, collects messages from all connected nodes and edges, thus reserves permutation invariance. Our approach utilize a graph neural network to directly encode relationships other than a similarity based self-attention module commonly used in transformers.

[3]There are also preliminary researches studying dynamically changing graph structures [122, 138].

5.3Preliminaries: graph neural networks

This thesis considers representing a task from an image using graphs composed of geometric features. Many real-world applications [170], such as social network analysis, traffic prediction and scene understanding, require graph-structured data processing. Graph neural networks prove to be an efficient tool for such roles to encode the whole graph nodes as compact vectors, prediction graph nodes' connections and cluster graphs. Based on whether the graph structure is fixed or not, approaches of graph neural networks can be overall categorized as methods that deal with fixed graphs and methods that deal with dynamic graphs. Here we consider only fixed graph approaches.

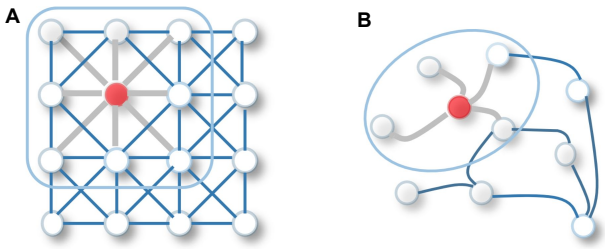


Figure 5.2: Comparison of 2D convolution (A) and a graph node convolution (B) [154]. The 2D convolutional operator takes a weighted average of a fixed-size neighbouring window around the red dotted pixel. The graph convlutional operator also takes the average value of all neighbouring nodes but with variable node numbers and orders.

Alternatively, different approaches of graph neural networks can also be categorized into spectral approaches and spatial approaches. The early proposed graph neural networks are spectral approaches [13] that have a solid foundation in graph signal processing based on convolution operators in the spectral domain. These methods involves intensively computing the eigenvector of the graph Laplacian matrix as the operator filters. On the other hand, the newly proposed spatial-based graph neural networks [44, 102] directly operate convolutions on the graph nodes based on the node's spatial relationships. Fig. 5.2 shows the difference between a traditional 2D convolutional operator and a node operator on the graph. This thesis applies the spatial-based approaches.

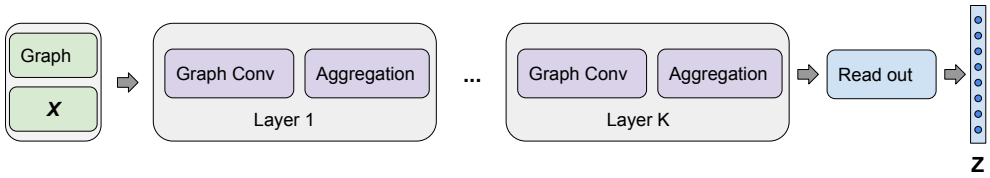


Figure 5.3: A typical framework of the spatial approach based graph neural network.

5.3.1Message passing neural network (MPNN)

As shown in Fig. 5.3, a typical framework of the spatial approaches contains multiple-layer convolution blocks (graph convolution, node aggregation) followed by a readout function that outputs a compact latent vector to represent the graph. On each module in the framework, many different advances such as NN4G [102], DCNN [9], have been made recently, wherein the earliest inspiring approach is the message passing neural network (MPNN) [44]. MPNN treats the graph convolution as a message-passing process that generates messages and propagates messages based on node connections. Fig. 5.4 shows the process of message passing.

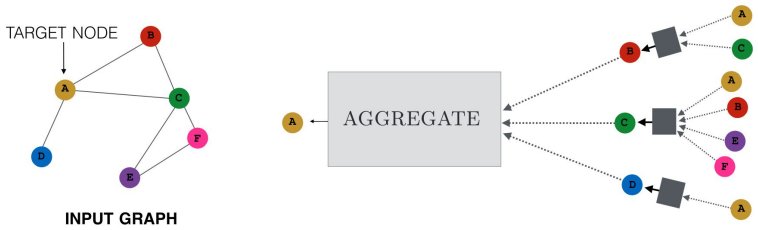


Figure 5.4: Overview of the message pasing machanism [56]. Take the example of node A, it aggregates all incoming messages from its neighbouring nodes. Likewise, all the other nodes take the same aggregation process.

For example, suppose an undirected graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ along with a set of node features $\mathbf{X} \in \mathbb{R}^{d \times |\mathcal{V}|}$, where $\mathcal{V}$ denotes the nodes and $\mathcal{E}$ defines the edges. MPNN assumes each node $u \in \mathcal{V}$ has a hidden embedding state $\mathbf{h}_u^k$ and an accompanying message $\mathbf{m}_u^k$, at the kth layer convolutional block. From layer k to k+1, the update is made in three steps. (1) A pair message generation using a generator $\mathcal{M}$:

$$\mathbf{m}_{u\to v}^{(k)} = \mathcal{M}(\mathbf{h}_u^{(k)}, \mathbf{h}_v^{(k)}), \forall v \in Neighbor(u) \tag{5.1}$$

, which generates pair-wise message between all connected nodes. $Neighbor(u)$ denotes the neighbours of node u.

(2) A message aggregation module:

$$\mathbf{m}_u^{(k)} = AGGREGATE(\mathbf{m}_{u\to v}^{(k)}), \forall v \in Neighbor(u) \tag{5.2}$$

, where in this thesis, we consider a simple message aggregation operator by summing up all neighborhood incoming messages.

$$\mathbf{m}_u^{(k)} = \sum(\mathbf{m}_{u\to v}^{(k)}), \forall v \in Neighbor(u) \tag{5.3}$$

(3) A hidden state update module:

$$\mathbf{h}_u^{(k+1)} = UPDATE(\mathbf{h}_u^{(k)}, \mathbf{m}_u^{(k)}) \tag{5.4}$$

, where a gated recurrent unit (GRU) is used as the update module in our implementation:

After K layers of such convolutional blocks, each node will have a final hidden state denoted as $\mathbf{h}_u^{(K)}, \forall u \in \mathcal{V}$. Then, a readout function parse all nodes' final state to a new embedding $\mathbf{z}$ to represent the graph. In our implementation, the readout funcion is the summation of all $\mathbf{h}_u^{(K)}$.

5.3.2Permutation invariance

One critical issue considering processing graph-structured data is the permutation invariance. Generally, suppose we can define a graph encoder f which takes the nodes of the adjacency matrix of an undirected graph $\mathcal{G}$, which in all we define as graph features $\mathbf{A}$, and output a vector $\mathbf{z}$ representing the graph. Permutation invariance is defined as:

$$f(\pi(\mathbf{A})) = f(\mathbf{A}) \tag{5.5}$$

, where π is a permutation operator which selects the input order of graph nodes. The intuition is simply that given the same graph structure, the output of f should not depend on the arbitrary ordering of the graph nodes.

Now let us consider the case that f is a simple multi-layer perceptron. Obviously, such permutation invariance can not be satisfied. As mentioned above, the graph neural network is supposed to have this permutation invariance property by the design of multi-layer graph convolutional blocks since node orders are not used, and only the adjacent relationships matter the layer update [100].

5.4VGS-IL: visual geometric skill imitation learning

5.4.1Representing geometric constraints using graphs

As discussed in Chapter 2, section 2.1.3, geometric constraints provide a systematic framework for visual task specifcation. In this chapter, we propose to use graphs as structural priors to encode the geometric constraints. More specifically, given a set of geometric features $\mathbf{X} = \{x_i\}$, we define an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a representation of the innerconnections between geometric features, where the nodes $\mathcal{V}$ are variables that take input of feature descriptors $x_i \in \mathbf{X}$ and edges $\mathcal{E}$ define their association relationships. Furthermore, suppose a graph neural network g is used to encode the graph structure $\mathcal{G}$ along with the node feature inputs $\mathbf{X}$ and output a latent embedding $\mathbf{z}$ representing this geometric constraint. We have:

$$\mathbf{z} = g(\mathbf{X}|\mathcal{G}) \tag{5.6}$$

Representing complex geometric features using keypoints: The above equation is the basic idea of geometric constraint graphs. However, one major challenge in practice is how to represent complex geometric features, such as lines, conics, planes, as inputs to the graph neural network. In computer vision, the problem of designing or learning complex geometric feature descriptors per se is intriguing to solve. Nevertheless, for keypoints, there have been many off-the-shelf hand-crafted feature point descriptors (SIFT, SURF, ORB), or deep learning-based descriptors [168]. Can we represent complex

geometric features using a set of keypoints to simplify the problem? This approach aligns with recently proposed methods of learning complex geometric feature descriptors. For convenience, we use $[x_i, ..., x_m]$ to define a complex geometric feature, where x_i are the keypoints used for its representation.

Commonly, any geometric constraint is a binary relationship on two geometric features, such as two points, a point and a line, two lines, a point and a plane. To generally describe complex geometric features using keypoints, as proposed above, a geometric constraint is now extended to a multiple-entity relationship that operates on a set of feature points. Given this defintion, eq. 5.6 is rewritten as below:

$$\mathbf{z} = g(\mathbf{X} = \{[x_1, ..., x_m], [x_{m+1}, ..., x_n]\}|\mathcal{G}) \tag{5.7}$$

, where x_i are keypoints, $[x_1, ..., x_m]$ and $[x_{m+1}, ..., x_n]$ represent the two geometric features, the square brackets define a complex feature composed from a set of keypoints. For simplicity, we denote eq. 5.7 as:

$$\mathbf{z} = g([x_1, ..., x_m], [x_{m+1}, ..., x_n]) \tag{5.8}$$

Since we've introduced an extra composition structure to represent a complex geometric feature, the graph structure $\mathcal{G}$ should satisfy the below two properties. For convience, let us give the line-to-line constriant as an example, its graph representation is $g([x_1, x_2], [x_3, x_4])$ where each line is represented using two points:

- **Permutation-invariant**, which means an arbitrary order of the two complex features and the keypoints inside each feature should contribute to an invariant graph structure, namely:

$$g([x_1, x_2], [x_3, x_4]) = g([x_3, x_4], [x_1, x_2]) \tag{5.9}$$

$$g([x_1, x_2], [x_3, x_4]) = g(\pi([x_1, x_2]), \pi([x_3, x_4])) \tag{5.10}$$

, where π is the permutation operator.

- **Non-inner-associative**, which means changing the inner association rules of the complex features will change the graph structure, namely

$$\begin{aligned} g([x_1, x_2], [x_3, x_4]) &\neq g([x_1, x_4], [x_3, x_2]) \\ g([x_1, x_2], [x_3, x_4]) &\neq g([x_1, x_3], [x_2, x_4]) \end{aligned} \tag{5.11}$$

From section 5.3.2, we know that eq. 5.9 is naturally satisfied by the graph definition and graph neural network. However, to satisfy eq. 5.10 and eq. 5.11, specially designed $\mathcal{G}$ is required for each geometric constraint representation, which is discussed below.

Graph structure of basic geometric constraints: Now we give the graph structure $\mathcal{G}$ of the three basic geomeric constraints as shown in Fig. 5.5: (1) point-to-point; (2) point-to-line; (3) line-to-line.

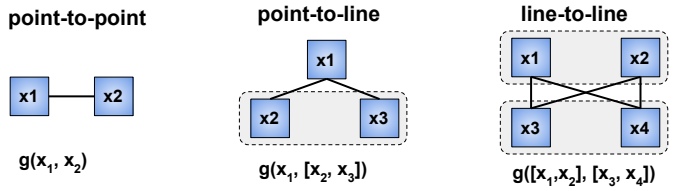


Figure 5.5: The select-out graph structure of three basic geometric constraints. The grey shaded region means the two points represent a line. The selection process is shown in Fig. 5.6

Each graph structure is derived by enumerating all possible node connections and filtering out graphs that do not satisfy the "permutation-invariant" or "non-inner-associative" property. To list all possible node connections inside a graph $\mathcal{G}$, we require any node $v \in \mathcal{G}$ that its degree should satisfy $deg(v) \geq 1$ and the intermediate connection nodes with $deg(v) \geq 2$, since the graph needs organize all the points without any node or edge isolations. For example, consider the line-to-line constraint using four keypoints. There are a total of 38 possible structures of $\mathcal{G}$. After filtering out, the structure shown in Fig. 5.5 is selected out. Fig. 5.6 shows the selection process.

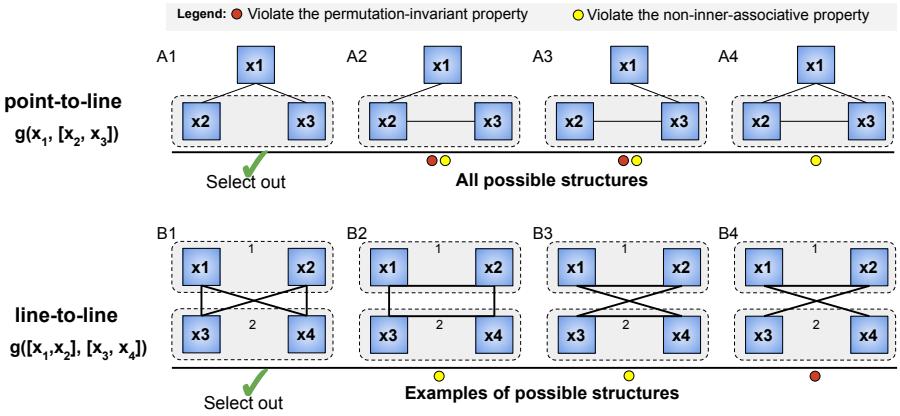


Figure 5.6: Examples of how the graph structure of point-to-line and line-to-line constraints are selected out. We firstly enumerate all possible node connections without any node or edge isolation, wherein the point-to-line has 4 candicates and the line-to-line has 38 candidates. Then we filter out candicates that violate the above mentioned two properties. As shown above, red dot indicates violation of the permutation-invariant property and yellow dot for the non-inner-associative property. To give examples of such violations, in B3, non-inner-associative property is violated since $g([x_1, x_2], [x_3, x_4]) = g([x_1, x_4], [x_3, x_2])$.In B4, permutation-invariant property is violated since $g([x_1, x_2], [x_3, x_4]) \neq g([x_3, x_4], [x_1, x_2])$.

5.4.2Graph-structured visual task encoders

In previous sections, we have covered how a graph neural network encodes node relationships (section 5.3.1) and the graph structure $\mathcal{G}$ of geometric constraints (section 5.4.1). Now the only missing block is the input to a graph neural network—node features $\mathbf{X}$. Suppose an encoder ϕ maps image point features $\{f_i\}$ to latent vectors that are graph nodes, we define:

$$\mathbf{X} = \phi(\{f_i\}) \tag{5.12}$$

In this thesis, we consider two types of node feature, one that uses off-the-shelf feature descriptors (SIFT, ORB) and one that uses a deep neural network to encode image features as the point feature descriptor.

Hand-crafted feature graph: As shown in Fig. 5.7A, an intuitive way to construct graph node features $\mathbf{X}$ from an image is to use hand-crafted features $\{f_i\}$, such as SIFT or ORB. However, it does not learn task-relevant descriptors since they are not end-to-end trainable.

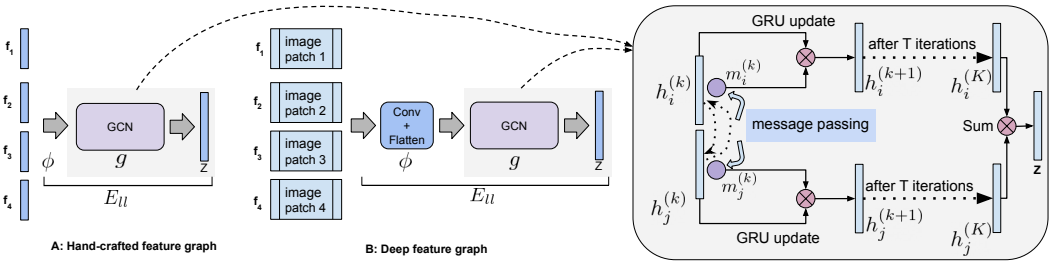


Figure 5.7: Graph structured visual task encoder E_{ll} of a line-to-line geometric constraint. (A): A hand-crafted feature graph that takes off-the-shelf feature descriptors such as ORB or SIFT as node features $\mathbf{X}$ feeding to the graph neural network. (B): A deep feature graph that uses a deep neural network to encode the image features as node features $\mathbf{X}$. A simple example of a two-node graph neural network is given, as shown in the right. More details are depicted in section 5.3.1.

Deep feature graph: Alternatively, we can extract image features $\{f_i\}$ that represent the keypoint and use a convolutional neural network (Fig. 5.7B) to construct graph node features $\mathbf{X}$. Following this approach, it is possible to learn task-relevant feature descriptors and, as will be discussed in Chapter 6, to learn task generalizable feature descriptors.

Generally, given any feature encoder ϕ, combined with the graph neural network g, we construct a graph-structured task encoder E_k, which takes input of point features $\{f_i\}$ and the graph structure $\mathbf{G}$ of a geometric constraint k, outputs a d dimensional latent vector $\mathbf{z} \in \mathbb{R}^d$:

$$\mathbf{z} = E_k(\mathbf{X}|\mathcal{G}_k) = g \circ \phi(\{f_i\}|\mathcal{G}_k) \tag{5.13}$$

5.4.3Task function design

Graph selector: A graph selector U_k is defined as: $b_j = U_k(\mathbf{z}_j)$, which maps a graph neural network's output vector $\mathbf{z}_i$ to a scale value measures the task-relevance factor of this graph instance. U_k selects top k graph instances based the rank of their task-relevance factor.

Task function: Combining all above modules, a task function is defined as:

$$T_k = U_k \circ E_k \tag{5.14}$$

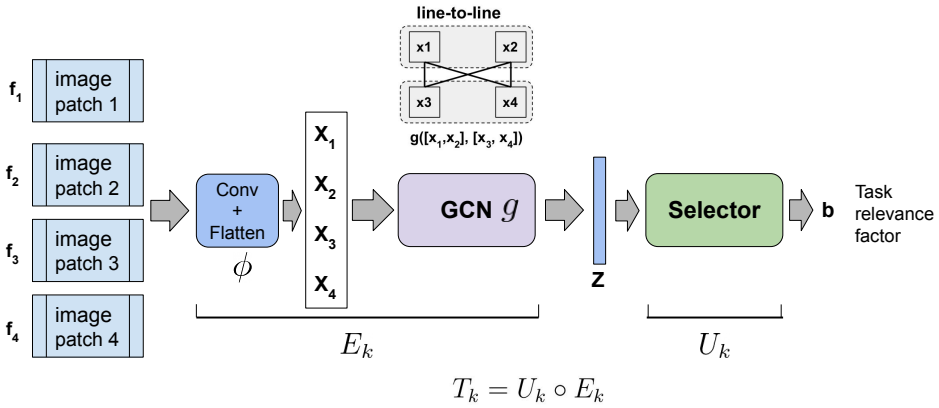


Figure 5.8: Graph structured task function T. Given image features and the graph structure prior that defines a type of goemetric constraint, the task function firstly use *deep feature graph* encoder to encode the interconnections between image features as an embedding of the geometric constraint z. Then pass the embedding to a selector that rank the task-relevance weight of this geometric constraint. Given an image with a pool of feature points, the selector furture selects out the largest weight valued geometric constraint, as shown in Fig. 5.9.

Applying task function on image: Given an image O_t, the process taht task function T extracts task-relevant geometric constraints is as below:

- (1) Keypoint detection: suppose a third party keypoint detector outputs m keypoints.
- (2) Admissible space $\mathcal{F}$ construction: Construct admissible space $\mathcal{F} = \{G_j\}$ by enumerating all possible keypoint combinations to construct graph instances, where G_j denotes a graph instance.
- (3) Compute task-relevance factor value b_j for each graph instance. And select top p graph instances that represent the task.

5.4.4Score function design

Score function based on geometric error output: To recap what has been covered in chapter 4, to use the "temporal-frame-orders" as an unsupervised clue in human demonstration videos, a score function that measures how good or bad an image observation transit from o_t to o_{t+1} is required.

One solution is to measure the geometric error output from the selected geometric constraint ans asumming the norm of error vector is overall decreas-

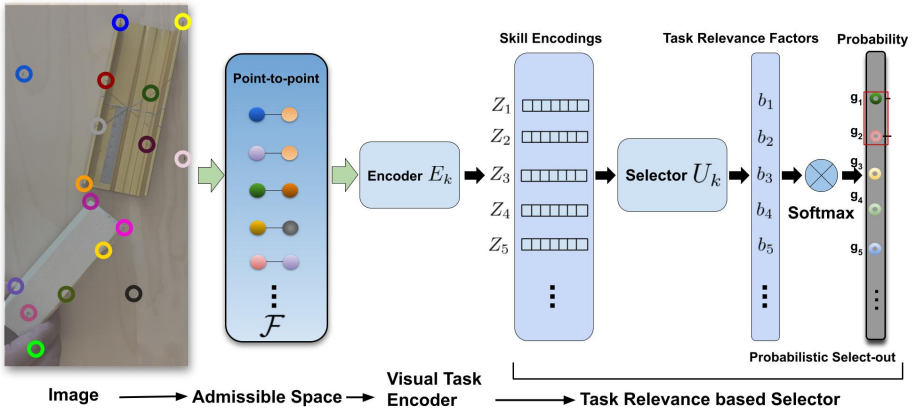


Figure 5.9: Task relevance based graph selector. Given image and constructed admissible space $\mathcal{F}$, the selector maps each graph instance's embedding vector z_j to a task relevance factor b_j. After a Softmax layer, it is further converted to the probability that the corresponding graph instance defines the task.

ing in an expert demonstration. For example, in a point-to-point task, if our task function selects out the correct geometric constraints both on image o_t and o_{t+1}, the observed geometric errors, which are err_t and err_{t+1} respectively, will be overall decreasing. This clue is used to define a score function R as below. Let $\Delta err_t = \|err_t\| - \|err_{t+1}\|$, we define:

$$r_t = \frac{2}{1 + exp(\beta \Delta err_t)} - 1 \tag{5.15}$$

, where $r_t^* \in (-1, 1)$ measures the "temporal-frame-orderness" of observation transtion from o_t to o_{t+1} and, β normalizes the scale of different geometric constraint's output err.

Differential linkage to task function However, the difficulty is that the task function selects out geomeric constraints whose output geometric errors do not directly link to back to the task function itself. To solve this problem, we add a *Softmax* operator on task-relevance factors.

$$g_j = Softmax(b_j, \{b_1, ..., b_j, ...\}) \tag{5.16}$$

$$err_t = \sum g_j \mathcal{E}_j \tag{5.17}$$

, where g_j is the probability that the corresponding graph instance define the task and $\mathcal{E}_j$ is the geometric error of the geometric constraint that this

graph instance defined. Then, each image o_t outputs a weighted geometric error err_t. Now the score function links to the task function, by optimizing the score function, the task function will selectout task-relevant geometric constraints over a pool of all possible candicates.

5.4.5Optimization

Since the task function T and score function R are clearly defined, applying the InMaxEnt-IRL algorithm proposed in chapter 4 will solve the problem.

Cost function: The cost function is as below:

$$\mathcal{L} = \arg\max_{E_k, U_k} \sum r_t^* - log\mathcal{Z}_t \tag{5.18}$$

, where the partition function $\mathcal{Z}_t$ is estimated by a Monte Carlo estimator as stated in chapter 4, section 4.3.4.

RSW regularizer: In practice, using an average weighted geometric error as a guided signal in optimization will cause the selector U_k to predict a narrowly distributed task relevance probability that can not differentiate the top selections from all the candidates. To make U_k selects in a more deterministic manner, we introduce *Residual Sum of Weights* (RSW) penalty to the loss function as $-\lambda RSW$, which makes U_k output major task relevance weights on selected p alternatives while minimizing the residual sum of weights. RSW penalty is appended to the loss function defined in eq. 5.18.

GCR regularizer: To ensures selecting out consisent features between different observations, we introduce a *Geometry Consistent selection Regularizer* (GCR), which is defined as: $-\alpha \left\| b_{t+1} - b_t \right\|_2^2$, where α is a hyper-parameter. A GCR regularizer is appended to the loss function defined in eq. 5.18.

The VGS-IL algorithm details are summarized in algorithm 3.

Algorithm 3: VGS-IL

Input: Expert demonstration video frames $\{o_1, ..., o_n\}$, demonstrator's confidence level α, graph structural priori $\mathcal{G}_k$ of a geometric constraint k.

Result: Optimal weights $\boldsymbol{\theta}^*$ of task function T_k

Construct graph instances on each frame

Define $\mathcal{S} = \{\}$

for t = *1:n* **do**

  Feature extraction on o_t according to k defined in Section 5.4.3

  $s_t \leftarrow$ Construct all graph instances by feature association

  $\mathcal{S} \leftarrow$ Append s_t

end

Prepare State Change Samples $\mathcal{D}s = \{s_t \rightarrow s_{t+1}\}$

$\boldsymbol{\theta}^* = \text{InMaxEnt-IRL}(\mathcal{D}s, \alpha, T_k)$

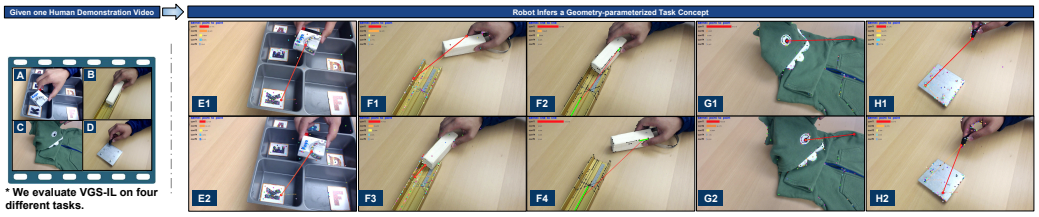


Figure 5.10: Left: Four tasks designed in evaluation: *Sorting*, *Insertion*, *Folding* and *Screw*. **Right**: Qualitative evaluation results of VGS-IL in the four tasks. We select two frames for each task. The *Insertion* task includes two columns representing point-to-point and line-to-line constraint respectively. For a fair test, we changed the background and target pose in each trial. Red line indicates selected feature association with highest confidence.

5.5Evaluation of VGS-IL

Evaluation objectives: We aim to evaluate: (1) what kind of tasks are VGS-IL capable of and what are not; (2) will the learned geometry structured task function transfer from a human demonstrator to robot imitator; (3) will the learned task function generalize to environmental changes; (4) what exactly is InMaxEnt-IRL optimizing when learning the graph-structured task function.

Additionally, we conduct ablation studies to determine the effectiveness of the RSW and GCR regularizer.

Tasks design: Four types of tasks are tested (Fig. 5.10A, B, C, D). **A:** *Sorting* task represents a regular setting; **B:** *Insertion* is for tasks that need a combination of point-to-point and line-to-line constraints; **C:** *Folding* is for

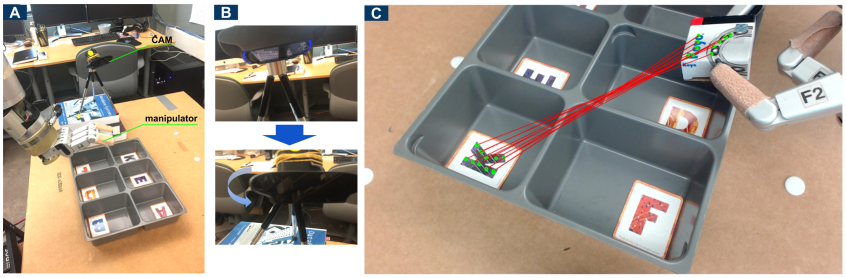


Figure 5.11: **A:** Camera and robot setup. **B:** Changing camera pose in order to evaluate generalization performance under projection variances. **C:** The hand-tracking baseline.

manipulation with deformable objects; **D:** *Screw* task represents types that have low image textures.

Evaluation protocol: Each task is evaluated firstly on videos showing a human holding the object and moving in random actions. Then we evaluate videos of a robot performing the same task with random motions. The objective is to test if the learned task funcion can select correct geometric constraints to define the demonstrated task. Camera and robot setup is shown in Fig. 5.11A.

Each task function is trained by feeding only one human demonstration video and tested if our proposed method will output correct and consistent geometric constraints to define the task. Increasing the number of human demonstration videos will improve the performance, which we leave for future work. Note that this evaluation protocol is challenging as studied intensively in the visual imitation learning literature [40, 156].

Metrics: We design two evaluation metrics: (1) *Acc* to measure accuracy; and (2) *conAcc* to measure consistency. Specifically, given N video frames, $Acc = \frac{M \times 100}{N}\%$, where M is the number of frames with correct geometric task concept inference. Defining *conAcc* is challenging since directly measuring the inference consistency involves complex statistical methods [142]. For simplicity, we measure the time-series control error output $\{err_t\}$ (i.e. the inference outcome) and define $conAcc = Autocorr(\{\|err_t\|\}, k)$, which is the autocorrelation measurement over time-series error norms with *shift*=k. We fix k=*2* in

all experiments.

Baselines: There are no existing methods that learn geometric feature associations from human demonstration videos to our best knowledge. Therefore, for a better analysis of our proposed method, we hand-designed three baselines as below.

(1) A *hand-tracking-baseline* based on a video-tracking module and a redundant feature set, which is often used by conventional visual servoing. As shown in Fig. 5.10 C, it involves humans carefully selecting N pairs of geometric features (point or lines) to represent a task and initializing N feature trackers for each camera. As long as one pair out of N can track throughout the entire task process, we define the baseline succeeds. Moreover, measuring the *conAcc* is hard since it is a collection of redundant pairs of trackers. In this case we assume that *conAcc=1*, the maximum, if baseline succeeds. A KLT point tracker and a moving edge tracker from the ViSP [67] toolkit library are used. As a result, the hand-tracking baseline is a strong one since the above special considerations.

(2) A *ORB-graph* baseline, which relies on ORB descriptors [123].

(3) A *SIFT-graph* baseline, which which relies on SIFT descriptors [96].

Our proposed method *deep-feature graph* uses generalized image features as graph inputs without any hand-engineered feature descriptors.

Training: All task functions have the same graph layer size=5 with hidden state dimension=512. In training, we set the regularizer coefficient $\lambda = 0.1$, and human factor $\sigma_0 = 0.55$. Each encoder and selector with different descriptors or deep features are trained individually on a moderate lab PC with an Intel Core i7-3770 CPU and Nvidia GeForce GTX 1080 Ti GPU.

5.5.1Task capability evaluation

To evaluate the proposed *VGS-IL*'s capability with different tasks, we train the task function using one human demonstration video for each task. Then we conduct evaluations on videos showing a human performing the same task but

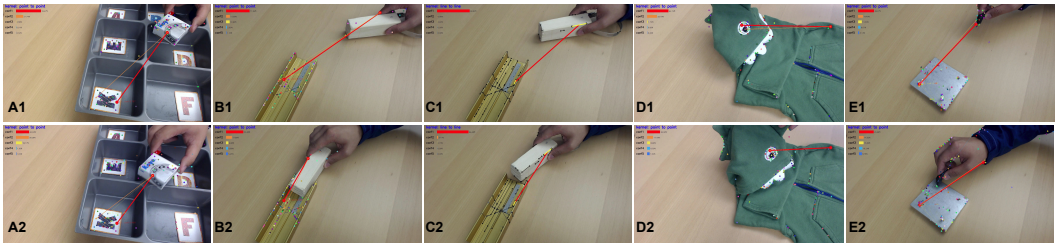


Figure 5.12: Qualitative evaluations of task capability. Each task is shown with 3 frames averagely selected spanning the whole input video sequence. **A1, A2:** a point-to-point *Sorting* task; **B1, B2, C1, C2**: a point-to-point and line-to-line *insertion* task; **D1, D2**: a point-to-point *folding* task; **E1, E2:** a point-to-point *screwing* task. Top five feature connections are selected out. Only the top one, as marked in red color, is used in evaluation to compute *Acc* and *ConAcc*.

with random behaviours. The *Acc* and *ConAcc* metric is used for comparison. Fig. 5.12 displays qualitative evaluations and Table 5.1 shows quantitative results respectively. Experiments show VGS-IL succeeds in learning a consistent geometry-parameterized task concept from a human demonstrator in all the four tasks. Quantitative results are displayed in Table 5.1.

Results: Results (Table 5.1) of the 4 tasks show our method is capable of the *Sorting* and *Insertion* tasks but performs moderately in *folding* and *Screwing* tasks. Overall, a deep feature based visual task encoder is better than SIFT and ORB.

Analysis: In experiments, we observed that when both object and target have rich textures, results are better. One reason is the use of hand-crafted feature detectors (point, line) that the detected feature number will be low when image textures are plain. We find that the more features that can be fed into the task function, the better accuracy it performs. Due to our hardware GPU limitation, we can only test using a small number (60 on average) of features. Due to our lab GPU limitation, we expect to see better results when a deeper network structure is used.

Methods	Hand-tracking		SIFT-graph		ORB-graph		Deep-feature graph	
Tasks	Acc	ConAcc	Acc	ConAcc	Acc	ConAcc	Acc	ConAcc
Sorting	100.0%	1.00	100.0%	0.86	78.9%	0.78	100.0%	0.98
Insertion-pp	100.0%	1.00	100.0%	0.90	68.7%	0.82	100.0%	0.85
Insertion-ll	100.0%	1.00	81.2%	0.42	81.2%	0.31	93.0%	0.91
Folding	10.0%	-0.12	80.0%	0.93	74.2%	0.50	84.0%	0.98
Screwing	8.2%	-0.36	33.0%	0.93	27.0%	0.75	49.0%	0.92

Table 5.1: Evaluation results of different methods. *Deep-feature* graph is our proposed method that uses image features in a graph structured task function.

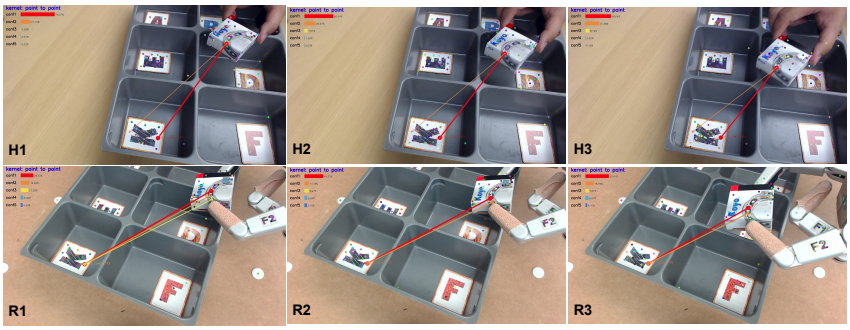


Figure 5.13: Qualitative evaluation of transferring from human demonstrator to a robot imitator under a different camera pose, background and various target poses. Three different frames are displayed. **H1, H2, H3:** three frames from a human evaluation video. **R1, R2, R3:** three frames from a robot evaluation video. Results show the selected geometric constraints stay consistent from a human demonstrator to a robot imitator.

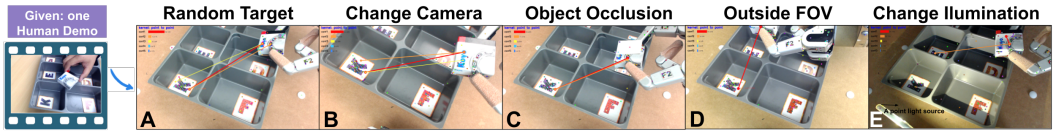


Figure 5.14: Evaluation on different environmental changes.

5.5.2Transfer from human to robot

Next, we test if the learned task function generalizes from a human demonstrator to robot imitator. The task function is trained using our deep-feature graph encoder with a GCR regularizer and evaluated on videos showing a robot performing the same task under random settings. A WAM robot equipped with a Barret Hand is used to test the *Sorting* task (Fig. 5.11A). A qualitative evaluation is shown in Fig. 5.13.

5.5.3Generalization to environmental changes

Furthermore, we keep testing on the robot while exploring more variance settings, as shown in Fig. 5.14: (A) *random target*; (B) *move camera*: We test for real-world projective invariance by randomly translating and rotating the camera; (C) *object occlusion*; (D) *object outside FOV*: The object moves outside the camera's field of view, and each method is required to recover when the object is back in the image automatically; and (E) *change illumination*: the lighting condition is changed by adding a spotlight light source. We pick the task *Sorting* for evaluation. A detailed environmental change evaluation setup can be found in Appendix A. Fig. A.2.

Results: Quantitative results are shown in Table 5.2 and a qualitative study of how the selection behavior differs under various environmental settings are displayed in Appendix A, Fig. A.3.

Analysis: In general, 1) our method exhibits adaptive behaviours that when some geometric features are occluded, the task function selects a candidate geometric constraint to make it up; 2) our method exhibits robust behaviour

Methods	hand-tracking		SIFT-graph		ORB-graph		deep-feature graph	
Settings	Acc	conAcc	Acc	conAcc	Acc	conAcc	Acc	conAcc
A	**100.0%**	**1.00**	99.1%	-0.03	83.5%	0.08	100.0%	0.55
B	0.0%	n/a	96.7%	-0.10	79.3%	-0.10	**95.0%**	**0.61**
C	0.0%	n/a	92.7%	-0.05	89.6%	-0.17	**97.3%**	**0.10**
D	0.0%	n/a	81.2%	-0.03	72.7%	-0.09	**79.8%**	**0.19**
E	0.0%	n/a	0.0%	n/a	0.0%	n/a	**19.2%**	**0.42**

Table 5.2: Evaluation results of transferring the learned encoder, selector from human demonstrator to a robot imitator under various environmental settings. **A**: random target; **B:** change camera; **C:** object occlusion; **D:** object running outside of FOV; **E:** changing illumination conditions.

that the failure between frames doesn't affect successive frames since it directly selects the feature association on each frame. In contrast, the baseline depends on the initialization of trackers and continuous tracking. We observe that the learned model tends to select fixed feature associations while showing the flexibility of selecting alternatives when fixed ones are not observable. It reaches high accuracy under projective variance (**B**), however, fails under illumination changes (**E**).

5.5.4Ablation studies

Evaluation of RSW regularizer: As an ablation study, we continue to investigate how a *Residual Sum of Weights* (RSW) regularizer helps to learn a "good" selector that selects in a deterministic manner so that the differentiation of "correct" and "wrong" feature connections is learned. Fig. 5.15 shows evaluation results.

RSW measures how much relevance contributed from the non-selected connection instances. The lower RSW, the more deterministic in select-out. Fig. 5.15A shows the training curve with RSW regularizer. The relevance from the remaining instances stays below 0.1%. 5.15B is without RSW regularizer. Though the cost function is optimized, the non-selected ones still occupy 75% of relevance. Note that we are maximizing the loss. Therefore, an RSW regularizer effectively enforces the graph-structured task function to differentiate "correct" geometric constraints that define the task from "wrong" constraints as much as possible.

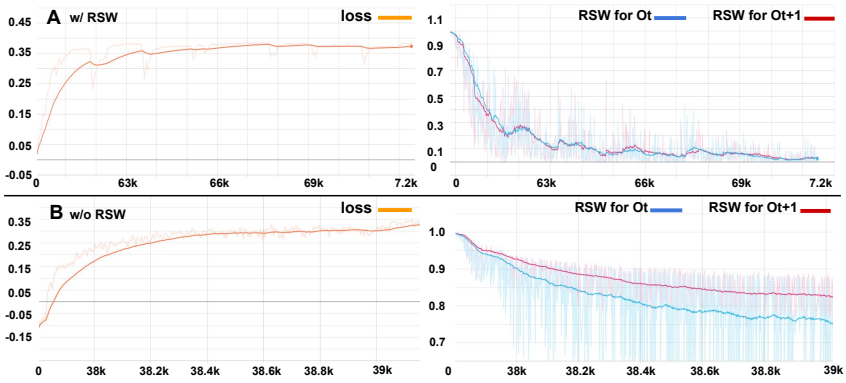


Figure 5.15: Ablation studies the training curve and the sum of residual weights with (**A**) and without **B** a RSW regularizer. A RSW regularizer effectively enforces the task function to differentiate "correct" and "wrong" geometric constraints as much as possible.

Method	w/ GCR regularizer		w/o GCG regularizer	
Metrics	Acc	ConAcc	Acc	ConAcc
Sorting	100.00%	0.98	92.50%	0.12
Insertion-pp	100.00%	0.85	89.30%	0.01
Insertion-ll	93.00%	0.91	51.50%	-0.11
Folding	84.00%	0.98	70.00%	0.03
Screwing	49.00%	0.92	41.20%	0.06

Table 5.3: Ablation studies on GCR regularizer.

Evaluation of GCR regularizer: Recall that, a *Geometry Consistent Regularizer* (GCR) ensures a consistent geometric constraint selection. To test its effectiveness, tboth euantivative and qualitative evaluations are conducted.

Quantitative results: To evaluate how GCR works, we conduct quantitative evaluations on different methods by training the task function with and without GCR. Results (Table. 5.3) clearly shows the task function trained with GCR performs the best selection consistency.

Qualitative results: Furthermore, we test qualitatively how a GCR regularizer affects selection consistency by testing the task function on 6 successive image frames on different tasks. Fig. 5.16) clearly shows GCR improves selection consistency. However, the results also show our proposed method's performance is downgraded in a line-to-line *insertion* task and a point-to-point *screwing task*, which means a generalized image patch based line representation needs further improvement and our method can not work very well in tasks

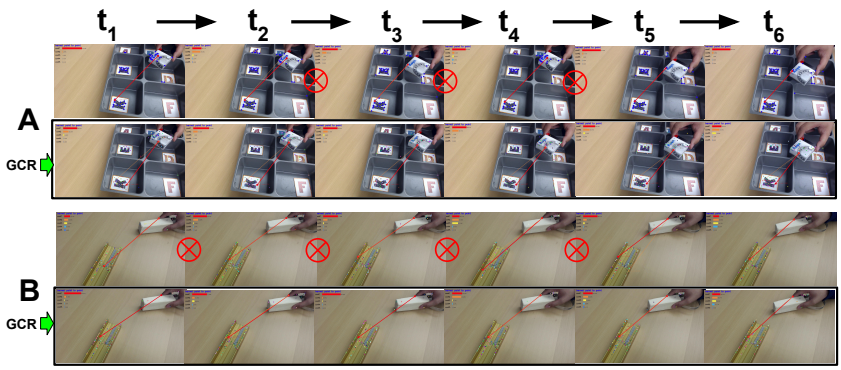


Figure 5.16: Qualitative evaluations of the task functions trained with GCR and without GCR for the the **A:** a point-to-point *Sorting* task; **B:** a point-to-point *insertion* task. 6 image frames for each task are selected. Each of the same image is evaluated twice by a model trained without GCR (top row of each task) and with GCR (down row of each task) respectively. A circled red cross means inconsistent selection detected in the located frame transition. Full results of all the tasks are included in Appendix A Fig. A.4

with low image textures. All the above issues are worth further studying.

GCR regularizer improves smooth geometric error output: Lastly, we visualize the time-series geometric error curve using a model trained with GCR and without GCR, respectively (Fig. 5.17). Note that both two models can output correct geometric constraints that represent the same task for every input frame. However, a GCR regularized model can output more smooth geometric loss signals because it ensures both correct and consistent selection. Considering that the geometric loss output will guide a robot controller, a smooth signal output will be important.

5.5.5Visualizing the learned task function

Visualize the geometric error output: We use the point-to-point *sorting* task to visualize how the geometric error output from the learned task function changes in different training stages. We save the snapshot of the learned task function at three different training stages (Fig. 5.18): S1, S2 and S3 represent the early, middle and final training stages, respectively. Then a robot task video is fed into the three model snapshots and outputs the time-series geometric errors from the selected geometric constraints. To do this, we had the robot perform the *sorting* task via teleoperation, then record video

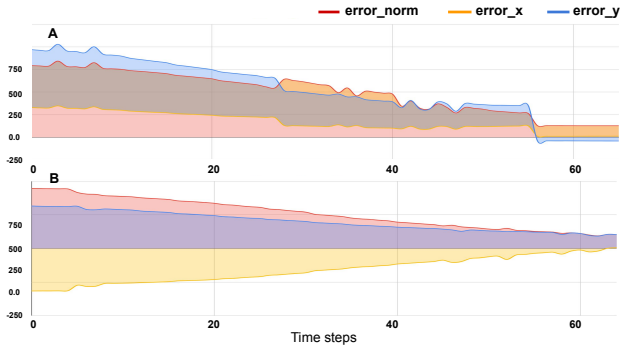


Figure 5.17: GCR w.r.t. smooth geometric loss signal output. We visualize the select-out geometric constraint's loss signal output by feeding the same human demonstration video to two models trained with GCR (B) and without GCR (A) respectively. GCR regularized model can output more smooth geometric loss signals because it ensures both correct and consistent selection.

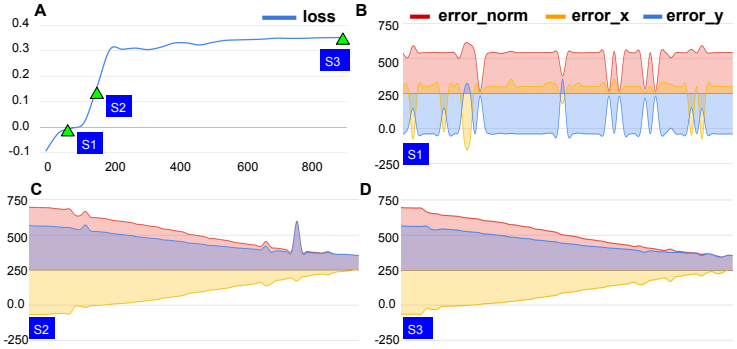


Figure 5.18: Visualization of time-series control signal outputs from different optimization stages. **A**: Training curve of VGS-IL with three different stages picked for evaluation. **B, C, D**: Geometric error signals output from VGS-IL trained in stage *S1, S2, S3*.

frames. Results are shown in Fig. 5.18 that our proposed *VGS-IL* is actually optimizing the task function to produce "good" control error signals. Therefore, the optimization process is optimizing the quality of control error signals.

Visualize the geometric feature selection: Lastly, we are interested in evaluating how geometric constraints are selected out for all the tasks. As shown in Appendix A. Fig. A.5, the output of the graph-structured task function selects correct geometric constraints across different tasks. The learned task function is interpretable by the use of the structural prior $\mathcal{G}$.

5.6Summary

This chapter firstly introduces a graph-structured task function that can be used to encode geometric constraints. Then, we propose VGS-IL, which trains the task function using the "temporal-frame-orders" in human demonstration videos. The learned task function selects correct geometric constraints that specify the task. Experiments are performed to study the capability, generalization of the learned task function. Ablation studies of each module of our proposed method are also conducted. In summary, this chapter makes learning geometry-based task specifications from human demonstration videos possible.

Limitations and future directions:

(1) The generalization performance, as shown above, only concerns environmental changes. Besides environmental changes, task generalization to categorical objects with the same task functionality will be a more interesting direction. We argue that the proposed graph-structured task function should have the generalization potential to categorical objects since it encodes image patch relationships rather than using the whole image pixels to represent a task. This limitation will be studied in Chapter 6.

(2) Though we demonstrated the use of combined geometric constraints, for example combing a point-to-point and line-to-line constraint by learning two task function separately, it is worth further exploring how to sequentially link those constraints under predicates of perceptual conditions and final configurations as discussed in [29]. A hierarchical learning framework will be an interesting direction to explore.

CHAPTER 6

Generalizable task representation learning

TL;DR:

> "Task generalizable representation learning is achieved by extracting the interconnection of image features on different objects that have the same task functionality. It is trained using human demonstration videos showing the same task but using categorical objects."

This chapter considers the problem of generalizable task representation learning. Unlike end-to-end methods that either learn generalizable state representation [86, 169] or generalizable policy [118] via massive robot-environment interactions, our method introduces a structural geometry priori to learn a task function that transfers across categorical objects. The structural priori enables the task function to extract interconnection of image features on categorical objects that have the same task functionality, therefore, forms the representation of the task on each image state. Experimental results show the learned task representation consistently defines the task across the human demonstrator, robot imitator and categorical objects.

6.1Introduction

Task-specification correspondence: Commonly, a task has two parts: a 'what' part that specifies the task and a 'how' part that drives task execution. One attractive characteristic of the 'what' part—the task specification, is its consistency, which means task specification will not change from human demonstrator to robot imitator, and different task settings, such as different tools, objects or backgrounds. As a result, the consistency of task specification builds correspondence between different task settings, which is the key to task generalization.

The *task-specification correspondence* brings in task generalization. For example, in generalizable reinforcement learning, although we have seen different approaches that tackle the generalization problem on either the state level [86, 147], the policy level [118], or the more subtle MDP blocks level [169], the anchor point for why it generalizes is the reward signal which consistently defines the same task and guides the task learning. Therefore, the consistent reward function provides task-specification correspondence across different task domains.

Forming the task-specification correspondence using a reward function could be a scalable and general method. However, the training requires not only tedious reward shaping but also massive robot-environment interactions. Considering the training cost on the real-world robot, as discussed in Chapter 2, learning directly from human demonstration videos is more cost-efficient. So then, in the problem of task generalization, how to build task-specification correspondence using human demonstration videos?

Our approach: We build *task-specification correspondence* between categorical objects using a task function learning approach from a geometric perspective. We approximate a task function that directly encodes task specification from an image observation by extracting task-relevant geometric constraints. As a result, the learned task function is the bridge between different task settings, forms the task specification correspondence, and enables generalizable

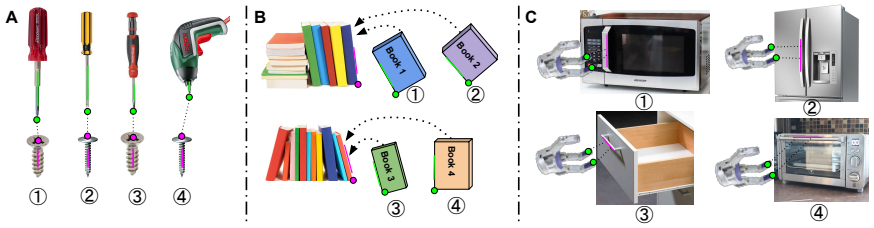


Figure 6.1: Task specification correspondence. The same geometric constraint from categorical objects defines the same task, which is called *task specification correspondence*. The affordance of object parts alone does not define any task. Beyond affordance, the interconnections between multiple object parts provides task specification, thus represents the task and can be used to build *task specification correspondence*.

task representation learning.

6.2Related works

Object affordance and functional parts: The study of how parts of the different objects have the same functionality has decades of research in "affordance". As commonly studied in functional objects [111], object affordance [88, 110], key-point based task representation [41, 42, 116] and scene graphs [51, 76], functional parts of an objects (commonly they are basic geometry primitives) generally applies to different task settings. These parts are world clues that give the sense of how a manipulation task should be described and constructed. For example, the shape of the teapot handle provides functional clues that how a human hand should pick up the mug [106].

In a manipulation task setting, such clues generally apply as well. However, the affordance clue only defines how object parts should be used, but not how a task should be executed. For example (Fig. 6.1 A), although the screw driver's functional part is always its pen tip regardless of what size, colour it is, it does not define any task. Instead, a screwing task is an interconnection between a screwdriver's pen tip to a screw top. More examples are given in Fig. 6.1. As a result, *beyond affordance*, object parts relations bring more insights into generalizable task representation. However, representing a task using the association between object parts is rarely studied in current researches.

Task generalization using geometric features: Recent studies have shown the generalization performance by formatting a manipulation task using keypoint features. Florence et al. 2019 propose to learn invariant keypoint descriptors across categorical objects by human labelled samples. The learned keypoint descriptor is then used to guide the robot motions via a Cartesian controller, thus fulfil task generalization. Qin et al. 2019 propose to unsupervised learn task-relevant keypoints on objects and use the learned representation in a reinforcement learning task for generalizable task learning. Looking at extracting task-relevant keypoints could be a way towards task generalization. However, the ability of how keypoints convey task-relevant information and how the task is represented has not been studied.

In fact, beyond keypoint only, using the association of more general geometric features to specify a task has been systematically studied in visual servoing (chamuteell 1995, Hagger). The generalization potential also has been discussed by Hager et al. 2000 [54], "when describing how a screwdriver is used for a task, the description should apply to all screwdrivers, not a specific instantiation of one.". However, due to the computation resource limitations and the lack of an efficient representation tool, such ideas were not further explored. With the significant advances in computation resources and the representation power of neural networks, *our work* continues the research towards this direction and proposes a method that makes it work.

Relational representation for generalization: *Our work* tackles generalizable task representation learning by looking at relationships between object parts. Likewise, Battaglia et al. 2018 [11] cast relational inductive bias as a key path to combinatorial generalization for building AI (artificial intelligence) with human-like abilities. In their successive work [165], a self-attention-based neural network is introduced to encode relationships between different image features, achieving the best performance in reinforcement learning of StarCraft game. Their self-attention module is a pair-wise similarity measure (scaled-dot-product [144]) between different entities output from an image encoder (e.g., convolutional layers). As a result, it is using pair-wise similarity weights

to encode entity-level relationships. The idea of encoding relationships as the key to generalization does not get sufficient attention in the research community. However, as more progress has been made in relational encoders such as graph neural networks, visual transformers, and capsule networks, we expect to see a lot more works in the near future.

Generalizable visual descriptors: Our work is inspired by the various approaches that learn generalizable visual descriptors for robotic tasks. Commonly, these approaches learn the generalizable visual descriptors based on 3D correspondence, involving 3D reconstruction and then reprojection generated correspondence samples on the images. For example, *Dense Object Descriptor* [42], which combines object segmentation pipelines to ignore learning descriptors for the background. Their further work kPAM [99] uses contrastive learning to explore learning dense object descriptor that is invariant across categorical objects. However, such training requires laborious annotations on each of the images. *Our work* proposes to build the correspondence in an unsupervised way by extracting task-relevant geometric constraints across categorial objects. Apart from the task generalization gain in our approach, compared to these methods that require hand-engineered robot controllers, using geometric constraints is more controller friendly that can be directly applied to a geometric vision controller (visual servoing).

6.3Background: task representation learning

In this thesis, we propose to learn a task function, which encodes the task definition, from human demonstration videos. The idea of using a task function to approximate task specification has a long history of research, which we name as task function-based approaches.

The early works rely on hand designing the task function and are restricted to its task representation capability [124, 125]. Recently, learning a task definition/task function using human demonstration has gained much attention, among which one major challenge is how to parameterize a task function gen-

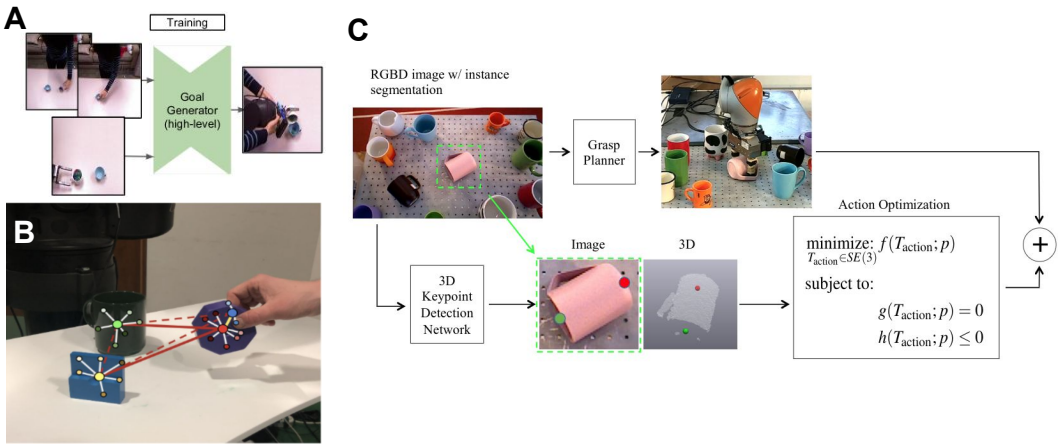


Figure 6.2: Examples of learning task specifications. **A, raw image level** [134] : Robot learns a goal generator parameterized by a neural network to approximate the specification of human demonstrated task. **B, object level** [135]: Robot learns the task specification by infererring relationships between objects using a graph neural network. **C, keypoint level**: kPAM [99] firstly learns an invariant dense object descriptor [42] by 3D reconstruction and then requires user click on a target point. Lastly, a controller is trained using loss defined inside the black box.

erally. Based on the inductive bias introduced, the state-of-art works can be categorized into three types:

(1) Directly using a neural network to approximate the task function [72, 134]. Then the latent output vector represents the task. A typical example is third person visual imitation learning [134] (Fig. 6.2A). The authors prove that decoupling task *'what'* and *'how'* significantly reduces human demonstrations needed in training and improves learning efficiency compared to conventional visual imitation learning approaches. However, the training process of third person visual imitation learning is still tedious.

(2) Introducing the prior knowledge that object-level spatial relationship strongly relates to the task definition. This is intensively studied on object-level imitation learning [135, 143](Fig. 6.2B), object-oriented reinforcement learning [28, 104] and our previous work [72] in learning a controllable object spatial relationship encoding. While it is fair to say object-level spatial relationship-based task encoding can cover many manipulation tasks, it lacks enough granularity to represent tasks like object alignment or more complex tasks. So it can not be used to describe a task generally.

(3) Introducing key points and their relationships. This method has a more substantial task representation power and generalization performance by looking deep into features on objects. For example, in Dense Object Descriptor [41, 42] and their extended work kPAM [99](Fig. 6.2C), the task of hanging a mug on the rack can be precisely described as the relationship between the handle center and rack endpoint. A similar idea has also been tried in KETO [116]. Dense Object Descriptor relies on robots driving the eye-in-hand camera to collect samples. The training has multiple phases. Firstly a dense descriptor, if learned by correspondence pairs generated using 3D reconstruction, then a manipulation task still relies on humans specifying a target point to define the task. So it does not circumvent the hand specification process. At each image frame, it searches over pixels to find the target point by descriptor vector matching. After the eye-hand calibration, the controller commands the robot to that 3D target with a preset orientation. As a result, it is a point-to-point coincidence, a particular case of our geometric task encoding method. Meanwhile, most such approaches have to train a controller tediously.

Our work extends key point-based approaches to a more general geometry-based approach. We give a more thorough discussion on using geometric features like points, lines and conics to represent a task. Moreover, we take extra considerations on studying how to design a proper parameterization of *what* to make designing *how* much easier.

6.4CoVGS-IL: generalization by structural projection

We propose CoVGS-IL, a categorical object generalization version of VGS-IL, which learns generalizable task functions by extracting geometric constraints of categorical objects with the same functionality. CoVGS-IL is learned from multiple human demonstration videos of categorical objects without the need for human annotation or robot-environment interactions.

6.4.1Global structures for task representation

The key part of CoVGS-IL is to project each video frame image on a geometric structural prior $\mathcal{G}$ to get a structured representation of the task. The prior $\mathcal{G}$ plays the role of a global structure for the task representation across categorical objects.

Global semantic structures: In the field of extracting semantic task representations, we have seen multiple successes by projecting human demonstration videos on a grammar-based tree structure for long-term action planning by generating action trees [161] or semantic task plans [92, 160]. The critical insight among these approaches is the existence of a global semantic structure representing the task by parsing human demonstration videos.

Global geometric structures: Apart from the global semantic structures, a global geometric structure can also be used for task representation, observing that various manipulation tasks can be specified by combinations of geometric constraints (see chapter 3 for more details). Moreover, compared to semantic structures, using a geometric structure will output geometric errors that are more controller friendly and directly link to geometric vision-based controllers (visual servoing).

Using a global geometric structure for task representation is motivated by decades of task specification research using geometric constraints. The critical insight is that a geometric constraint is inherently a connection relationship between geometric features. It is such a relationship that defines the task. For example, a point-to-point constraint is the relationship of two-point connections. Similarly, a point-to-line constraint is a relationship between three points if we describe a line using two points as depicted in chapter 5, section 5.4.1.

Such structure is global because the relationship itself stays consistently between arbitrary video frames and across video frames showing categorical objects. Thus, empirically, it is like human experience that the task objective (task specification) exists in our working memory, spans the whole duration

when we do the task, and remains unchanged when trying a different object to do the same task.

The structural prior $\mathcal{G}$: Therefore, we define the global geometric structure as a graph $\mathcal{G}$ that encodes the interconnection relationship between geometric features. Formally, $\mathcal{G}$ is un undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes $\mathcal{V}$ are variables that take input of geometric features $x_i \in \mathbf{X}$ and edges $\mathcal{E}$ define their association relationships. Finding the graph structure of $\mathcal{G}$ for each geometric constraint is defined in chapter 5.4.2.

6.4.2Task representation by global structure projection

Given a global structural prior $\mathcal{G}$, the projection operator on an image observation o_t, as shown in Fig. 6.1, is selecting task-relevant geometric features to fit in the graph structure and to form a geometric constraint that defines the task.

Therefore, projecting an image observation o_t on $\mathcal{G}$ will output a d dimensional latent vector $\mathbf{z}_t$ for relationship encoding and a feature coordinates based geometric error signal $\mathcal{E}_t$ for the controller. Assuming a graph neural network g is used, then:

$$\mathbf{z}_t = g(\mathbf{X}|\mathcal{G}) \tag{6.1}$$

, assuming a function ψ takes input of feature coordinates $\mathbf{y}$ and compute the geometric errors, then:

$$\mathcal{E}_t = \psi(\mathbf{y}|\mathcal{G}) \tag{6.2}$$

, dtails about computing geometric error based on the geometric constraint structure $\mathcal{G}$ can be found in [48, 58].

So the output $\mathbf{z}_t$ is a descriptor that encodes feature connections. $\mathcal{E}_t$ is a task objective error signal that encodes feature coordinates.

6.4.3Building task-specification correspondence

Task-specification correspondence relates task-relevant features from different domains, such as categorical objects, different backgrounds, and task execu-

Algorithm 4: CoVGS-IL

Input: Human demonstration videos $\mathcal{S} = \{\{o_t^i\}, \{o_t^j\}, ...\}$ with object i, j, ...; demonstrator's confidence level α; graph structural priori $\mathcal{G}_k$ of a geometric constraint k.
Result: Optimal weights $\boldsymbol{\theta}^*$ of the task function T_k
Randomly initialize θ of T_k, cloning T_k as T_k^{sim} with weights θ_{sim}
Prepare dataset: define $\mathcal{D}_s = \{\}$, $\mathcal{D}_{img} = \{\}$
for *each video* $\{o_t^i\} \in \mathcal{S}$ **do**
  Prepara state change samples $\mathcal{D}^i$
  $\mathcal{D}_s \leftarrow$ Append $\mathcal{D}^i$
end
for *each image frame* o_t *in all demo videos* $\mathcal{S}$ **do**
  Feacture extraction on o_t according to k defined in Section 5.4.3.
  $s_t \leftarrow$ Construct all graph instances by feature association
  $\mathcal{D}_{img} \leftarrow$ Append s_t
end
Shuffule $\mathcal{D}, \mathcal{D}_{img}$
for *n=1:N* **do**
  Optimize T_k using the "temporal-frame-orders" loss $\mathcal{L}$ for N1 steps:
  $\theta^{n+1} = \text{VGS-IL}(\mathcal{D}, \alpha, T_k, \theta^n)$
  Copy weights θ^{n+1} to θ_{sim}^n.
  Optimize T_k^{sim} using the similarity loss $\mathcal{L}_{sim}$ for N2 steps:
  $\theta_{sim}^{n+1} = \theta_{sim}^n + \nabla_{\theta_{sim}} \mathcal{L}_{sim}(\mathcal{D}_{img})$
  Perform one step momentum update on T_k:
  $\theta^{n+1} = \beta\theta^{n+1} + (1-\beta)\theta_{sim}^{n+1}$, where $\beta \in (0, 1)$ is a momentum coefficient.
end

tors, but with the same task functionality. As a result, task specification correspondence enables generalizable state representation learning.

In this chapter, we propose to use a geometric structural prior $\mathcal{G}$ to extract task-relevant geometric constraints as the task representation. By projecting an image over $\mathcal{G}$, we get the task embedding $\mathbf{z}$ as the task specification.

Task embedding similiarity loss: Therefore, given two different images o^i and o^j showing human doing the same task with two different object i and j respectively, their task embedding $\mathbf{z}^i$ and $\mathbf{z}^j$ build a correspondence link since they should both define the same task, which means by maximizing the similarity between $\mathbf{z}^i$ and $\mathbf{z}^j$:

$$\mathcal{L}_{sim} = \arg\max_{T_k} \frac{1}{N} \sum sim(\mathbf{z}^i, \mathbf{z}^j) \tag{6.3}$$

, where T_k is the task function as defined in chapter 5, section 5.4.3, N is the total number of image pairs from object i and j, $sim(\mathbf{z}^i, \mathbf{z}^j)$ is a similarity metric. By maximizing the loss function, we can enforce the task function T_k learns generalizable task representations for categorical objects.

In the thesis, we use a cosine distance as the similarity metric to normalize the magnitude of task embeddings.

$$sim(\mathbf{z}^i, \mathbf{z}^j) = \frac{\mathbf{z}^i \cdot \mathbf{z}^j}{\|\mathbf{z}^i\| \, \|\mathbf{z}^j\|} \tag{6.4}$$

The consideration is the corresponding geometric constraint on different objects, though defining the same task functionality usually has different textures. As a result, a cosine similarity metric is preferred over a l_2 norm distance metric.

6.4.4Differential linkage to task function

When directly linking the above loss function to the task function will cause difficulties during training. This is because the task function operates on the whole admissible space $\mathcal{F}$, which is a pool of candidate geometric constraints, to select out the one with the highest task-relevance factor value. So the gradient signal from the above loss function will only pass through the select graph instance. To avoid this issue, we add a differential linkage from the loss function in eq.6.3 to the task function T_k, by replacing $\mathbf{z}$ with a weighted vector $\tilde{\mathbf{z}}$ considering all graph instances in the selection pool. $\tilde{\mathbf{z}}$ is defined as below:

$$g_j = Softmax(b_j, \{b_1, ..., b_j, ...\}) \tag{6.5}$$

$$\tilde{\mathbf{z}} = \sum g_j \mathbf{z}_j \tag{6.6}$$

6.4.5Joint optimization

We perform a joint optimization approach to maximize the "temporal-frame-orders" loss (eq. 5.18) proposed in VGS-IL and the task embedding similarity loss (eq. 6.3). Each loss function used to optimize the task functions for several

steps on the whole dataset until both converge. To ensure training stability, we momentum update is performed in the switched opimitzation phase.

The CoVGS-IL algorithm details are summarized in algorithm 4.

6.5Evaluation of CoVGS-IL

CoVGS-IL, Categorical object/tool generalizable visual geometric skill imitation learning.

Evaluation objectives: We aim to evaluate: (1) will the learned task function extract correct and consistent geometric constraints that specify the task from human evaluation video frames? (2) will the learned task function transfer from a human demonstrator to robot imitator; (2) will the learned task function generalize to categorical objects; (3) what exactly is CoVGS-IL learning when optimizing the graph-structured task function.

Tasks: We use the hammering task, as shown in Fig. 6.3, which means hammering a nail that requires a robot holding a hammer with random grasping frame displacement and approach the hammer to the nail with the correct orientation. The hammering task does not consider the striking motion in the final steps since (1) a striking motion involves force/torque control beyond our study scope; (2) a striking motion can be made when visually an initial alignment between the hammer and nail is satisfied.

Therefore, a hammering task can be expressed as a point-to-point constraint (a point on hammer bottom to a point on nail top) and line-to-line parallelism (an edge on the hammer part to the body of the nail). Obviously, there are multiple points or lines that can define such constraints, especially for the features on the hammer. Therefore, the task function needs to learn how to select them considering categorical object generalization.

The learned task function is required to extract correct point-to-point and line-to-line constraints that represent the task.

Figure 6.3: Task design for categorical object generalization evaluation. **A**: Robot-camera-demonstrator setup. **B**: Human demonstrates the hammering task from the camera's view. **C**: Robot imitates the hammering task from the camera's view.

Baselines: To our best knowledge, no existing works are learning generalizable task representations that explicitly extract a task's geometric definition from one image. Therefore, to better analyze our proposed method, we design a tracking baseline that serves as a strong baseline.

The *hand-tracking* baseline needs humans to manually select multiple points and lines to define the task for each hammer. Therefore, as described in Chapter 5, Section 5.5.1, the hand-tracking baseline is a strong one since it involves humans manually defining each hammer's task.

The closest work is k-PAM, which builds keypoint correspondence on categorical objects by 3D reconstruction and human-annotated samples. k-PAM can only represent keypoint reaching tasks like reaching the point for grasping, hanging the mug on the rack by keypoint alignment. Compared to k-PAM, our work (1) does not require human annotation or 3D reconstruction to extract keypoint correspondence; (2) is more general for geometry-based task specification. Since the above limitation, k-PAM is not suitable for our evaluation purpose.

Table. 6.1 shows the comparison between CoVGS-IL, the hand-tracking baseline, and k-PAM.

Training: All tasks are trained using the same neural network architecture for the task function. In addition, the task function is jointly optimized by the task embedding similarity loss and the "temporal-frame-orders" loss with both RSW and GCR regularizers, as introduced in Chapter 5.

Methods	Feature selection	Correspondence matching	Task generalization
Hand-tracking	hand selected	by continuous tracking per frame	No
kPAM [99]	hand selected	by 3D reconstruction + human annotation	Yes
coVGS-IL (ours)	learnable	learned by time step projection on G	Yes

Table 6.1: Comparison between CoVGS-IL (ours) and baselines.

Methods	Hand-tracking		Ours	
Task	Acc	ConACC	Acc	ConAcc
Hammering-pp	100.0% ± 0.0%	1.00 ± 0.00	100.0% ± 0.0%	0.94 ± 0.05
Hammering-ll	100.0% ± 0.0%	1.00 ± 0.00	95.4% ± 3.1%	0.89 ± 0.05

Table 6.2: Evaluation results of the hammering task. *Hammering-pp:* defines using a point-to-point constraint in the hammering task. *Hammering-ll* defines using a line-to-line parallelism constraint in the hammering task. In practice, the above two constraints are combined for one hammering task. For analysis, we seperately evaluate them. Results show the learned task function matches the performance with baseline which involes human hand selecting 10 pairs of features and running multiple trackers.

6.5.1Human video evaluation

We train the task function using ten videos showing humans demonstrating the use of the hammer (Fig. 6.5A). Then we evaluate the task function on five different videos showing humans holding the hammer in the task scene and moving in arbitrary motions.

For each video, we use the "correctness" metric *Acc* and the "consistency" metric *ConAcc* as defined in Chapter 5. After running on the 5 different videos, we report average *Acc* and *ConAcc* with their corresponding standard variance. Quantitative results are shown in Table.6.2.

Results show that the learned task function successfully extracts task-relevant geometric constraints of the hammering task.

6.5.2Transfer from human to robot

We continue to test if the learned task function transfers from human to robot. Using the same task function trained from 10 human videos, we evaluate its performance on five robot videos showing the robot holding the hammer in the

Methods	Hand-tracking		Ours	
Task	Acc	ConACC	Acc	ConAcc
Hammering-pp	100.0% ± 0.0%	1.00 ± 0.00	95.9% ± 6.3%	0.96 ± 0.06
Hammering-ll	97.4% ± 6.0%	0.97 ± 0.06	96.8% ± 6.1%	0.90 ± 0.09

Table 6.3: Evaluation results on task function transferring from human to robot. Results show the learned task function matches the performance of hand selection baseline. **Note that** the baseline does not transfer from human to robot since it requires hand clicking 10 feature pairs on the robot's view image again otherwise a directly transfer will fail. Instead, our method is trained using only human demonstraion video and tested on robot's view images.

task scene and moving in arbitrary motions. This protocol is shown in Fig. 6.4. Average *Acc* and *ConAcc* with their corresponding standard variance are reported.

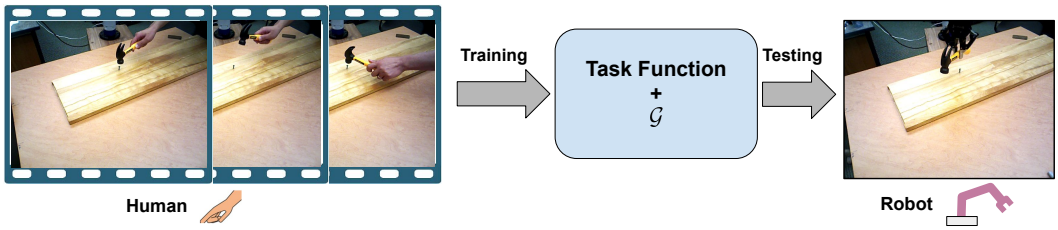


Figure 6.4: Evaluation of human to robot transfer. The graph-structured task function is trained using human demonstration videos and evaluated on a robot imitator's task video showing the robot holding the hammer in the task scene and moving in arbitrary motions. The goal is to evaluate if the learned task function extracts correct geomeric constraints on a robot's view images.

Evaluation results (Table. 6.3) shows the task task function, when applied on the robot imitator, can successfully extract task-relevant geometric features. The same successful transfer has been observed in Chapter 5.

6.5.3Categorical object generalization

To evaluate categorical object generalization, we use four different hammers to test generalization regarding interpolation and extrapolation.

Interpolation: As shown in Fig. 6.5, three types of hammers (A, B, C) are used in training by humans demonstrating how to use them, and a single task function is required to extract task-relevant geometric constraints for the three hammers when used by the robot imitator.

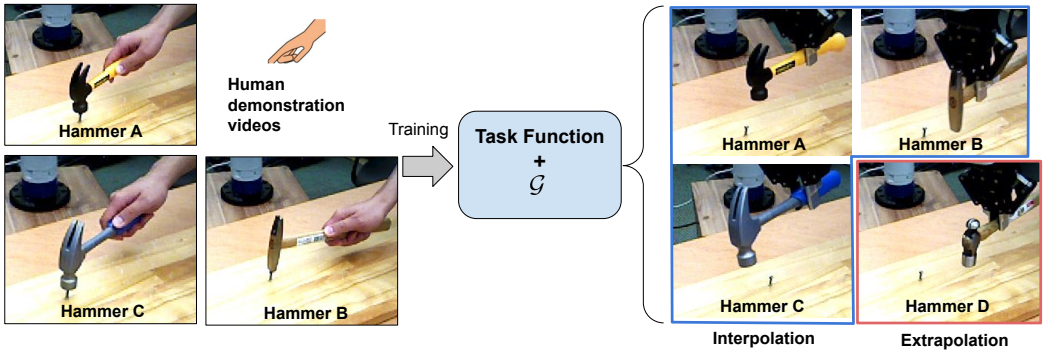


Figure 6.5: Evaluating categorical object generalization regarding interpolation and extrapolation.

Methods		Hand-tracking		Ours	
Tasks		Acc	ConACC	Acc	ConAcc
Hammer A	PP	100.0% ± 0.0%	1.00 ± 0.00	100.0% ± 0.0%	0.91 ± 0.05
	LL	100.0% ± 0.0%	1.00 ± 0.00	95.4% ± 3.1%	0.89 ± 0.09
Hammer B	PP	89.4% ± 8.7%	0.89 ± 0.07	88.4% ± 7.1%	0.80 ± 0.11
	LL	81.4% ± 17.7%	0.49 ± 0.59	86.2% ± 7.2%	0.60 ± 0.28
Hammer C	PP	100.0% ± 0.0%	1.00 ± 0.00	97.2% ± 3.5%	0.60 ± 0.28
	LL	96.0% ± 4.1%	0.88 ± 0.13	98.6% ± 2.3%	0.89 ± 0.08

Table 6.4: Categorical object generalization regarding interpolation. Results show the learned task function's performance moderately matches the *hand-trakcing* baseline. The *hand-trakcing* baseline requires human mannually select 10 pairs of geometric features for each constraint for each of the hammer tpye, and then requires multiple trackers initiated. This gives us further clues to combine both our learned task function and short-term trackers, which will be studied in Chapter 7.

This protocol evaluates the interpolation performance since the testing samples are different from the training samples that (1) their robot poses during the task are different; (2) their robot end-effector to hammer tool frame displacements are different.

We train a single task function by feeding human demonstration videos using hammers A, B and C. Each hammer has ten videos starting from random poses. Then we evaluate videos showing the robot holding hammers A, B and C in the task scene with random motions. Each hammer is evaluated on five different videos. Average *Acc* and *ConAcc* with their corresponding standard variance are reported.Results are shown in Table 6.4.

Methods		Hand-tracking		Ours	
Tasks		Acc	ConACC	Acc	ConAcc
Hammer D	PP	88.3% ± 8.9%	0.71 ± 0.15	91.1% ± 4.7%	0.87 ± 0.08
	LL	84.0% ± 15.1%	0.61 ± 0.42	86.7% ± 7.0%	0.74 ± 0.21

Table 6.5: Categorical object generalization regarding extrapolation. Results show the learned task function's performance matches the *hand-tracking* baseline. Again, note that the *hand tracking* baseline does not generalize to hammer D since it requires human select a pool of features and keeps tracking them.

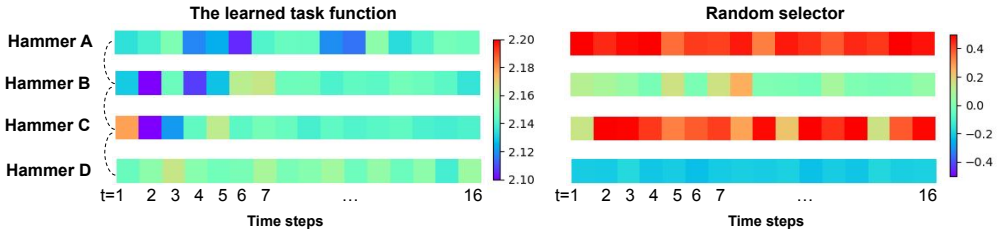


Figure 6.6: Task-specification correspondence visualization of the hammering-PP task which involes two points coincidence to define the task. **Left:** the learned task function's output z_t. For convenience, we visualize the first component of z_t in 16 successive time steps. Results show that the embedding z_t for hammer A, B, C and D stays similar to each other while allowing slight value changes. This is done by selecting image features on objects and construct a graph to represent their geometric constraints. z_t is the representation of the constructed graph. **Right:** a random selector's output z_t. Results show a random selector, though selects image features and constructs the same graph structure; the embedding z_t of the four hammers do not match with each other. A complete visualziaton of all components of z_t can be found in Fig. A.6.

Extrapolation: We train the same task function using human demonstration videos with hammer A, B, and C, then tests its performance on a robot imitator using hammer D. Five different videos showing the robot holding hammer D in the task scene with random movements are used to report the average *Acc* and *ConAcc* with their corresponding standard variance. Results are shown in Table 6.5.

This protocol evaluates extrapolation performance since hammer D is different from the training samples using hammer A, B and C. We test if the learned single task function can extract task-relevant geometric constraints on hammer D considering its texturing similarity to hammer B and shape similarity to hammer A and C.

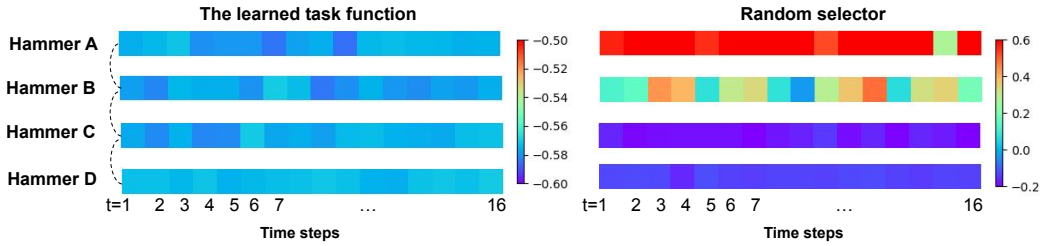


Figure 6.7: Task-specification correspondence visualization of the hammering-LL task which involes two line parallelism to define the task. Results show our task function's outputs that are the line-to-line constraint's representation z_t on categorical objects staying similar to each other while allowing individual varying factor changes. z_t is visualized using 16 successive frames' output and select the first component for visualization convenience. A complete visualziaton of all components of z_t can be found in Fig. A.7.

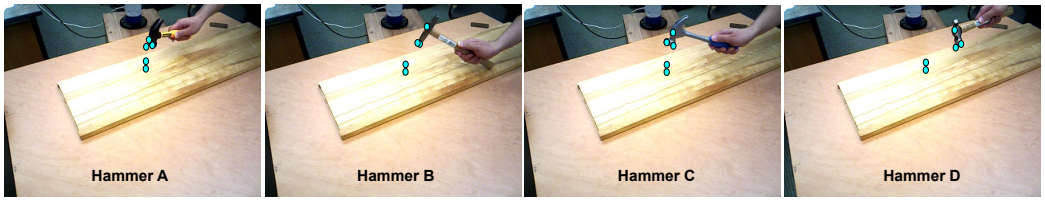


Figure 6.8: Visualization of the select-out goemetric constraints in the four types of hammers. From multiple possible selection candicates that all "correctly" define the task, the learned task function selects out image features that share common charactersistics among categorical objects.

6.5.4Visualization and analysis

Visualize the task-specification correspondence: Furthermore, we test if the task-specification correspondence is successfully built by the global structure projection. Specifically, we visual the graph-structured task function's output z_t given input image observation o_t from categorical hammers A, B, C, and D as shown in Fig. 6.5.

Results are displayed in Fig. 6.6 and Fig. 6.7, which show the global consistency of a task descriptor z_t from categorical objects across multiple time steps is learned in the task function, thus builds the task-specification correspondence and achieves categorical object generalization.

A complete visualziaton of all components of z_t along 16 time steps of the hammering-PP task can be found in Appendix A, Fig. A.6 and the hammering-LL task visualized in Appendix A, Fig. A.7

We also visualize the learned task function's geometric constraint selection of the four hammers, as shown in Fig. 6.8. Results indicate the task function can select both correct geometric constraints that represent the task and image features are share more common characteristics, like image clues that represent the shape, in order to make the output z_t generalize to categorical objects.

The geometric error output for control: At last, we visualize the geometric error output for control, again on hammers A, B, C and D by feed-in each with a human demonstration video. Results are shown in Fig. 6.9.

Though the global task descriptor z_t stays almost unchanged between time steps, Fig. 6.9 indicates the geometric error output which is computed based on features' local coordinates, changes consistently along with actions, which provides a good signal for control. This will be further studied in Chapter 7.

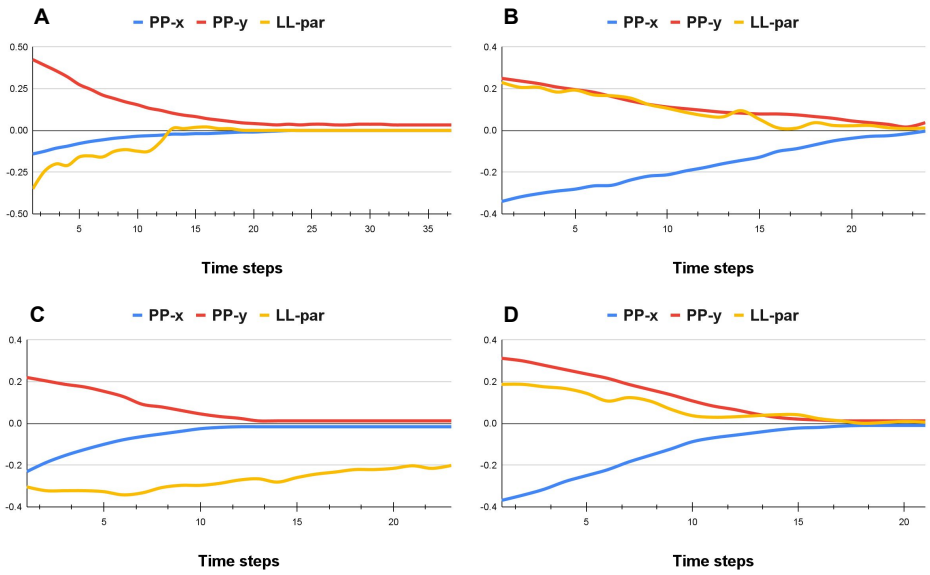


Figure 6.9: Image error outputs of the task function using local coordinates of geometric features and their constraint definition. Each curve is generated by running the task function on a human demonstration video. Therefore, the image error changes indicate how human demonstrated the task. **A, B, C, D** represents human demonstration of hammer A, B, C and D respectively. *PP-x, PP-y*: x, y errors of the point-to-ponit constraint. *LL-par*: parallel line error computed by the dot product of two lines. All image errors are calculated using normalized coordinates. Details are covered in Chapter 7, Section 7.4.2.

6.6Summary

This chapter presents CoVGS-IL, which learns generalizable task representation of an image state by building *task-specification correspondence* between categorical objects. Specifically, we introduce a graph-structured task function to extract task-relevant geometric constraints on different objects but with the same task functionality. Experimental results show the effectiveness of CoVGS-IL in extracting generalizable task representations.

Limitations and future work: Although the learned task function can consistently select task-relevant geometric features, such consistency can not be always guaranteed along with the whole task execution when deploying a learned task function on the real robot. Because the GCR regularizer only enforces consistent task-relevance factors between two frames but does not directly enforce the selected geometric constraints of the two frames are the same. Additionally, in real-world practice, some slight illumination changes may impact the accuracy of the learned task function as well. This limitation relates to the robustness of the task function and will result in large turbulent error signal outputs when plugins a controller. This consistency issue could be solved by combining short-term trackers, which will make the learned task function much easier to be deployed on the robot.

Part II

Geometric Task Controller and Systems

CHAPTER 7

The geometric task learning system

TL;DR:

> "A geometric task learning system includes modules proposed in previous chapters and combined as a solution that takes the input of human demonstration videos and enables the robot to acquire the demonstrated skills."

7.1Introduction

In this chapter, we assembly the previous modules and propose a practical geometric task learning system. Following the discussion on cost-effective real-world robot learning, we continue the "what" and "how" decoupling approach that separates the learning of task specification and the design of a task controller. Specifically, a geometric structured task representation enables the design of a visual servoing controller with a few robot-environment interactions.

This chapter describes the whole process, including the hardware/software architecture, dataset preparation, training and deployment on a real-world

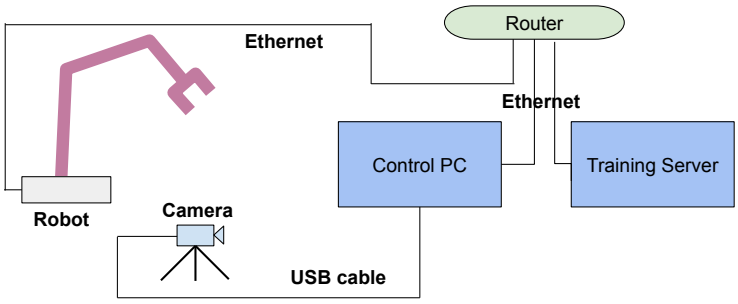


Figure 7.1: System's hardware architecture.

#	Device name	Model No.	Specifications
1	Manipulator	Konova Gen 3 Ultra lightweight	7 Dof, Control frequency: 1kHz, joint position/velocity control force/torque control
2	Camera	Asus Xtion Pro	VGA (640 X 480), 30 fps
3	Router	Belkin AC 1800	100Mb ethernet connection
4	Control PC	Assemblyl machine	CPU: Intel Core i7-4500 RAM: 32GB
5	Training server	Assemblyl machine	GPU: NVIDIA GeForce GTX 1080 Ti

Table 7.1: Device list.

robot. During deployment, the robot feeds the observed image into the learned task function (as described in Chapter 5 and 6) and then uses the select-out geometric constraints to guide robot actions.

At last, we use the hammering task as an example, testing skill transfer from human to robot and task generation to different hammers. A Kinova Gen 3 7-DOF manipulator is used in our experiments.

7.2System design

7.2.1Overview

Firstly, we give the system overview as follows.

Hardware architecture: Fig. 7.1 shows the system's hardware architecture, consisting of a robot, camera, control PC, and a training server. The robot, control PC and the training server are connected via ethernet cables to a 100 Mbps router. The camera uses a USB cable to connect to the control PC. Table 7.1 shows the list of each hardware device's specifications.

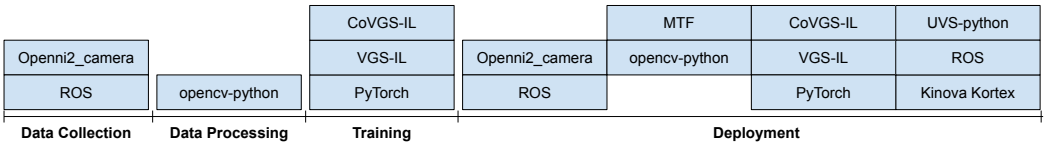


Figure 7.2: System's software architecture.

#	Software name	Source	License
1	ROS	Open Robotics Inc.	Free BSD
2	Openni2_camera	Julius Kammer [77]	Free BSD
3	opencv-python	Alexander Smorkalov [4]	MIT
4	PyTorch	FAIR [34]	Free BSD
5	Kinova Kortex	Kinova Robotics Inc.	Proprietary
6	MTF	Abhineet Singh [137]	Free BSD
7	VGS-IL	Ours	Free BSD
8	CoVGS-IL	Ours	Free BSD
9	UVS-python	Ours	Free BSD

Table 7.2: Software list.

Software architecture: Our system is built on several open-source software/libraries. Arranging by the system's workflow order, the software stack architecture of the system is depicted in Fig. 7.2. Top blocks are built upon the bottom blocks. A full list of software packages and libraries are summarized in Table 7.2.

7.2.2Interfaces for human demonstration

The dataset preparation has the following steps:

Camera calibration: Before data collection, we need to calibrate the camera's intrinsic parameters. The calibration process uses a 9×8 chessboard and runs the ROS "camera_calibration" node. After calibration, the intrinsic parameters are saved as a .yaml file and configured in the "openni2_launch" package's camera launch file.

Human demonstration video recording: In the video recording process, we first start the camera video stream by running the "openni2" library's launch file. During recording, a human demonstrator must slowly show the task on

an average task duration of 13.4s. Moreover, it is recommended that humans demonstrate the task in a more deterministic way to manifest a strong "temporal-frame-orders" assumption used in unsupervised training. In the following experiments, we use the expert's confidence level $\alpha = 0.9$. Fig. 7.5 shows the setup of human demonstration video recording.

Data processing: As shown in Section 5.4.3, the VGS-IL and CoVGS-IL algorithms rely on standalone feature point detectors. We use the ORB feature detector provided by the opencv-python library. Each image is pre-processed by running the ORB feature detector and save the keypoint coordinates, keypoint image features as a dictionary, which is further stored as a pickle file. The dictionary structure is defined as:

" {'image id': {'points': [[x1, y1], [x2, y2],...], 'features': [P1, P2, ...]}}
"

, where x_i, y_i are noramlized values in the range of [-1, 1] computed from image pixel coordinates u_i, v_i and width w, height h as below:

$$\begin{aligned} x_i &= u_i * 2/w - 1 \\ y_i &= v_i * 2/h - 1 \end{aligned} \tag{7.1}$$

$P_i \in R^{c*w_0*h_0}$ is a fixed length $(w_0 \times h_0)$ image patch extracted from the original image given a centroid coordinate (x_i, y_i). In the task we use $w_0 = h_0 = 20$ in pixel unit from an image of 640×480 pixel resolution.

7.2.3Model training

Our experiments test two algorithms to learn the task function: (1) VGS-IL for an essential task function learning that extracts task-relevant geometric constraints. (2) CoVGS-IL for a generalizable task function learning that extracts task-relevant geometric constraints on categorical objects. The training is done on an NVIDIA GeForce GTX 1080 Ti GPU server. The average training time of VGS-IL is around 4 hrs and the CoVGS-IL 6 hrs.

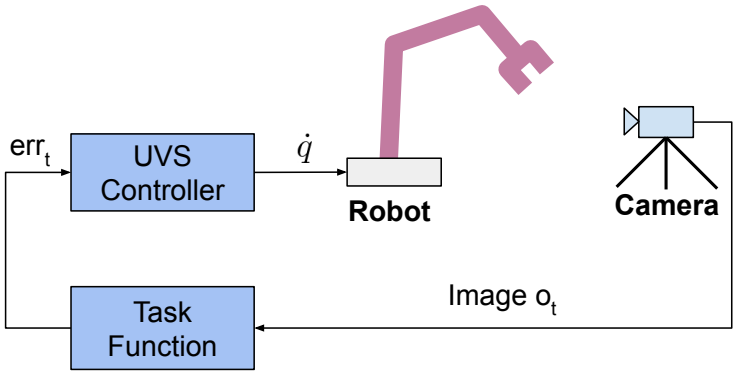


Figure 7.3: Diagram of the feed back control loop.

7.2.4Deployment

Deploy the trained task function model: After training, the learned task function is then deployed on the control PC. Finally, we transfer the weights saved on the training sever to the control PC. During testing, a task function model instance loads the pre-trained weights, takes in an image observation and outputs the select-out geometric constraint.

The feedback control loop design: The task function is used in a feedback control loop as described in Fig. First, the camera's video stream is fed into the task function, mapped to the selected out geometric constraints that define the task. Successively, geometric errors are computed from the selected geometric constraint and used for the uncalibrated visual servoing (UVS) controller [119].

Details of how geometric errors are computed from basic geometric constraints are as follows.

(1) For the point-to-point constraint:

$$\mathcal{E} = (\mathbf{y}_2 - \mathbf{y}_1) \tag{7.2}$$

, where $\mathbf{y}_1, \mathbf{y}_2$ are two points normalized coordinates, as defined in Section 7.3.

(2) For the point-to-line constraint:

$$\mathcal{E} = (\mathbf{y}_1 \cdot (\mathbf{y}_2 \times \mathbf{y}_3)) \tag{7.3}$$

, where $\mathbf{y}_1$ are the coordinates of the point, and $\mathbf{y}_2, \mathbf{y}_3$ are the coordinates of

the two point that represent the line. All coordinates here are used in the their homogenous form.

(3) For the parallel line-to-line constraint:

$$\mathcal{E} = ((\mathbf{y}_1 \times \mathbf{y}_2) \times (\mathbf{y}_3 \times \mathbf{y}_4)) \tag{7.4}$$

, where $\mathbf{y}_1, \mathbf{y}_2$ are the coordinates of the two point representing the first line and $\mathbf{y}_3, \mathbf{y}_4$ are the coordinates of the two point representing the other line. All coordinates here are used in the their homogenous form. Note that, the line coordinates computed from $(\mathbf{y}_1 \times \mathbf{y}_2)$, and $(\mathbf{y}_3 \times \mathbf{y}_4)$ are in their homogeneous form as $(a_1, b_1, c_1)^\top$ and $(a_2, b_2, c_2)^\top$. Before further computing their cross product, they need to be normalized with the last item dropped since parallel line constraint only considers the slope changes. For example $(a_1, b_1, c_1)^\top$ is normalized with last element dropped as $\frac{1}{\sqrt{(a^2+b^2)}}(a_1, b_1)^\top$.

UVS controller The uncalibrated visual servoing controller (UVS) is used to map geometric errors to robot actions. When camera model and the robot model are unknown, uncalibrated visual servoing uses online trial-and-errors to directly estimates the Jacobian $\tilde{\mathbf{J}} \in \mathbb{R}^{d \times n}$ [68, 119] which maps image observations to robot actions, where d is the visual feature dimension, and n is the number of joints used for control. As a result, given the output geometric error signal $\mathcal{E}_t$ at time t, the control law is formulated as:

$$\dot{\mathbf{q}} = -\lambda \tilde{\mathbf{J}}_t^+ \mathcal{E}_t \tag{7.5}$$

, where $\tilde{\mathbf{J}}_t^+$ is the pseudo-inverse of $\tilde{\mathbf{J}}_t$.

A UVS controller starts with an initial estimation of $\tilde{\mathbf{J}}_0$ by exploratory motions, which measure how robot action affects the numerical value changes of observed features. The initial Jacobian estimation via trial-and-errors is based on the following equation:

$$\tilde{\mathbf{J}}_0 = \left[\left[\frac{\Delta \mathbf{e}_{q_1}}{\Delta q_1} \right] ... \left[\frac{\Delta \mathbf{e}_{q_m}}{\Delta q_m} \right] \right] \tag{7.6}$$

, which means the Δq_i amount of joint i movement results in geometric error change $\Delta \mathbf{e}_{q_i}$. After initial estimation, this Jacobian is then continuously updating online via Broyden update during iterations:

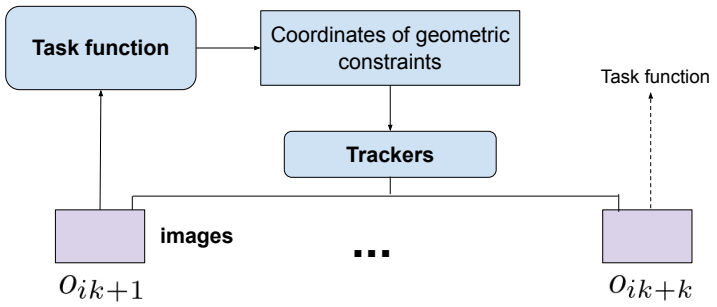


Figure 7.4: Diagram of the long-short term trackers design.

$$\tilde{\mathbf{J}}_{t+1} = \tilde{\mathbf{J}}_t + \alpha \frac{(\Delta \mathbf{e} - \hat{J}_t \Delta q)\Delta q^T}{\Delta q^T \Delta q} \tag{7.7}$$

, where α is the update step size. Except for the Broyden update method, more improvements can be found in [132] and [35].

Long-short-term tracker To ensure a higher success rate, we introduce long-short term trackers to the system. Although the learned task function can consistently select out geometric constraints on each image frame when trained with the GCR regularizer (see Chapter 5, Section 5.4.5), in practice, this is not rigorously guaranteed due to insufficient training samples. As we know, there is always a trade-off between the training cost and the performance gain in deploying a real-world learning system. To further improve the robustness, we utilize the off-the-shelf trackers, which work short-term while cooperating with the task function in the long term. Such design considers that trackers typically work well in the short term, and the task function can play the role of a tracker initializer in the long term when tracking is lost.

As shown in Fig. 7.4, the long-short-term tracker is designed as following. Given a pre-trained task function T, at the first time step, we initialize tracker M using the task function's output geometric constraint's coordinates and keeps tracking for the following $k-1$ steps. Then the tracker will be calibrated by the task function using its output geometric constraint's coordinates.

Figure 7.5: Experimetal setup of the hammering task.

7.3Experiments

Evaluation objectives: In the experiments, we continue using the hammering task as described in Chapter 6, Section 6.5, as an example to test if the proposed geometric task learning system can be deployed on a real-world robot. Specifically, we aim to evaluate the following aspects. (1) Skill transfer from human to the robot: will the system enable a acquire the demonstrated task given human demonstration videos; (2) Generalizable task learning: will the system enable generalizable task learning given human demonstration videos using categorical objects.

We evaluate using the task functions trained in Chapter 6 in an uncalibrated visual servoing (UVS) controller that maps the learned task function to robot actions. Fig. 7.5 shows the experimental setup.

Evaluation metric: Each task runs ten trials, and we report the success rate. For each trial, the robot holds the hammer with a random displacement between the end-effector frame to the object frame and starts the trial at a random pose.

Baselines: For a single task evaluation — evaluating skill transfer from human to robot, we design three baselines. Since no existing works directly learn from human demonstration videos and map to robot actions without pre-defined controller or extra training, we consider designing our baselines by removing the following limitations in our problem setting. (1) We lift the data source limitation so that any data source, including robot-environment interactions samples, is qualified. (2) We standardize the evaluation protocol as stated above. During testing trials, the robot starts at a random initial pose and holds the hammer with a random displacement.

Following the above considerations, we design three baselines as below:

- **Trajectory replay**: We record the robot joint trajectory given one initial pose of the robot and demonstrate the task by kinesthetic teaching.

- **RL (SAC)**: We train a soft-actor-critic (SAC) agent for the task. The state is the image observation o_t, and the actions are robot's 7 joint velocities $\dot{q}$. Since we do not have instrumentation that could determine the ground truth pose of the hammer, we use a reward classifier, similar to [33], as the reward function. The reward classifier collects failure and success samples and outputs the probability of an image in the successful task status. In the experiments, we did not successfully train the RL agent since the state-action space is large. Using a reward classifier introduces prediction errors that need special treatments that impede our further exploration. The training curve is included in Appendix A. Fig. A.8.

- **Behavior cloning**: We collect robot image and joint value pairs (o_t, q) using human kinesthetic teaching. The dataset for each object has 10 human demonstration videos via kinesthetic teaching, which are around 3400 image action pairs per object. We consider resizing the input image to $216 * 144$ dimension. Then, we supervised training a four-layer convolutional neural network that regresses the function mapping from image to robot joint values and uses the trained model on the robot.

Methods	Data	Training
Trajectory replay	Robot joints trajectory	n/a
RL (SAC)	Robot image states, actions	Actor-critic
Bahevior Cloning	Robot image states, actions	Supervised
Ours	Human image states	Self-supervised

Table 7.3: Comparison of baselines.

Methods	skill transfer	Categorical object generalization			
		object 1	object 2	object 3	object 4
Trajectory replay	10%	10%	10%	10%	10%
RL (SAC)	F	n/a	n/a	n/a	n/a
Bahevior Cloning	60%	20%	20%	F	F
Ours	100%	100%	90%	100%	70%

Table 7.4: Overview of results in the hammering task considering skill transferring and task generalization. **"F"** marks failure.

A comparison of all methods is given in Table 7.3.

For the task generalization evaluation — evaluating generalizable task learning, we use the *Behavior cloning* and *trajectory replay* baseline for comparison. The Behavior cloning baselines trains with image action pairs (o_t, q) covering hammers A, B, and C. Ideally, we expect the baseline learn a generalizable regressor that maps categorical object image states to robot actions. The other two baselines are not considered for the generalization evaluation since generalizable reinforcement learning is still challenging. No existing work can be trained on the real-world robot to learn a generalizable policy that maps image states to robot actions.

7.3.1Results and analysis

An overview of all the results is shown in Table 7.4. From the results, we have the following observations.

(1) The *trajectory replay* baseline success only at the first trial when the initial robot pose exactly matches the human teaching pose.

(2) RL (SAC) agent training failed since the high dimensional state-action space of this problem. Fine-tuning the convolutional neural network or using a ResNet [133] backbone could help, but we did not try due to resource limits.

Therefore, we terminate the training after 10K steps, which is about 6 hrs in wall time. The training curve is included in Appendix A. Fig. A.8.

(3) The *Behaviour cloning* performed moderately that the trained image to joint regressor's accuracy matters the most in the final performance. Moreover, the *Behaviour cloning* baseline has worse performance in the generalization test when using a convolutional neural network directly maps categorical object images to robot actions.

7.3.2System user interface

To better visualize the output of the learned task function and facilitate experiments in the robot control phase, we design the following user interface. As shown in Fig. 7.6, four windows are displayed.

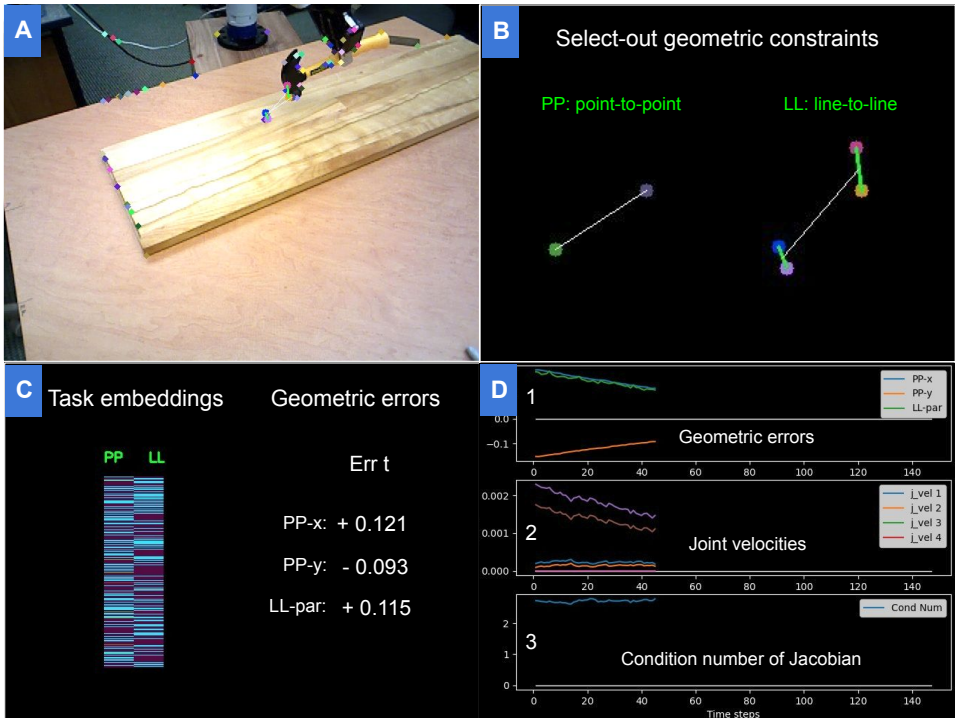


Figure 7.6: System's user interface. The top left (A) shows all candidate geometric features. The top right (B) shows the select-out top 1 geometric constraint by the task function along with the long-short-term trackers. The bottom left (C) shows the output task embeddings and geometric errors, where *PP* means point-to-point and *LL* means line-to-line (parallel lines). Bottom right visualize the (1) geometric error output, (2) command joint velocities and the (3) estimated Jacobian's condition number.

7.3.3Skill transfer from human to robot

We evaluate the system by feeding ten human demonstration videos and test if the robot can acquire the task using an uncalibrated visual servoing controller. Quantitative results are shown in Table 7.4. Fig. 7.7 plots the image error, robot joint velocity and the Jacobian matrix's condition number during the task execution.

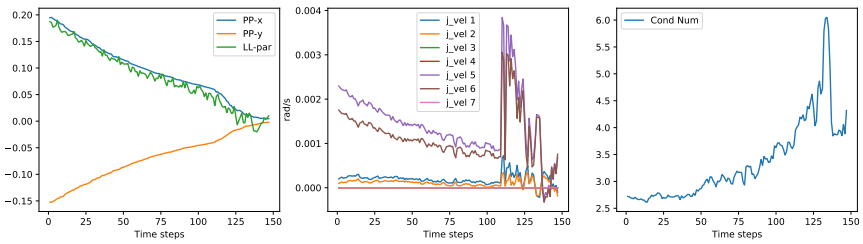


Figure 7.7: Control curves of the hammering task. We use a total number of 4 joints in the UVS controller. **Left**: image errors per time step when plugin the task function with the UVS controller. **Middle:** joint velocity command from the UVS controller. The large joint velocity changes mostly happen when the estimation Jacobian matrix's condition number jumps high. This phenomenon also gives us the clue for future study of robust UVS. **Right:** the Jacobian matrix's condition number per time step. During the experiment, the condition number of the estimated Jacobian remains relatively low, which benefits from consistent geometric constraint selection and our normalized image errors described in Section 7.2.2.

7.3.4Skill transfer of categorical object generalization

We consider using hammers A, B, C, D (Fig. 6.5. In evaluation, we feed in 30 human demonstration videos of hammer A, B, and C (10 for each), and test if the robot can acquire a generalizable skill — uses a single task function to generalize to A, B, C, and D. A detailed analysis of the generalizable task function can be found in Chapter 6, Section 6.5. The success rate is reported in Table 7.4. Details about image error, joint velocities and condition number of the Jacobian matrix during the task execution using all the four hammers can be found in Appendix A. Fig. A.9.

Methods	Success rate	Percentage of successfully tracked frames
W/ LSTT	100%	100.0% ± 0.0%
W/O LSTT	20%	90.9% ± 4.8%

Table 7.5: Ablation study on the imparct of using long-short-term tracker (LSTT) in the hammering task.

7.3.5Ablation studies

Long-short term trackers: Additionally, we conduct ablation studies to assess the impact of our proposed long-short-term trackers by removing the trackers using only the learned task function. As shown in Table 7.5, using the proposed long-short-term trackers significantly improves the final performance.

Number of human demo videos

Lastly, we test how the number of human demonstration videos affects the success rate of task execution. We consider using 1, 5, 10, and 20 demonstration videos to access the success rate. Fig. 7.8 shows the results. Therefore, increasing the number of human demonstration videos will increase the success rate since more samples will cover more image state space.

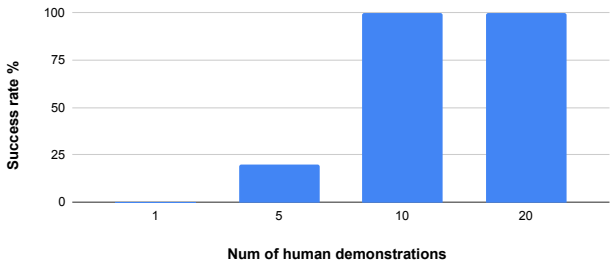


Figure 7.8: Ablation studies on different number of human demonstrations used in training the task function. Results are evaluated using hammer A. Success rate is based on 5 runs of each learned task function.

7.4Summary

This chapter introduces the geometric task learning system that assembly each module presented in Chapters 4, 5, and 6. We describe the system from

hardware architecture, software architecture, training and deployment, respectively. To make the learned task function practically applied in a real-world robot controller, we introduce long-short-term trackers, which in the long-term initialize the tracker by the learned task function and utilize existing off-the-shelf trackers in the short-term improve its geometric constraint selection consistency. Experiments are done using a Kinova Gen3 7-DOF robot.

CHAPTER 8

Conclusions

8.1Summary

This thesis mainly focuses on how to learn geometry-based task specifications from human demonstration videos. The learning paradigm we introduced is sample efficient in that it only requires a few real-world robot-environment interactions when mapping the learned task specifications to robot actions. Furthermore, this learning paradigm represents a task's specification from a geometric perspective, which enables task generalization.

Specifically, chapter 4 empirically studied the task specification capability using geometric constraints. Chapters 3 and 5 introduced a graph-structured task function and use *temporal-frame-orders* as an unsupervised learning clue in human demonstration videos to extract task-relevant geometric constraints. Chapter 6 introduced how such graph structure in the task function enables generalizable task representation learning that extracts geometric constraints with the same task functionality from different objects. Finally, chapter 7 introduced how to deploy a geometric task learning system in the real world.

8.2Discussion

8.2.1The decoupled what and how learning paradigm

One characteristic of this thesis is the choice of a decoupled what and how learning paradigm, wherein only human demonstration videos are given, which means there are no robot view images or robot actions, and we require the robot to learn the demonstrated task. This problem setting is challenging since no robot actions are given, there are no clues about mapping the video samples to robot actions. Furthermore, there are very few learning clues in the video samples that can be used to decide what geometric features are helpful and how should they be associated. So why do we choose this approach?

Firstly, it involves almost zero cost of real-world robot interaction. Only a few samples are needed when mapping the learned task function to a robot controller, for example, when in the initial Jacobian estimation of a UVS controller. Moreover, since the task function outputs compact geometric errors, designing a UVS controller or learning a policy online will be efficient. [1]

Second, it enables model transfer from human to a robot by learning only the representation of the task since the action mechanisms of humans and robots are fundamentally different. Compared to learning a direct mapping from observation to actions, it decouples out learning what the task is—the task specification to learn a shared geometric task representation between human and robot.

8.2.2Our proposed geometric approach

Our proposed approach uses a graph-structure task function to extract task-relevant geometric constraints. It has the following advantages.

Firstly, the graph structure $\mathcal{G}$ enables the task function to extract only task-relevant image feature associations which defines a task and does not relate to whether the task executor is a human desmontrator or robot imitator.

[1] Though we did not explore using an RL controller with the task function's output geometric errors, since its compact output, we can expect training an RL controller will be more efficient than directly using image states.

Meanwhile, introducing a graph structure to the task function also brings in generalization since task-relevant geometric constraints are extracted from different object but with the same task functionality. Therefore, *task-specification correspondence*(Chapter 7, Section 7.1) is built between different objects.

Moreover, introducing a graph structure to the task function also brings advantages in the controller design or policy learning since a graph instance corresponds to a geometric constraint that will output geometric errors, as compact and task-relevant signals for control.

Limitation: It is worth noting that introducing such a strong inductive bias also has the below limitations. Firstly, not all tasks can be specified using geometric constraints. Second, even if a task can be defined using geometric constraints, it is challenging to determine if we should combine them or link them, or add more triggering conditions [29].

Lastly, apart from geometric visual clues, factors regarding force feedback and a tool's physical properties are also crucial to the success of robotic task learning. For example, in the hammering task shown in Chapter 6 and Chapter 7, we only consider geometric feature alignments of the task by assuming the striking motions can be easily pre-defined. However, when the hammer's mass is large, the final strking motion to nail down will be difficult to execute if we simply follow a pre-defined motion pattern. A method combining both geometric visual clues and force feedback is required.

8.3Open research questions

8.3.1End-to-end learning geometric constraints

The proposed method still relies on hand-crafted feature detectors, which outputs feature points based on saliency instead of task relevance. This general feature detector will produce a large number of candidate geometric features, resulting in quadratic complexity when computing the task-relevance factor for each geometric feature combination since the constructed combinatorial admissible space $\mathcal{F}$ is large. To reduce computation complexity, if we simply

restrict the number of key points detected, then we can not guarantee that the learned task function is optimal since the input features are filtered out. One promising direction to go is to train the feature detector and the task function so that the detector will only output a fixed number of task-relevant features. For example, using an FCN [95] neural network architecture, or region proposal network (RPN [45]), or dense visual descriptors [128] could be promising to explore.

8.3.2Task-camera-robot factorization

In a robot learning problem, systematically, we have three factors — task, camera, and robot. "Task" determines the specifications. "Camera" determines how features are observed —the distribution of observations. "Robot" determines how observation maps to an action. In visual servoing control, these factors are explicitly factorized [22]. For example, a typical visual servoing control law is as below:

$$\dot{s} = -L_s \, {}^{c}V_F \, J_e^r \, \dot{q} \tag{8.1}$$

This equation provides explicit modelling of the task-camera-robot factorization where $\dot{s}$ represents the task, L_s represents the Jacobian (interaction matrix [20]) that maps observed feature motion to camera motion . ${}^{c}V_F$ represents camera model, and J_e^r represents the robot's proprioceptive model — the kinematics model. Such factorization enables fast reconfiguration of a task. Examples are as below.

- To construct a new task, we only need to stack new task features into $\dot{s}$ and compute the corresponding interaction matrix L_s.
- To adapt to a new camera configuration, we only need to get the calibration parameters and replace the ${}^{c}V_F$ with the new one.
- To adapt to a new robot, we only need to get a new Jacobian to replace J_e^r, which means J_e^r represents the kinematics model of the robot itself, and can be easily defined.

In contrast to the visual servoing methods [20, 21], a typical robot learning method directly maps observation to robot action so that it does not explicitly differentiate the three factors. In some cases, methods that addressing task generalization only implicitly differentiate the three factors. For example, when tackling task generalization [86, 118, 169] regarding task factor changes, the underlying assumption is the other two factors remaining unchanged.

Factorziation enables model-reuse and fast adaptation: The problem of the approaches mentioned above is that there is no way to reuse or fast adapt models to tackle each factor. We may develop a task generalization model, a robotic platform generalization model, and a camera-configuration generalization model respectively. However, how to flexibly reuse these models when we deploy them remains challenging. For example, the camera may be relocated, the tool frame may be changed, or we may need to fast adapt to a new robot.

If the above factorziation can be built and an explicit replationship between the models tackling each factor is known, we can easily reconfigure any robotic task by reusing models designed for each factor, thus, achieving a higher generalization level.

In robot learning, how should we formulate such factorization? Factor "task" concerns what is a standardized form to represent a task. Factor "camera" concerns how to model the mapping from observed state changes to a robot's workspace by the camera model. Lastly, factor "robot" concerns how to represent a robot's proprioception — its ability to move, that can be fast adapted to any specific tasks. We look forward to seeing future works done in this direction.

8.3.3Robotic task specification research

This thesis views task specification learning from a geometric perspective, extracting the geometric meaning of a task. More fundamentally, our method is based on the understanding of robotic task specifications —"what is a task" and "how should it be defined".

As we know, the problem of how to systematically define manipulation tasks remains challenging that even the basics regarding building a proper taxonomy are difficult. New robotic task specification solutions that remove the requirements of robotic experts, generally apply to a wide range of manipulation tasks, and enable the robot's semi-autonomy to compose the task details given a task context automatically, are still demanding.

Therefore, we believe the research on general task specification is worth exploring further since only if we understand "what is a task", building a general-purpose robotic system will become possible.

References

[1]ABB Robotics Inc. *ABB Robot Studio*. [Online; accessed 5-Sept-2019]. 2021. URL: https://new.abb.com/products/robotics/robotstudio.

[2]Gerald J Agin. *Servoing with visual feedback*. SRI International, 1977.

[3]Seyed Reza Ahmadzadeh et al. "Learning symbolic representations of actions from human demonstrations." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 3801–3808.

[4]Alexander Smorkalov. *OpenCV Python*. [Online; accessed 5-Sept-2019]. 2019. URL: https://pypi.org/project/opencv-python/.

[5]Peter Anderson et al. "Spice: Semantic propositional image caption evaluation." In: *European Conference on Computer Vision*. Springer. 2016, pp. 382–398.

[6]"Apprenticeship learning via inverse reinforcement learning." In: *Twenty-first international conference on Machine learning - ICML '04* (2004), p. 1.

[7]Brenna D Argall et al. "A survey of robot learning from demonstration." In: *Robotics and autonomous systems* 57.5 (2009), pp. 469–483.

[8]Brenna D. Argall et al. "A survey of robot learning from demonstration." In: *Robotics and Autonomous Systems* 57.5 (2009), pp. 469–483. ISSN: 09218890. DOI: 10.1016/j.robot.2008.10.024. arXiv: arXiv:1105.1186v1.

[9]James Atwood and Don Towsley. "Diffusion-convolutional neural networks." In: *Advances in neural information processing systems*. 2016, pp. 1993–2001.

[10]Quentin Bateux et al. "Training Deep Neural Networks for Visual Servoing." In: *ICRA 2018-IEEE International Conference on Robotics and Automation*. 2018, pp. 1–8.

[11]Peter W Battaglia et al. "Relational inductive biases, deep learning, and graph networks." In: *arXiv preprint arXiv:1806.01261* (2018).

[12]Aude Billard and Roland Siegwart. "Robot learning from demonstration." In: *Robotics and Autonomous Systems* 2.47 (2004), pp. 65–67.

[13]Joan Bruna et al. "Spectral networks and locally connected networks on graphs." In: *arXiv preprint arXiv:1312.6203* (2013).

[14]Ian M Bullock, Thomas Feix, and Aaron M Dollar. "The Yale human grasping dataset: Grasp, object, and task data in household and machine shop environments." In: *The International Journal of Robotics Research* 34.3 (2015), pp. 251–255.

[15]Christopher J Burke et al. "Neural mechanisms of observational learning." In: *Proceedings of the National Academy of Sciences* 107.32 (2010), pp. 14431–14436.

[16]Sylvain Calinon, Florent Guenter, and Aude Billard. "On learning, representing, and generalizing a task in a humanoid robot." In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37.2 (2007), pp. 286–298.

[17]Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer science & business media, 2013.

[18]Kaidi Cao et al. "Few-shot video classification via temporal alignment." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10618–10627.

[19]Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset." In: *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6299–6308.

[20]François Chaumette. "Visual servoing using image features defined upon geometrical primitives." In: *Proceedings of 1994 33rd IEEE Conference on Decision and Control*. Vol. 4. IEEE. 1994, pp. 3782–3787.

[21]François Chaumette. "Geometric and photometric vision-based robot control: modeling approach." In: 2018.

[22]François Chaumette and Seth Hutchinson. "Visual servo control. I. Basic approaches." In: *IEEE Robotics & Automation Magazine* 13.4 (2006), pp. 82–90.

[23]Vincent S Chen et al. "Scene graph prediction with limited labels." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 2580–2590.

[24]Christine Connolly. "Technology and applications of ABB RobotStudio." In: *Industrial Robot: An International Journal* (2009).

[25]M. Dehghan et al. "Online Object and Task Learning via Human Robot Interaction." In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 2132–2138.

[26]Masood Dehghan et al. "Online Object and Task learning via Human Robot Interaction." In: *arXiv preprint arXiv:1809.08722* (2018).

[27]Rüdiger Dillmann. "Teaching and learning of robot tasks via observation of human performance." In: *Robotics and Autonomous Systems* 47.2-3 (2004), pp. 109–116. ISSN: 09218890. DOI: 10.1016/j.robot.2004.03.005.

[28]Carlos Diuk, Andre Cohen, and Michael L Littman. "An object-oriented representation for efficient reinforcement learning." In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 240–247.

[29]Z Dodds, M Jägersand, and G Hager. "A Hierarchical Architecture for Vision-Based Robotic Manipulation Tasks." In: *First Int. Conf. on Computer Vision Systems* 542 (1999), pp. 312–330.

[30]Z Dodds, A S Morse, and New Haven. "Task Specification and Monitoring for Uncali brat ed Hand / E ye Coordination *." In: May (1999).

[31]Zachary Dodds et al. "Task specification and monitoring for uncalibrated hand/eye coordination." In: *Proceedings 1999 IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE. 1999, pp. 1607–1613.

[32]Apoorva Dornadula et al. "Learning Predicates as Functions to Enable Few-shot Scene Graph Prediction." In: *arXiv* (2019), arXiv–1906.

[33]Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. "Challenges of real-world reinforcement learning." In: *arXiv preprint arXiv:1904.12901* (2019).

[34]FAIR. *PyTorch*. [Online; accessed 5-Sept-2019]. 2019. URL: https://pytorch.org/.

[35]Amir massoud Farahmand, Azad Shademan, and Martin Jagersand. "Global visual-motor estimation for uncalibrated visual servoing." In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2007, pp. 1969–1974.

[36]Alireza Fathi, Yin Li, and James M Rehg. "Learning to recognize daily actions using gaze." In: *European Conference on Computer Vision*. Springer. 2012, pp. 314–327.

[37]Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization." In: *International conference on machine learning*. 2016, pp. 49–58.

[38]Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided cost learning: Deep inverse optimal control via policy optimization." In: *International Conference on Machine Learning*. 2016, pp. 49–58.

[39]Chelsea Finn et al. "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models." In: *arXiv preprint arXiv:1611.03852* (2016).

[40]Chelsea Finn et al. "One-shot visual imitation learning via meta-learning." In: *arXiv preprint arXiv:1709.04905* (2017).

[41]Peter Florence, Lucas Manuelli, and Russ Tedrake. "Self-Supervised Correspondence in Visuomotor Policy Learning." In: *IEEE Robotics and Automation Letters* (2019).

[42]Peter R Florence, Lucas Manuelli, and Russ Tedrake. "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation." In: *arXiv preprint arXiv:1806.08756* (2018).

[43]Justin Fu, Katie Luo, and Sergey Levine. "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning." In: *arXiv preprint arXiv:1710.11248* (2017).

[44]Justin Gilmer et al. "Neural message passing for quantum chemistry." In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1263–1272.

[45]Ross Girshick. "Fast r-cnn." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[46]Raghav Goyal et al. "The" Something Something" Video Database for Learning and Evaluating Visual Common Sense." In: *ICCV*. Vol. 1. 4. 2017, p. 5.

[47]Daniel Graves et al. "Learning robust driving policies without online exploration." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

[48]Mona Gridseth. "Visual Task Specification User Interface for Uncalibrated Visual Servoing." In: (2015).

[49]Mona Gridseth et al. "ViTa: Visual task specification interface for manipulation with uncalibrated visual servoing." In: *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June (2016), pp. 3434–3440. ISSN: 10504729. DOI: 10.1109/ICRA.2016.7487521.

[50]Mona Gridseth et al. "Vita: Visual task specification interface for manipulation with uncalibrated visual servoing." In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 3434–3440.

[51]Jiuxiang Gu et al. "Scene graph generation with external knowledge and image reconstruction." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1969–1978.

[52]Qipeng Guo et al. "Star-transformer." In: *arXiv preprint arXiv:1902.09113* (2019).

[53]Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor." In: *International conference on machine learning*. PMLR. 2018, pp. 1861–1870.

[54]Gregory D Hager and Zachary Dodds. "On specifying and performing visual tasks with qualitative object models." In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE. 2000, pp. 636–643.

[55]Gregory D. Hager and Zachary Dodds. "On specifying and performing visual tasks with qualitative object models." In: *Proceedings-IEEE International Conference on Robotics and Automation* 1.April (2000), pp. 636–643. ISSN: 10504729. DOI: 10.1109/ROBOT.2000.844124.

[56]William L. Hamilton. "Graph Representation Learning." In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14.3 (), pp. 1–159.

[57]Kai Han et al. "A Survey on Visual Transformer." In: *arXiv preprint arXiv:2012.12556* (2020).

[58]Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[59]João Pedro Hespanha et al. "What tasks can be performed with an uncalibrated stereo vision system?" In: *International Journal of Computer Vision* 35.1 (1999), pp. 65–85.

[60]Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning." In: (2016), pp. 4565–4573.

[61]Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[62]Yongqiang Huang and Yu Sun. "A dataset of daily interactive manipulation." In: *The International Journal of Robotics Research* 38.8 (2019), pp. 879–886.

[63]Ahmed Hussein et al. "Imitation learning: A survey of learning methods." In: *ACM Computing Surveys (CSUR)* 50.2 (2017), pp. 1–35.

[64]Seth Hutchinson, Gregory D Hager, and Peter I Corke. "A tutorial on visual servo control." In: *IEEE transactions on robotics and automation* 12.5 (1996), pp. 651–670.

[65]Auke Jan Ijspeert et al. "Dynamical movement primitives: learning attractor models for motor behaviors." In: *Neural computation* 25.2 (2013), pp. 328–373.

[66]Katsushi Ikeuchi and Takashi Suehiro. "Toward an assembly plan from observation. I. Task recognition with polyhedral objects." In: *IEEE transactions on robotics and automation* 10.3 (1994), pp. 368–385.

[67]INRIA. *ViSP Project*. 2020. URL: `http://visp.inria.fr/` (visited on 01/07/2021).

[68]M. Jagersand, O. Fuentes, and R. Nelson. "Experimental evaluation of uncalibrated visual servoing for precision manipulation." In: *Proceedings of International Conference on Robotics and Automation* 4.April (1997), pp. 2874–2880. ISSN: 10504729. DOI: `10.1109/ROBOT.1997.606723`.

[69]Martin Jagersand. *CMPUT615: 3-Dimensional Computer Vision*. 2020. URL: `http://ugweb.cs.ualberta.ca/~vis/courses/gradCompVis12/` (visited on 01/07/2021).

[70]Eric Jang et al. "Grasp2vec: Learning object representations from self-supervised grasping." In: *arXiv preprint arXiv:1811.06964* (2018).

[71]Jun Jin et al. "Evaluation of state representation methods in robot hand-eye coordination learning from demonstration." In: *arXiv preprint arXiv:1903.00634* (2019).

[72]Jun Jin et al. "Robot eye-hand coordination learning by watching human demonstrations: a task function approximation approach." In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 6624–6630.

[73]Jun Jin et al. "A Geometric Perspective on Visual Imitation Learning." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 2655–2662.

[74]Jun Jin et al. "Mapless Navigation among Dynamics with Social-safety-awareness: a reinforcement learning approach from 2D laser scans." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6979–6985.

[75]Jun Jin et al. "Visual geometric skill inference by watching human demonstration." In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8985–8991.

[76]Justin Johnson et al. "Inferring and executing programs for visual reasoning." In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2989–2998.

[77]Julius Kammer. *A ROS driver for OpenNI depth (+ RGB) cameras.* [Online; accessed 5-Sept-2019]. 2019. URL: `https://github.com/ros-drivers/openni_camera`.

[78]Manuel Kaspar, Juan D Muñoz Osorio, and Jürgen Bock. "Sim2real transfer for reinforcement learning without dynamics randomization." In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 4383–4388.

[79]Kel Guerin. *IVRE – An Immersive Virtual Robotics Environment*. [Online; accessed 5-Sept-2019]. 2021. URL: `https://cirl.lcsr.jhu.edu/research/human-machine-collaborative-systems/ivre/`.

[80]Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." In: *arXiv preprint arXiv:1312.6114* (2013).

[81]Jens Kober et al. "Movement templates for learning of hitting and batting." In: *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE. 2010, pp. 853–858.

[82]George Konidaris et al. "Robot learning from demonstration by constructing skill trees." In: *The International Journal of Robotics Research* 31.3 (2012), pp. 360–375.

[83]Petar Kormushev, Sylvain Calinon, and Darwin G Caldwell. "Robot motor skill coordination with EM-based reinforcement learning." In: *2010 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2010, pp. 3232–3237.

[84]Tejas Kulkarni et al. "Unsupervised learning of object keypoints for perception and control." In: *arXiv preprint arXiv:1906.11883* (2019).

[85]Yasuo Kuniyoshi, Masayuki Inaba, and Hirochika Inoue. "Learning by watching: Extracting reusable task knowledge from visual observation of human performance." In: *IEEE transactions on robotics and automation* 10.6 (1994), pp. 799–822.

[86]Michael Laskin et al. "Reinforcement learning with augmented data." In: *arXiv preprint arXiv:2004.14990* (2020).

[87]Michelle A Lee et al. "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks." In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8943–8950.

[88]Daniel Leidner, Christoph Borst, and Gerd Hirzinger. "Things are made for what they are: Solving manipulation tasks by using functional object classes." In: *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*. IEEE. 2012, pp. 429–435.

[89]Sergey Levine et al. "End-to-end training of deep visuomotor policies." In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 1334–1373.

[90]Sergey Levine et al. "Offline reinforcement learning: Tutorial, review, and perspectives on open problems." In: *arXiv preprint arXiv:2005.01643* (2020).

[91]Xueting Li et al. "Joint-task self-supervised learning for temporal correspondence." In: *Advances in Neural Information Processing Systems*. 2019, pp. 318–328.

[92]Changsong Liu et al. "Jointly learning grounded task structures from language instruction and visual demonstration." In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 1482–1492.

[93]Jia Liu et al. "A taxonomy of everyday grasps in action." In: *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE. 2014, pp. 573–580.

[94]YuXuan Liu et al. "Imitation from observation: Learning to imitate behaviors from raw video via context translation." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1118–1125.

[95]Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[96]David G Lowe et al. "Object recognition from local scale-invariant features." In: *iccv*. Vol. 99. 2. 1999, pp. 1150–1157.

[97]Cewu Lu et al. "Visual relationship detection with language priors." In: *European conference on computer vision*. Springer. 2016, pp. 852–869.

[98]Ajay Mandlekar et al. "Roboturk: A crowdsourcing platform for robotic skill learning through imitation." In: *Conference on Robot Learning*. PMLR. 2018, pp. 879–893.

[99]Lucas Manuelli et al. "kpam: Keypoint affordances for category-level robotic manipulation." In: *arXiv preprint arXiv:1903.06684* (2019).

[100]Haggai Maron et al. "Invariant and equivariant graph networks." In: *arXiv preprint arXiv:1812.09902* (2018).

[101]Joanna Materzynska et al. "Something-else: Compositional action recognition with spatial-temporal interaction networks." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1049–1059.

[102]Alessio Micheli. "Neural network for graphs: A contextual constructive approach." In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 498–511.

[103]Hiroyuki Miyamoto et al. "A kendama learning robot based on bi-directional theory." In: *Neural networks* 9.8 (1996), pp. 1281–1302.

[104]Shiwali Mohan and John Laird. "An object-oriented approach to reinforcement learning in an action game." In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Vol. 6. 1. 2011.

[105]Katharina Mülling et al. "Learning to select and generalize striking movements in robot table tennis." In: *The International Journal of Robotics Research* 32.3 (2013), pp. 263–279.

[106]Don Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[107]Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding." In: *arXiv preprint arXiv:1807.03748* (2018).

[108]Deepak Pathak et al. "Zero-shot visual imitation." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 2050–2053.

[109]David Paulius, Nicholas Eales, and Yu Sun. "A Motion Taxonomy for Manipulation Embedding." In: *Robotics: Science and Systems*. 2020.

[110]David Paulius and Yu Sun. "A survey of knowledge representation in service robotics." In: *Robotics and Autonomous Systems* 118 (2019), pp. 13–30.

[111]David Paulius et al. "Functional object-oriented network for manipulation learning." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 2655–2662.

[112]Chris Paxton et al. "CoSTAR: Instructing collaborative robots with behavior trees and vision." In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017, pp. 564–571.

[113]Jan Peters and Stefan Schaal. "Reinforcement learning of motor skills with policy gradients." In: *Neural Networks* 21.4 (2008), pp. 682–697. ISSN: 08936080. DOI: 10.1016/j.neunet.2008.02.003. arXiv: arXiv:1411.3159v1.

[114]Thomas Pierrot et al. "Learning Compositional Neural Programs for Continuous Control." In: *arXiv preprint arXiv:2007.13363* (2020).

[115]Xuebin Qin et al. "Real-Time Edge Template Tracking via Homography Estimation." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 607–612.

[116]Zengyi Qin et al. "KETO: Learning Keypoint Representations for Tool Manipulation." In: *arXiv preprint arXiv:1910.11977* (2019).

[117]Camilo Perez Quintero et al. "Flexible virtual fixture interface for path specification in tele-manipulation." In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 5363–5368.

[118]Aravind Rajeswaran et al. "Epopt: Learning robust neural network policies using model ensembles." In: *arXiv preprint arXiv:1610.01283* (2016).

[119]Oscar A. Ramirez and Martin Jagersand. "Practical considerations of uncalibrated visual servoing." In: *Proceedings - 2016 13th Conference on Computer and Robot Vision, CRV 2016* (2016), pp. 164–169. DOI: 10.1109/CRV.2016.44.

[120]ROBOTIQ Inc. *How to do Machine Tending using a Collaborative Robot? - Robotiq*. [Online; accessed 5-Sept-2019]. 2021. URL: https://www.youtube.com/watch?v=GF8E9vGEYu8&ab_channel=Robotiq.

[121]Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning." In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 627–635.

[122]Emanuele Rossi et al. "Temporal graph networks for deep learning on dynamic graphs." In: *arXiv preprint arXiv:2006.10637* (2020).

[123]Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF." In: *ICCV*. Vol. 11. 1. Citeseer. 2011, p. 2.

[124]C Samson and B Espiau. "Application of the task-function approach to sensor-based control of robot manipulators." In: *IFAC Proceedings Volumes* 23.8 (1990), pp. 269–274.

[125]Claude Samson, Bernard Espiau, and Michel Le Borgne. *Robot control: the task function approach*. Oxford University Press, Inc., 1991.

[126]Ashutosh Saxena et al. "Robobrain: Large-scale knowledge engine for robots." In: *arXiv preprint arXiv:1412.0691* (2014).

[127]Tom Schaul et al. "Universal value function approximators." In: *International conference on machine learning*. PMLR. 2015, pp. 1312–1320.

[128]Tanner Schmidt, Richard Newcombe, and Dieter Fox. "Self-supervised visual descriptor learning for dense correspondence." In: *IEEE Robotics and Automation Letters* 2.2 (2016), pp. 420–427.

[129]Pierre Sermanet, Kelvin Xu, and Sergey Levine. "Unsupervised perceptual rewards for imitation learning." In: *arXiv preprint arXiv:1612.06699* (2016).

[130]Pierre Sermanet et al. "Time-Contrastive Networks: Self-Supervised Learning from Video." In: *arXiv preprint arXiv:1704.06888* (2017).

[131]Pierre Sermanet et al. "Time-contrastive networks: Self-supervised learning from video." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[132]Azad Shademan, Amir-Massoud Farahmand, and Martin Jägersand. "Robust jacobian estimation for uncalibrated visual servoing." In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 5564–5569.

[133]Rutav Shah and Vikash Kumar. "RRL: Resnet as representation for Reinforcement Learning." In: *Self-Supervision for Reinforcement Learning Workshop-ICLR 2021*. 2021.

[134]Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. "Third-Person Visual Imitation Learning via Decoupled Hierarchical Controller." In: *Advances in Neural Information Processing Systems*. 2019, pp. 2593–2603.

[135]Maximilian Sieb et al. "Graph-Structured Visual Imitation." In: *arXiv preprint arXiv:1907.05518* (2019).

[136]Geraldo Silveira and Ezio Malis. "Direct visual servoing: Vision-based estimation and control using only nonmetric information." In: *IEEE Transactions on Robotics* 28.4 (2012), pp. 974–980.

[137]Abhineet Singh and Martin Jagersand. "Modular tracking framework: A fast library for high precision tracking." In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 3785–3790.

[138]Joakim Skarding, Bogdan Gabrys, and Katarzyna Musial. "Foundations and modelling of dynamic networks using Dynamic Graph Neural Networks: A survey." In: *arXiv preprint arXiv:2005.07496* (2020).

[139]SpeedBot Inc. *SpeedBot Inc.* [Online; accessed 5-Sept-2019]. 2021. URL: `https://new.qq.com/omn/20200910/20200910A05W9000.html?pc`.

[140]Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. "Third-person imitation learning." In: *arXiv preprint arXiv:1703.01703* (2017).

[141]Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[142]Thaddeus Tarpey and Bernard Flury. "Self-Consistency: A Fundamental Concept in Statistics." In: *Statistical Science* 3 (1996), pp. 229–243.

[143]Laetitia Teodorescu, Katja Hofmann, and Pierre-Yves Oudeyer. "Recognizing Spatial Configurations of Objects with Graph Neural Networks." In: *arXiv preprint arXiv:2004.04546* (2020).

[144]Ashish Vaswani et al. "Attention is all you need." In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[145]Mel Vecerik et al. "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards." In: *arXiv preprint arXiv:1707.08817* (2017).

[146]ViwiStar. *ViwiStar*. [Online; accessed 5-Sept-2019]. 2019. URL: http://www.viwistar.com/english.

[147]Kaixin Wang et al. "Improving generalization in reinforcement learning with mixture regularization." In: *arXiv preprint arXiv:2010.10814* (2020).

[148]Xiaolong Wang, Allan Jabri, and Alexei A Efros. "Learning correspondence from the cycle-consistency of time." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2566–2576.

[149]Donglai Wei et al. "Learning and using the arrow of time." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8052–8060.

[150]Wikipedia contributors. *Truncated normal distribution*. [Online; accessed 5-Sept-2019]. 2019. URL: https://en.wikipedia.org/wiki/Truncated%5C_normal%5C_distribution.

[151]Wikipedia contributors. *Truncated normal distribution*. [Online; accessed 5-Sept-2019]. 2019. URL: https://en.wikipedia.org/wiki/Truncated%5C_normal%5C_distribution.

[152]Ben Williams, Marc Toussaint, and Amos J Storkey. "Modelling motion primitives and their timing in biologically executed movements." In: *Advances in neural information processing systems*. Citeseer. 2008, pp. 1609–1616.

[153]Thomas Wolf et al. "Transformers: State-of-the-art natural language processing." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 38–45.

[154]Zonghan Wu et al. "A comprehensive survey on graph neural networks." In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[155]Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. "Maximum entropy deep inverse reinforcement learning." In: *arXiv preprint arXiv:1507.04888* (2015).

[156]Annie Xie et al. "Few-shot goal inference for visuomotor learning and planning." In: *arXiv preprint arXiv:1810.00482* (2018).

[157]Bowen Xie et al. "A Generative Model-Based Predictive Display for Robotic Teleoperation." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021.

[158]Caiming Xiong et al. "Robot learning with a spatial, temporal, and causal and-or graph." In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 2144–2151.

[159]Danfei Xu et al. "Scene graph generation by iterative message passing." In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5410–5419.

[160]Danfei Xu et al. "Neural task programming: Learning to generalize across hierarchical tasks." In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[161]Yezhou Yang et al. "Robot learning manipulation action plans by" Watching" unconstrained videos from the world wide web." In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2015.

[162]Zihao Ye et al. "Bp-transformer: Modelling long-range context via binary partitioning." In: *arXiv preprint arXiv:1911.04070* (2019).

[163]YPC. *YPC Technologies*. [Online; accessed 5-Sept-2019]. 2019. URL: `https://www.ypc-technologies.com/`.

[164]Tianhe Yu et al. "One-Shot Imitation from Observing Humans via Domain-Adaptive Meta-Learning." In: *arXiv preprint arXiv:1802.01557* (2018).

[165]Vinicius Zambaldi et al. "Deep reinforcement learning with relational inductive biases." In: *International Conference on Learning Representations*. 2018.

[166]Philipp Zech et al. "Action representations in robotics: A taxonomy and systematic classification." In: *The International Journal of Robotics Research* 38.5 (2019), pp. 518–562.

[167]Rowan Zellers et al. "Neural motifs: Scene graph parsing with global context." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5831–5840.

[168]Andy Zeng et al. "3dmatch: Learning local geometric descriptors from rgb-d reconstructions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1802–1811.

[169]Amy Zhang et al. "Learning invariant representations for reinforcement learning without reconstruction." In: *arXiv preprint arXiv:2006.10742* (2020).

[170]Jie Zhou et al. "Graph neural networks: A review of methods and applications." In: *AI Open* 1 (2020), pp. 57–81.

[171]Dimitri Zhukov et al. "Learning actionness via long-range temporal order verification." In: *European Conference on Computer Vision*. Springer. 2020, pp. 470–487.

[172]Brian D. Ziebart et al. "Maximum Entropy Inverse Reinforcement Learning." In: *AAAI Conference on Artificial Intelligence* (2008), pp. 1433–1438. ISSN: 10450823. arXiv: `arXiv:1507.04888v2`.

[173]Konrad Zolna et al. "Offline learning from demonstrations and unlabeled experience." In: *arXiv preprint arXiv:2011.13885* (2020).

APPENDIX A

Additional results

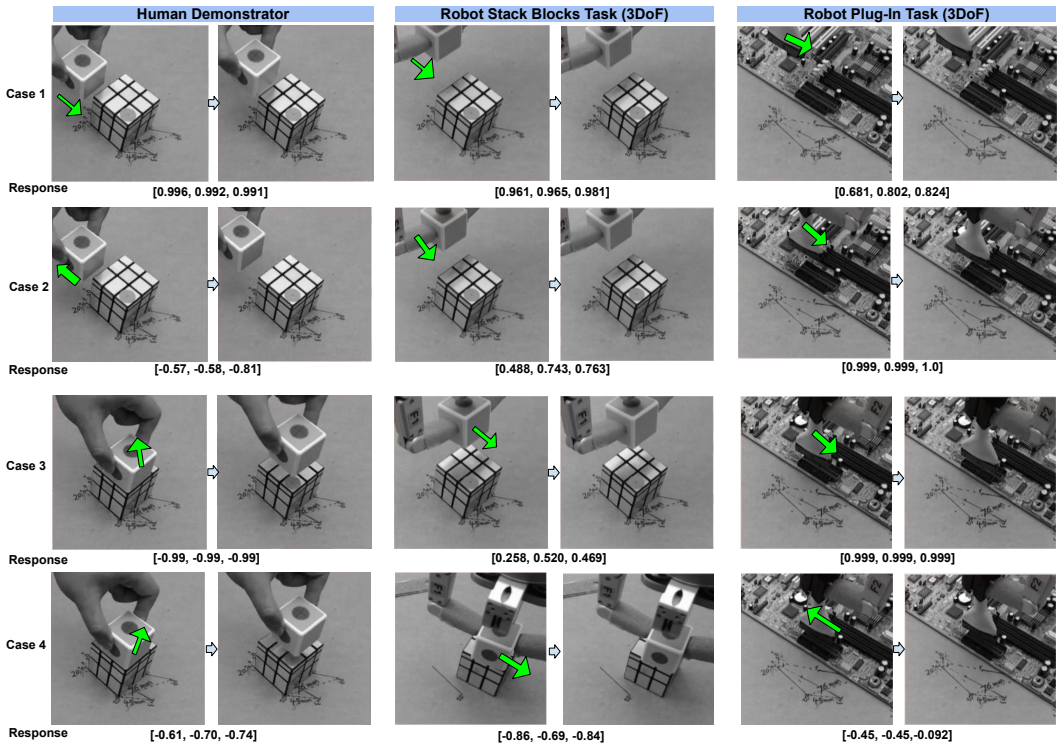


Figure A.1: Evaluation setup under different task/environment settings. Up row: initial settings; Down row: robot execution results, details are shown in Table 3.2. Results show that it can generalize well under moderately changed target positions and backgrounds, occlusions and illumination changes.

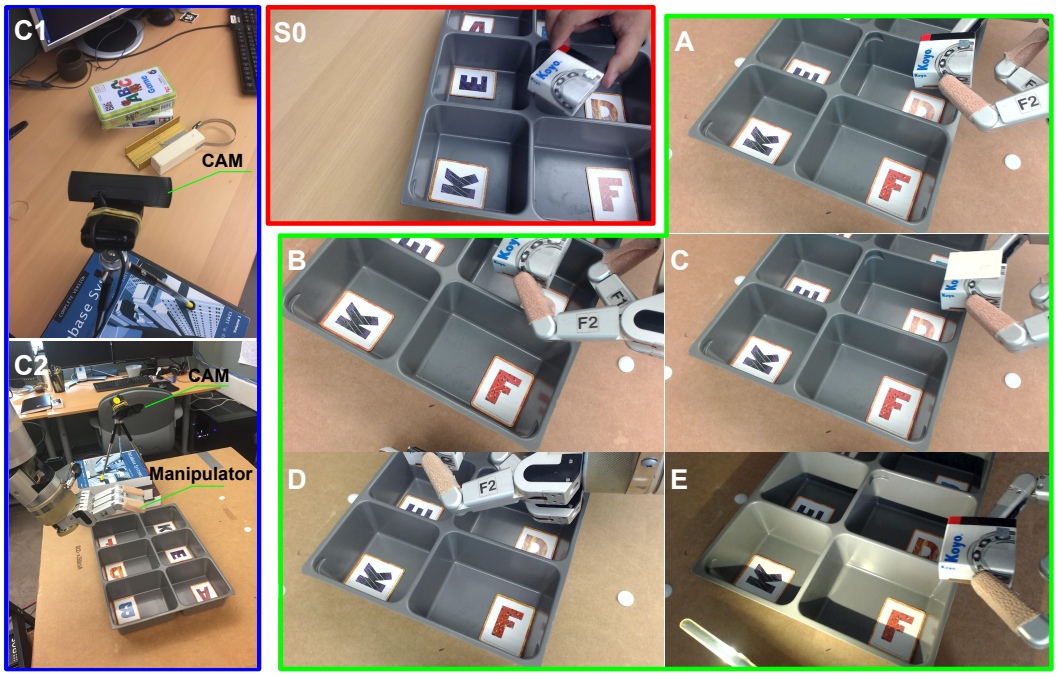


Figure A.2: **C1**: Human demonstration settings. **C2**: Robot imitation settings. **S0**: Human demonstration video used to train VGS-IRL. **A - E**: Evaluation on robot under five different environmental settings. A) *random target*; B) *change camera*; C) *object occlusion*; D) *object outside camera's FOV*; E) *change illumination*.

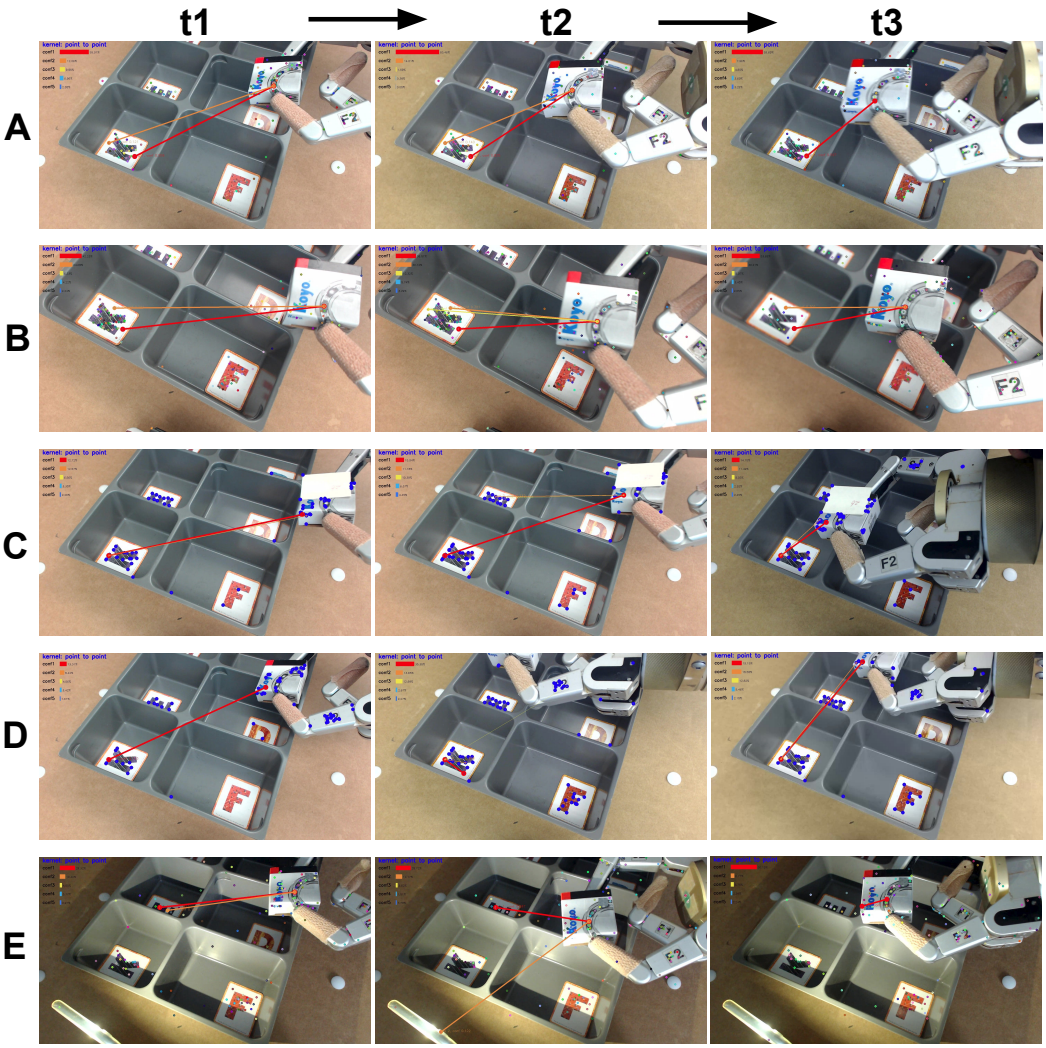


Figure A.3: Qualitative study of the learned task function under various environmental settings. **A**: random target; **B:** change camera; **C:** object occlusion; **D:** object running outside of FOV; **E:** changing illumination conditions.

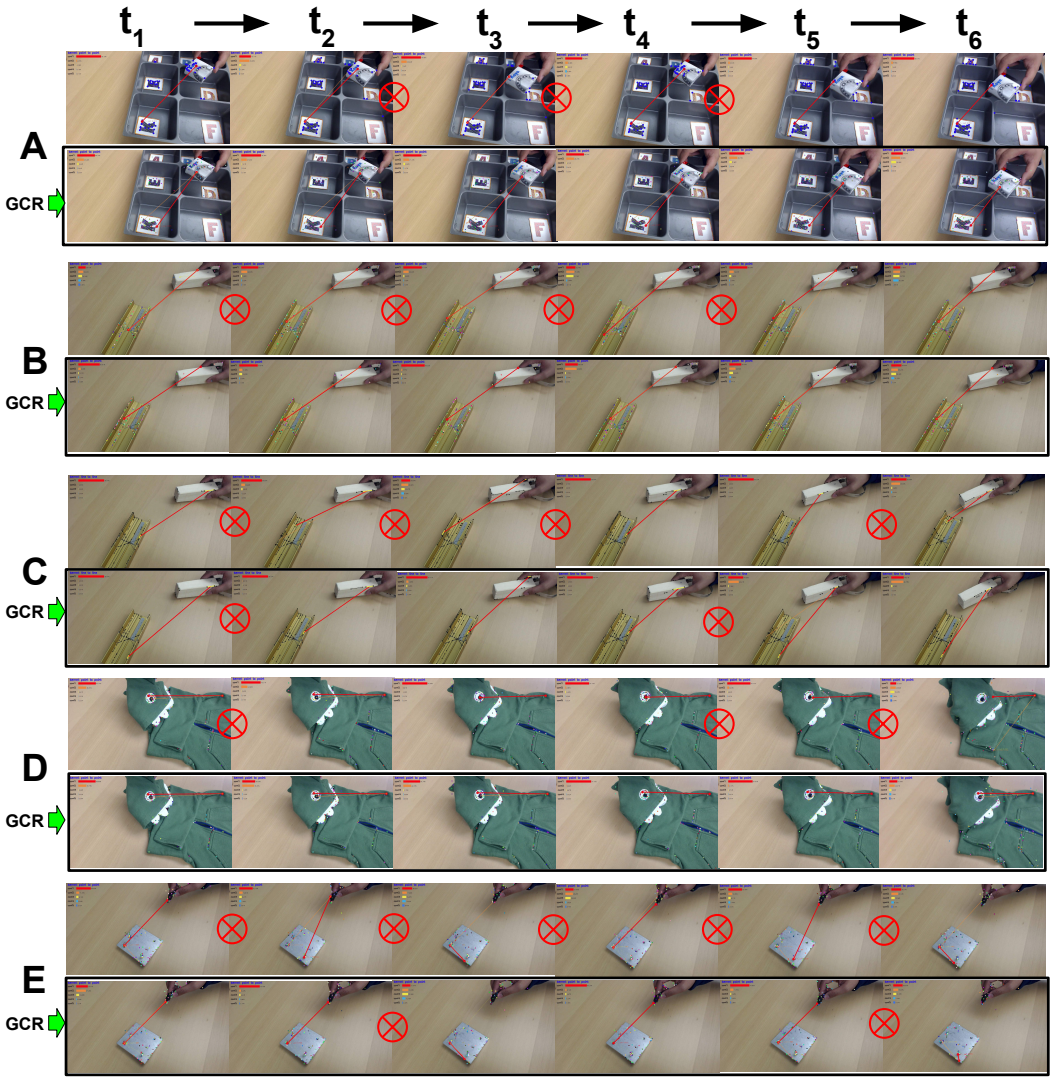


Figure A.4: Qualitative evaluations of GCR based on deep-feature method trained with and without GCR. **A:** a point-to-point *Sorting* task; **B, C:** a point-to-point and line-to-line *insertion* task; **D:** a point-to-point *folding* task; **E:** a point-to-point *screwing* task. 6 image frames for each task are selected from different time t_1, t_2, t_3, t_4, t_5 and t_6. Each of the same image is evaluated twice by a model trained without GCR (top row of each task) and with GCR (down row of each task) respectively. A circled red cross means inconsistent selection detected in the located frame transition.

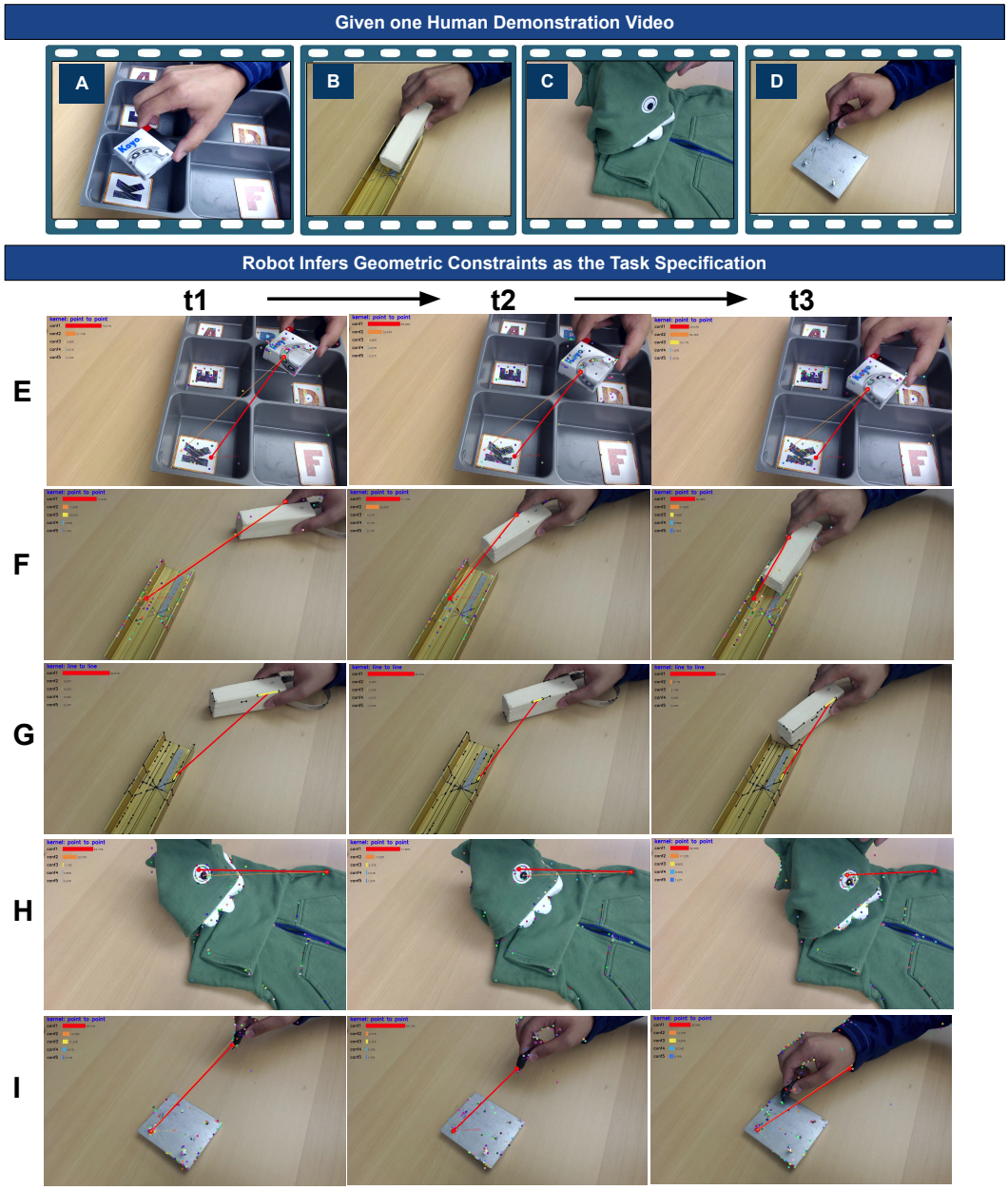


Figure A.5: Qualitative evaluations of different tasks. **A, B, C, D**: input human demonstration videos of four tasks. **E:** a point-to-point *Sorting* task; **F, G:** a point-to-point and line-to-line *insertion* task; **H:** a point-to-point *folding* task; **I:** a point-to-point *screwing* task. Each task is shown with 3 frames averagely selected spanning the whole input video sequence. Top five feature connections are selected out. Only the top one, as marked in red color, is used in evaluation to compute *Acc* and *ConAcc*.

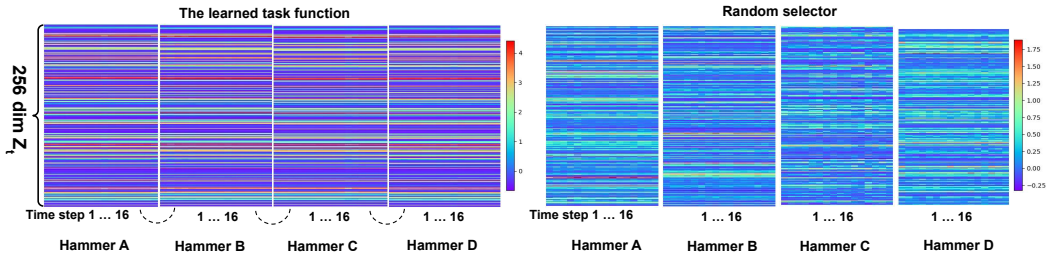


Figure A.6: Task-specification correspondence visualization of the hammering-PP task which involes two points coincidence to define the task. **Left:** the learned task function's output z_t. For convenience, we visualize the first component of z_t in 16 successive time steps. Results show that the embedding z_t for hammer A, B, C and D stays similar to each other while allowing slight value changes. This is done by selecting image features on objects and construct a graph to represent their geometric constraints. z_t is the representation of the constructed graph. **Right:** a random selector's output z_t. Results show a random selector, though selects image features and constructs the same graph structure; the embedding z_t of the four hammers do not match with each other. A complete visualziaton of all compooe

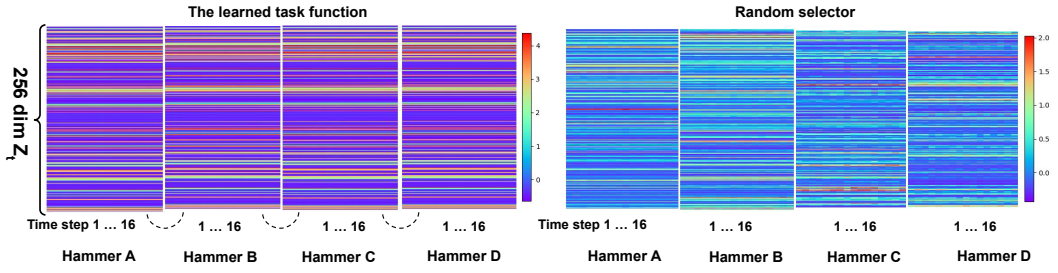


Figure A.7: Task-specification correspondence visualization of the hammering-LL task which involes two line parallelism to define the task. Again, results to the **left** are our task function's outputs that show the line-to-line constraint's representation z_t on categorical objects stays similar to each other while allowing individual varying factor changes. **Right** shows the results using a random selector. z_t is visualized using 16 successive frames' output and select the first component for visualization convenience.

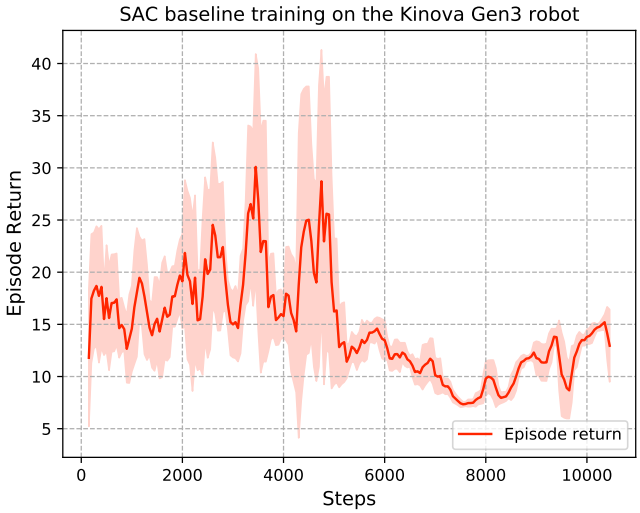
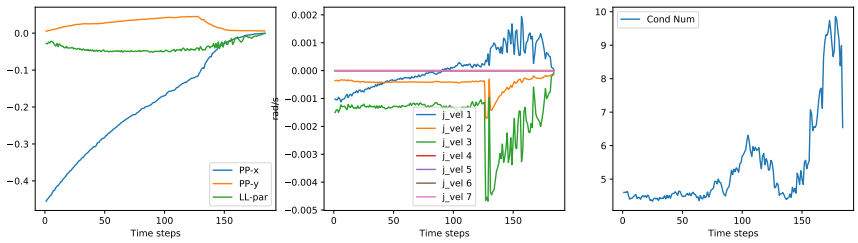
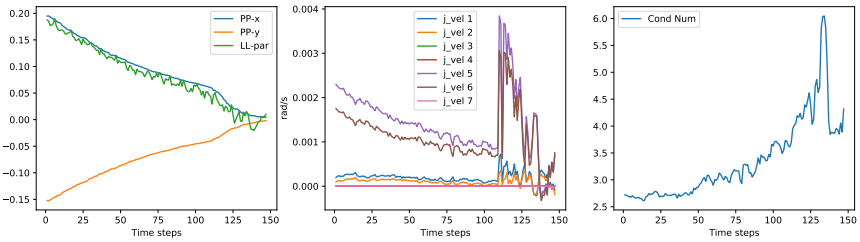


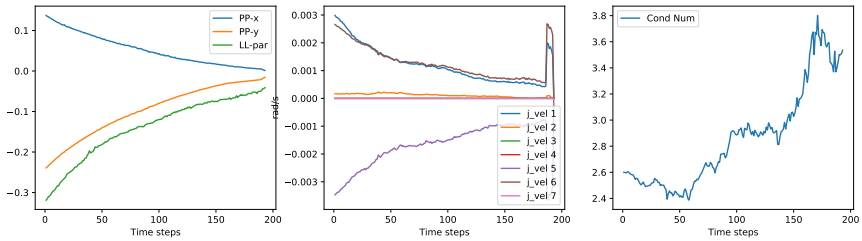
Figure A.8: Training curve of the baseline, RL (SAC [53]) agent using a reward classifier. The state consists of an image observation and the 7 joints positions and velocities. The policy's output action is a 7-dimensional continuous space that represents the seven joints' velocities. A reward classifier provides a reward signal that outputs the probability of successful state status within the range [0,1]. Traing step number is 100 for each episode. The training was not successful since the high dimensional state-action space of this problem. Fine-tuning the convolutional neural network or using a ResNet [133] backbone could help, but we did not try due to resource limits. The training was terminated after 10 K steps, which is about 6 hrs wall time. As a comparison, our proposed method requires human demonstration videos to learn "what" is the task, then map the learned task function to robot actions by uncalibrated visual servoing (UVS) controller.



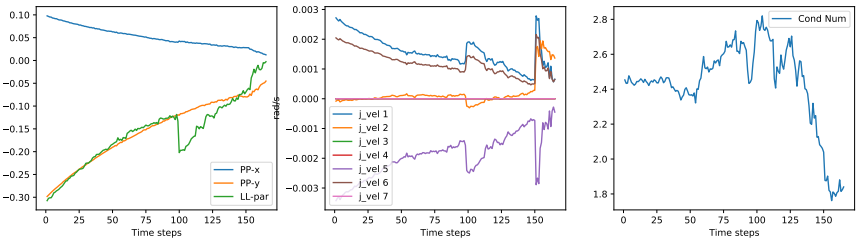
(a) Hammer A



(b) Hammer B



(c) Hammer C



(d) Hammer D

Figure A.9: Control curves of the hammering task by plugin the learned task function in a UVS controller. The task function was trained using human demonstrations of hammers A, B, and C. D is the new introduce hammer for testing. During UVS control, we use 4 joints in the task. **Left column**: image errors per time step when plugin the task function with the UVS controller. **Middle column:** joint velocity command from the UVS controller. The large joint velocity changes mostly happen when the estimation Jacobian matrix's condition number jumps high. This phenomenon also gives us the clue for future study of robust UVS. **Right column:** the Jacobian matrix's condition number per time step. During the experiment, the condition number of the estimated Jacobian remains low, which benefits from consistent geometric constraint selection and our normalized image errors described in Section 7.2.2.

APPENDIX B

Supplementary Material

B.1Task fucntion approximation from human demonstrations

B.1.1Conditions when the cost function is a constant

We prove that when $p(r_{tj})$ is a regular normal distribution in doman $[-\infty, \infty]$, the cost function Eq. (11) in our paper is a constant which is related to human factor[1] σ_0^2.

Firstly, let's review the cost function in Eq. (11):

$$\mathcal{L} = \arg\max_{\theta} \sum r_t^* - log\mathcal{Z}_t \tag{B.1}$$

, where $\mathcal{Z}_t$ is the partition function that integrates the exponential reward r_{tj} of all possible actions $\{a_{tj}\}$ when human demonstrator is at state s_t. Since human demonstrator makes selections only from promising actions instead of any uniform actions, we assume $r_{tj} \sim \mathcal{N}(r_t^*, \sigma_0)$, where r_t^* is reward from the selected action a_t^* that is observed in the demonstration. So $\mathcal{Z}_t$ can be written as:

[1] σ_0^2 is determined by human demonstrator's confidence level α.

$$\mathcal{Z}_t = \mathbb{E}_{p(r_{tj};r_t^*)}[exp(r_{tj})] \tag{B.2}$$

Considering a $[-\infty, \infty]$ domain of r_{tj}, we have:

$$\mathcal{Z}_t = \int_{-\infty}^{\infty} exp(r_{tj})p(r_{tj})dr_{tj} \tag{B.3}$$

where $p(r_{tj}) = \mathcal{N}(r_{tj}|r_t^*, \sigma_0)$, Eq. (3) can be rewritten as:

$$\begin{aligned}\mathcal{Z}_t = \frac{1}{\sqrt{2\pi}\sigma_0}\int_{-\infty}^{\infty} exp(&-\frac{1}{2\sigma_0^2}r_{tj}^2 + (\frac{r_t^*}{\sigma_0^2}+1)r_{tj} \\ &-\frac{1}{2\sigma_0^2}r_t^{*2})dr_{tj}\end{aligned} \tag{B.4}$$

Now $\mathcal{Z}_t$ has a standard form as a Gaussian integral, which is tractable in practice[151]:

$$\int_{-\infty}^{\infty} k\, exp(-fx^2 + gx + h)dx = k\sqrt{\frac{\pi}{f}}exp(\frac{g^2}{4f} + h) \tag{B.5}$$

So, we have:

$$\mathcal{Z}_t = exp(r_t^* + \frac{\sigma_0^2}{2}) \tag{B.6}$$

As a result, r_t^* is neutralized in the cost function, Eq. (1) can be rewritten as:

$$\mathcal{L} = \arg\max_{\theta} \sum -\frac{\sigma_0^2}{2} \tag{B.7}$$

which is now a constant related to human factor σ_0.

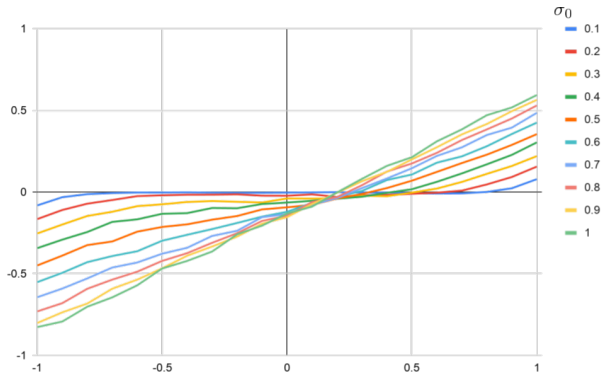


Figure B.1: Cost function values with different σ_0 and r_t^*.

B.1.2Cost function with truncated normal distribution

We empirically calculate the cost values given different r_t* and σ_0 (Fig. 2.). A Monte Carlo estimator with a sampling size=2000 is used for computation. Results show the cost value overall increases as r_t^* grows, however the slope is different. Lower σ_0 outputs a smaller gradient for learning the reward function while higher σ_0 outputs a larger one.

Intuitively, a lower σ_0 means human demonstrator is more confident in selecting actions, which will result the learned reward function easily over-fit to observed demonstrations. On the other side, a higher σ_0 means human demonstrator is not so confident in demonstration. So the demonstration samples have more randomness compared to smaller σ_0 demonstrations. Any updates in the resulting r_t^* should have more value in learning.

APPENDIX C

Task specification capability of geometric constraints: additional analysis

Total num of constraints	best η value	p-value (P_0=78.3%)	p-value (P_0=77.5%)
1	0.3	0.999	0.999
2	0.6	0.957	0.946
3	0.8	0.500	0.500
4	0.85	0.324	0.296
5	0.9	0.159	0.142
6	0.95	0.062	0.054
7	1	0.019	0.016
8	1	0.0186	0.016
9	1	0.0186	0.016
10	1	0.0186	0.016
11	1	0.0186	0.016
12	1	0.0186	0.016
13	1	0.0186	0.016
14	1	0.0186	0.016
15	1	0.0186	0.016
16	1	0.0186	0.016
17	1	0.0186	0.016
18	1	0.0186	0.016

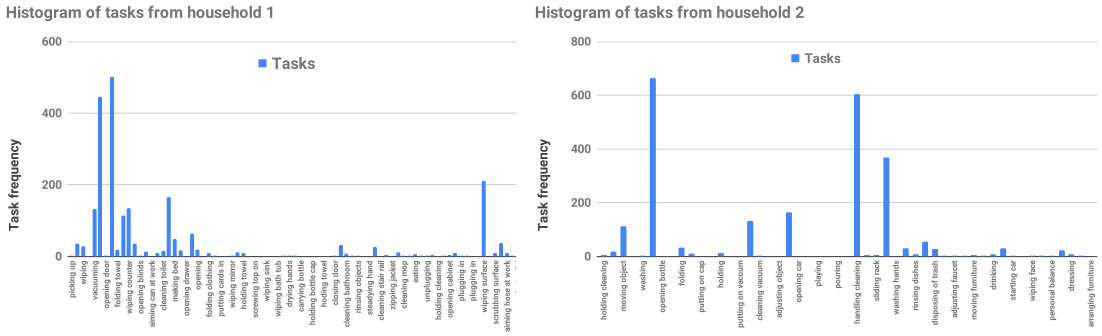
Table C.1: Statistican signifance test results of each test instance in $\mathbb{S}$.

Num of constraints	1	2	3	4	5	6	7	8	9
Num of test instances	3	6	10	15	21	28	33	36	37
Num of constraints	10	11	12	13	14	15	16	17	18
Num of test instances	36	33	28	21	15	10	6	3	1

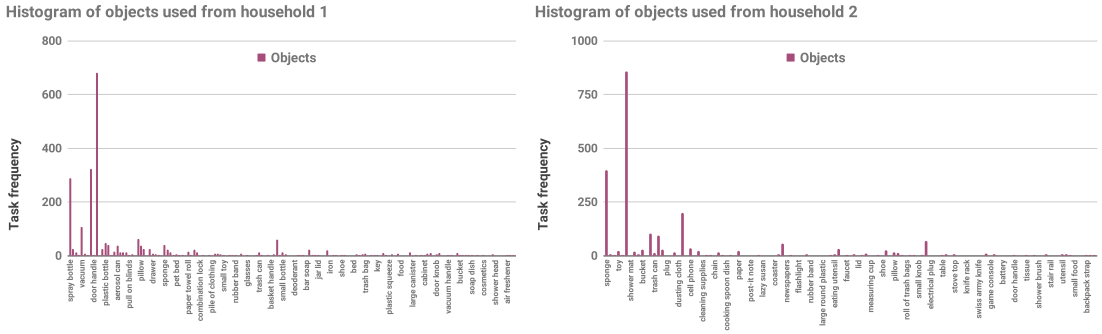
Table C.2: The correspondence between the total number of constraints and test instance number. Top row: total number of geometric constraints in a test instance. Bottom row: count of all test instances with the same number of constraints.

Action code	Video Num	Task name	Sub-skills	Vision task
m2	25	stir with spatula	stiring	1
m3	40	spinkle, shake pepper	move spinkle to pan	1
m4	25	spread/oil	spread	1
m6	25	vertical cut	cut	1
m7	35	use spoon to pick up		0
m8	25	pizza wheel	wheeling	1
m10	25	use black brush	brushing	1
m11	30	spear object using fork	spearing	1
m12	25	stir water using spoon	stiring	1
m13	40	fasten screw with screw driver	aligning	1
m14	0	loosen screw with screw driver		0
m15	75	unlock lock with key	aligning	1
m16	15	fasten nut with wrench	aligning	1
m17	25	use paint brush to dip and spread	dip, spread	1
m18	25	use hammer to hammer in nail		1
m19	25	brush teeth	touching teeth	1
m20	25	use file to file wooden thing	touching wood	1
m21	25	comb hair		1
m22	25	scrape substrace from surface	touching surface	1
m23	30	peel cucumber/potato	peeling	1
m24	25	slice cucumber	aligning in slicing	1
m25	74	flip bread	reaching and aligning	1
m26	25	use spoon to scoop and pour	reaching	1
m27	30	shave object	reaching	1
m28	30	use roller to roll out dough		0
m30	0	loosen nut with wrench		0
m31	30	scoop and pour with measuring spoon/cup		1
m32	0	insert peg into pegboard		0
m33	0	brush powder across grey tray		0
m34	0	insert straw through to-go cup lid		0
m35	25	m34 with eyes closed	insertion	1
m36	0	m31 without pour		0
Total	804		Valied vision tasks	24
			Avg. videos per task	33

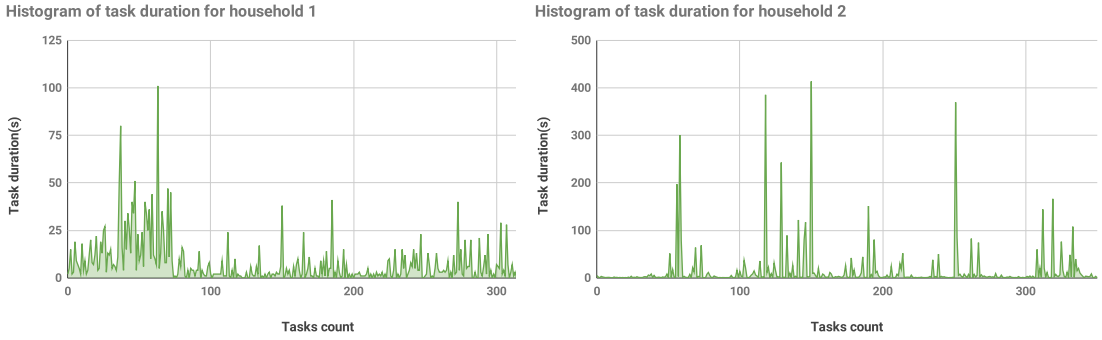
Table C.3: Statistics of the USF_DIM dataset [62].



(a) Top frequency tasks: arranging objects, handling cleaning suppliers, wiping surface wiping surface

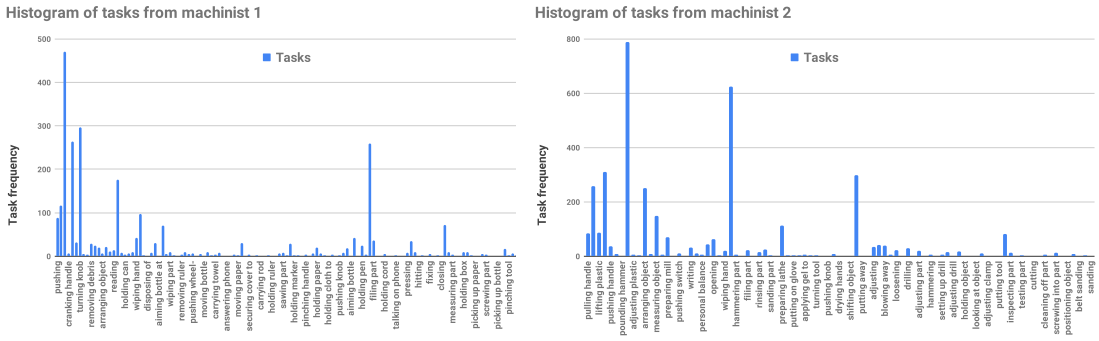


(b) Top frequency objects: Mop, spray bottle, sponge, towel.

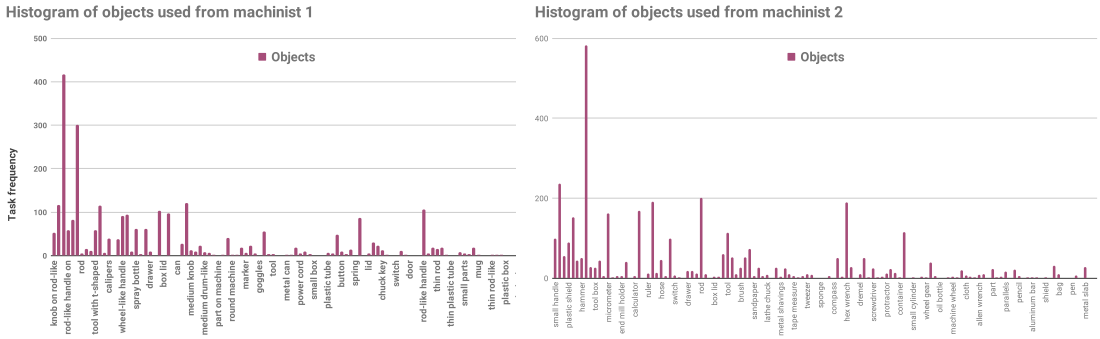


(c) Frequency of task durations

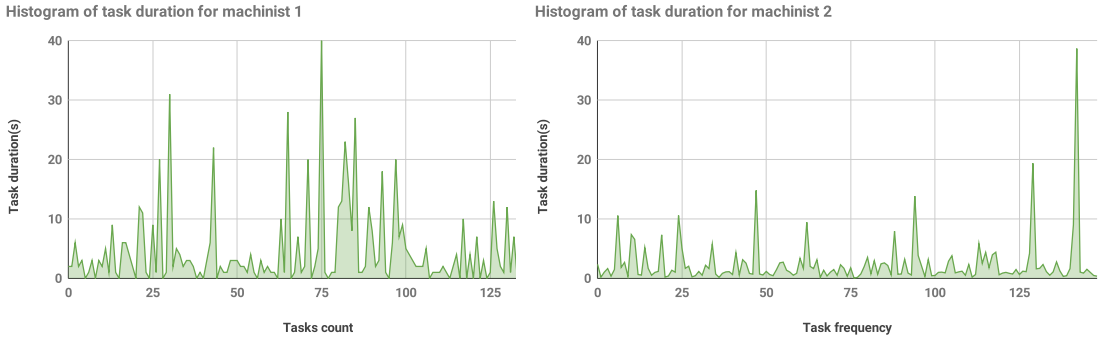
Figure C.1: Statistical analysis of the tasks in YAL_Grasp dataset [14], machine shop environment. In order to help us better study the task speciation capability using geometric constraints, we analyze our primary source in the task pool— the YALE_Grasp dataset. We compute the frequency of task types, objects and task duration to filter out the most common tasks and objects used in our everyday life. Results show that the top frequency tasks and their sub-skills relate to geometric constraint-based task specification. Furthermore, the top frequent objects have regular shapes that could be convenient to use geometric features to define their task functionality. Meanwhile, the analysis of task duration provides a hint for us to design a real-world robotic system that serves humans in our workspace or homes.



(a) Top frequency tasks: holding part, turning knob, holding tool, moving object.



(b) Top frequency objects: small part, small knob, ruler, rod



(c) Frequency of task durations

Figure C.2: Task statistics of YALE_GRASP [14] dataset-household environment. Detailed analysis stays the same as described in Fig. C.1.