Visual Geometric Skill Inference by Watching Human Demonstration

Jun Jin†, Laura Petrich†, Zichen Zhang†, Masood Dehghan† and Martin Jagersand†

Abstract—We study the problem of learning manipulation skills from human demonstration video by inferring association relationship between geometric features. Our motivation comes from the observation in human eye-hand coordination that a set of manipulation skills are actually minimizing the Euclidean distance between geometric primitives while regressing their association constraints in non-Euclidean space. We propose a graph based kernel regression method to directly infer the underlying association constraints from human demonstration video using Incremental Maximum Entropy Inverse Reinforcement Learning (InMaxEnt IRL). The learned skill inference provides human readable task definition and outputs control errors that can be directly plugged into traditional controllers. Our method removes the need of tedious feature selection and robust feature trackers in traditional approaches (e.g. feature based visual servoing). Experiments show our method reaches high accuracy even with only one human demonstration video and generalize well under variances.

I. INTRODUCTION

Understanding and applying the mechanism of learning by watching has more than two decades of research in robotics, where the core problem is to extract high-level reusable symbolic task plans by observing human demonstration [1], [2]. Most of the works are focused on learning task goal configuration rather than task execution [3]. This approach reduces the learning complexity and most importantly, extracts an abstract task representation which in turn enables task generalization. Symbolic task plans are represented as a tree [4] or graph structure [5], [6] based on the assumption that a task can be decomposed into low-level conditioned elementary skills [7] such as grasping, striking [8], alignment [9] and peg-in-hole [10]. In order to ground the symbols [3], (action, object, task) recognition techniques and predefined skills (sub-module) are hand engineered [3]. These predefined tasks are highly task-dependant and do not generalize well in practice.

The main question is whether there is a general solution to parametrize a task? There is no absolute answer, but even if there is a parametrization, it is difficult to find since manipulation tasks are too complex in general. One way of addressing this problem is to use low-level elementary skills as the corner stones of a task, which are potentially easier to be learned and generalized. Among the many types of skills, we are interested in those that can be generally parameterized using geometric association constraints (see Fig. 1) since a variety of skills can be further created from their combinations. We name them as geometric skills, which are inherently represented in the observation (image, point clouds) space by geometric primitives (points, lines, conics, planes, etc.) and align well with how human eye-hand coordination works [11]. This parameterization method was introduced by Dodds et al. [12] to solve the box-packing task and then implemented by Gridseth et al. [13] on various skills, including grasping, placing, insertion and cutting.

This approach, however, has several drawbacks. First, the task specification is tedious since we need to manually select geometric features and then assign an association constraint. Besides, it is highly dependent on robust feature trackers [14]. This paper aims to address these issues by learning from watching. We propose a method to directly regress the geometric association constraints on each frame.

The main contributions of this paper are:

- Remove the dependency on robust feature trackers: Geometric features are difficult to track but easy to detect. Feature extraction from image or point cloud is a well-studied computer vision topic. Directly estimating the association constraint will bypass the tracking process and avoid tracking errors.
- Provide an adaptive solution towards robust feature selection. On object level, a skill can be defined by several possible associations of geometric features. This stochastic property provides robustness. By contrast,

†Authors are with Department of Computing Science, University of Alberta, Edmonton AB., Canada, T6G 2E8. {jjin5, llorrain, zichen2, masood1, mj7}@ualberta.ca

†The earliest work can be traced back to Ikeuchi et al. [1] and Kuniyoshi et al. [2] in 1994.

2For example, when some features are occluded, the other candidates will make up and continue defining the task.
traditional tracking based methods fix such associations in the initial feature selection stage. This stiffness on constraints is removed in our maximum entropy based geometric constraint regression method.

- Provide a projective invariant representation of a skill by defining the relationship between geometric features. Also this invariance enables learning with off-the-shelf feature descriptors spanning from classical ones (SIFT, ORB, LBD [15] that manifest robust invariant properties such as rotation, scale, photometric) to deep learning ones [16], which in turn, enhance the generalization.

- Provide an interpretable robotic skill learning. The proposed parameterization of skills is easy to monitor and validate. Because of this interpretability, the representation of a task (skill) is disentangled from control, making it possible to plug in different controllers or policy learning methods.

There are two main challenges in the problem. First, how to generally encode different types of association constraints and build more complex geometric skills from them? Second is how to optimize such constraints given one human demonstration video\(^3\). These two challenges will be addressed in this paper.

II. RELATED WORKS

This paper is inspired by research works in robot learning, visual servoing and graph-based relational inference.

End to end learning by watching: This approach, which is commonly named as imitation learning [18], gained a lot of interest recently. Sermanet et al. presented TCN [19] to learn from contrastive positive and negative frame changes along time. Yu et al. proposed a meta-learning based method [17] to encode prior knowledge from a few thousands of human/robot demonstrations, then learned a new task from one demonstration. End to end learning approaches lack interpretability. Also, to the authors’ knowledge, learning by watching only from one human demonstration is still difficult.

Learning task plans by watching: This approach provides the most intuitive motivation and contributes many early works in learning by watching. Such approaches try to generate human readable symbolic representations at the semantic level [20], [4], [6] to provide high level task planning, which also make them promising to generalize well. Ikeuchi et al. presented a general framework [1] which relies on object/task/grasp recognition to generate assembly plans from observation. Modern approaches include using a grammar parser [4], causal inference [6] and neural task programming [21]. Konidaris et al. proposed constructing skill trees [22] in the trajectory level to acquire skills from human demonstration using hierarchical RL with options. This work presents a general framework to learn a tree level structured task. However such works either need hard coded recognition sub modules, or lacks generality in various tasks.

Geometric approaches in skill learning: Constructing skills using geometric features provides good interpretability. Apart from works mentioned in Sec. I, Ahmazadeh et al proposed a system called VSL [3] to learn skills from only one demonstration. VSL firstly detects objects on image and represent them using image feature extractors like SIFT. It computes object spatial motion changes by feature matching and then form a new task goal configuration, which is finally used to generate motion primitives by a trajectory-based learning from demonstration (LfD) method [23]. Landmark based pre/post action condition detection is also used to construct a task plan. Triantafyllou et al. proposed a geometric approach to solve the garment unfolding task [24]. Tremblay et al. proposed a human-readable plan generation method [25] which provides impressive interpretability by modeling the 3D wire frame of blocks, however needs prior 3D modelling in simulator training.

III. METHOD

A. Geometric Skill Kernels

Let $\mathcal{O}$ denote the observation space and $\mathcal{F}$ denote observed geometric feature\(^4\). Each feature has two parts: a descriptor $f_i$ that encodes locally invariant properties, and a coordinate parameter set $y_i$ that encodes globally geometric properties\(^5\).

A geometric skill kernel $k$ is a composite functional structure that describes association constraints between geometric features $(f_i, y_i) \in \mathcal{F}$. To ground our formalism, we show some basic examples:

- **point-to-point** $k_{pdp}$: the coincidence of two points.
- **point-to-line** $k_{pdl}$: a point fits in a line.
- **line-to-line** $k_{lll}$: a line aligns with another line.
- **coplanarity** $k_{cop}$: coplanar four points or two lines.

Each kernel has two parts: a **non-Euclidean part** that encodes geometric association constraints; an **Euclidean part** that generates control error\(^6\) to guide robot actions.

1) The non-Euclidean part: Inspired by graph motifs [27], each skill kernel is a unit graph with different structures. Let’s use an undirected graph $G = \{V, E\}$ to represent the association constraint, where nodes $V$ are variables that take input of feature descriptors $\{f_1, \ldots, f_n\}$, while edges $E$ define a fixed graph structure (as shown in Fig. 2A). For example, the graph for $k_{pdp}$ has two connected nodes, and each node $v_i$ corresponds to $f_i$. By feeding two points, we get a graph instance. Let a select-out function $g_k$ measure how relevant a graph instance is to define a skill. Then, we have:

$$g_k : G \rightarrow [0, 1] \in \mathbb{R}, \quad g_k(G(\{f_1, \ldots, f_n\})) \tag{1}$$

For example, in the ‘insertion’ skill (Fig. 2B), the graph instance of P3 and P4 has higher $g_k$ than that of P1 and P2, thus will be selected out.

\(^3\)An ideal solution should be able to learn from only one demonstration and exhibits better performance with more demonstrations, though methods requiring thousands of demonstrations are not practical [17].

\(^4\)points, lines, conics, planes, spheres etc. from an image or point cloud.

\(^5\)See more details on parameterization of geometric primitives in [26].

\(^6\)For example, the control error of a point-to-point kernel is the point distance, while of a point-to-line kernel is the dot product of their homogeneous coordinates. More examples can be found in [26].
A message passing graph neural network [29]. Each node B. Parameterization and generating control signals to guide robot action. Both the following optimization using human demonstrations E clouds. More examples are in [28], [13].

point-to-point and line-to-line associations. By feeding their corresponding descriptors \{f_i\} \rightarrow \mathbb{R}^d denotes the mapping that maps all nodes’ geometric parameters to a control error vector where d is the degree of freedom that this constraint contributes. For example, given a point-to-point skill, d = 2 for image points while d = 3 in point clouds. More examples are in [28], [13]. \(E_k\) will be used in both the following optimization using human demonstrations and generating control signals to guide robot action.

B. Parameterization

1) Parameterization of \(G\): \(G\) is parameterized by a T-layer message passing graph neural network [29]. Each node \(v_i \in G\) relates a h-dimensional hidden state \(h_i\). At layer t (or time step t), each node’s hidden state \(h_i\) is updated by three steps. a) pair-wise message generation \(M\):

\[
m_{i_1 \rightarrow i_2}^{t+1} = \mathcal{A}(h_{i_1}^{t}, h_{i_2}^{t})
\]

where \(h_i\) relates to any node \(v_j\) connected to \(v_i\). And b) message aggregation \(\mathcal{A}\) which collects all incoming messages:

\[
m_{i_1 \rightarrow \cdots \rightarrow i_n}^{t+1} = \mathcal{A}(m_{i_1 \rightarrow i_2}^{t+1}, \ldots, m_{i_n \rightarrow i_{n-1}}^{t+1})
\]

where \(\mathcal{A}\) is an aggregation function, e.g., average, max, or none.

2) The Euclidean part: Let \(E_k : \{y_1, \ldots, y_n\} \rightarrow \mathbb{R}^d\) denotes the mapping that maps all nodes’ geometric parameters to a control error vector where d is the degree of freedom that this constraint contributes. For example, given a point-to-point skill, d = 2 for image points while d = 3 in point clouds. More examples are in [28], [13]. \(E_k\) will be used in both the following optimization using human demonstrations and generating control signals to guide robot action.

B. Parameterization

1) Parameterization of \(G\): \(G\) is parameterized by a T-layer message passing graph neural network [29]. Each node \(v_i \in G\) relates a h-dimensional hidden state \(h_i\). At layer t (or time step t), each node’s hidden state \(h_i\) is updated by three steps. a) pair-wise message generation \(M\):

\[
m_{i_1 \rightarrow i_2}^{t+1} = \mathcal{A}(h_{i_1}^{t}, h_{i_2}^{t})
\]

where \(h_i\) relates to any node \(v_j\) connected to \(v_i\). And b) message aggregation \(\mathcal{A}\) which collects all incoming messages:

\[
m_{i_1 \rightarrow \cdots \rightarrow i_n}^{t+1} = \mathcal{A}(m_{i_1 \rightarrow i_2}^{t+1}, \ldots, m_{i_n \rightarrow i_{n-1}}^{t+1})
\]

where \(\mathcal{A}\) is an aggregation function, e.g., average, max, or none.

2) Parameterization of \(g_k\): Given one image, we can construct m graph instances by enumerating all possible geometric primitive combinations (e.g., point-to-point by listing association between any two points). Each instance \(\{G_i\}\) represents one association and will output its \(b_i\). A select-out function \(g_k\) outputs a relevance factor \(g_i\):

\[
g_i = g_k(G_i) = \text{softmax}(b_i, \{b_1, \ldots, b_m\})
\]

Now we define the control error \(E_c\) for a whole image:

\[
E_c = \sum_{i=1}^{m} g_i E_k
\]

C. InMaxEnt IRL for optimization

Given human demonstration video frames, we apply InMaxEnt IRL [9] for optimization. To this end, we define the reward function, which connects skill kernels to entropy models. By optimizing this reward function, skill kernel is also optimized. In practice, each skill kernel is optimized individually. Let’s use \(k_{p2p}\) as an example in the following discussion.
1) Reward function: Each state $s_t$ is an image related to a control error $E_{ct}$, where the subscript $t$ denotes the time step in RL. An optimized $k_{p2p}$ should consistently select ‘correct instances’ among all states. So during human demonstration we expect $E_{ct}$ to decrease globally (but not necessarily in each step). Intuitively, we should get a positive reward if we observe $E_{ct}$ decreases, otherwise negative. Let $\Delta E_{ct} = |E_{ct+1} - E_{ct}|$, we define:

$$r_t = \frac{2}{1 + \exp(\beta \Delta E_{ct})} - 1 \quad (7)$$

$r_t \in (-1, 1), \beta$ normalizes the scale of different skill kernels’ output $E_{ct}$.

2) The variational expert assumption: InMaxEnt IRL considers imperfect expert demonstrations with a confidence level $\alpha$. A higher confidence level results in smaller variance $\sigma_0$ in demonstration. We assume that in a human demonstration, at state $s_t$ the probability of selecting an action that transits to the observed state $s_{t+1}$ follows a Boltzmann distribution with conditions:

$$p(s_{t+1}|s_t) = \frac{1}{Z_t} \exp(r_t^* \sigma_0^2), \quad (8)$$

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

$$Z_t = \mathbb{E}_{p(r_{tj}|r_t^*)}[\exp(r_{tj})] \quad (9)$$

is the partition function, $r_{tj} \sim \mathcal{N}(r_t^*, \sigma_0^2)$ is a truncated normal distribution with domain in [-1,1]. This means expert prefers the action with highest reward among all possible actions $A_t = \{a_{tj}\}$. To emphasize high impact actions in $A_t$, suppose $a_{tj}$ gets reward $r_{tj}$, the chance of $a_{tj}$ included in the pool is: $p(r_{tj}) = \mathcal{N}(r_t^*, \sigma_0)$, which is called human factor [9] since it varies with human demonstrator’s confidence.

3) Loss function: To maximize the probability of observed human demonstration video sequence $p\{s_t\}$, by applying MDP property, we have:

$$\mathcal{L} = \arg\max_\theta \sum \log[p(s_{t+1}|s_t)] \quad (10)$$

With equation (8) and removing the last constant, the cost function can be further written as:

$$\mathcal{L} = \arg\max_\theta \sum r_t^* - \log Z_t \quad (11)$$

Note that if $p(r_{tj})$ has domain $(-\infty, \infty)$, the loss function is a constant. Proofs can be found on our website [30].

To force $g_k$ making selections more deterministic while considering ambiguity issue, a penalty regularizer $-\lambda RSW$ is added to the reward where $\lambda$ is a hyperparameter and $RSW$ is the residual sum of weights. It makes $g_k$ output major weights on selected $p$ alternatives while minimizing the residual sum of weights ($RSW$). Fig. 3 shows a comparison between with and without RSW penalty.

4) Optimization: The last item in eq. (11) is a constant and $Z_t$ is a function of $r_t^*$, which is further represented using skill kernels with parameters $\theta$. Then, we have:

$$\nabla_\theta \mathcal{L} = \sum \nabla_\theta r_t^* - \sum \frac{1}{Z_t} \nabla r_t^* \mathcal{L} \quad (12)$$

$\nabla_\theta r_t^*$ can be solved by back propagation from eq. (7) to the graph neural network in the skill kernel. $Z_t$ can be estimated by a Monte Carlo estimator sampling $s_1$ samples from the truncated normal distribution $p(r_{tj})$:

$$Z_t \approx \frac{1}{s_1} \sum_{s_1} \exp(r_{tj}), \quad r_{tj} \sim p(r_{tj}) \quad (13)$$

$\nabla_\theta Z_t$ is the derivative of an expectation. By applying the log derivative trick, we have:

$$\nabla r_t^* Z_t = \nabla r_t^* \mathbb{E}_{p(r_{tj})}[\exp(r_{tj})] = \mathbb{E}_{p(r_{tj})}[\exp(r_{tj}) \nabla r_t^* \log p(r_{tj})] \approx \frac{1}{s_2} \sum \exp(r_{tj}) \nabla r_t^* \log p(r_{tj}), \quad r_{tj} \sim p(r_{tj}) \quad (14)$$

Since $p(r_{tj})$ is tractable, it’s trivial to get:

$$\nabla r_t^* \log p(r_{tj}) = \frac{x_\mu}{\sigma_0} + \frac{\exp(-b_\mu^2/2) - \exp(-a_\mu^2/2)}{\sqrt{2\pi\sigma_0}[\phi(b_\mu - \phi(a_\mu))]} \quad (15)$$
\[
x_{\mu} = (r_{t_j} - r_{t_i}^*)/\sigma_0, \quad a_{\mu} = (-1 - r_{t_i}^*)/\sigma_0, \quad b_{\mu} = (1 - r_{t_i}^*)/\sigma_0, \quad \text{where } \phi \text{ is defined in [31].}
\]

By combining the above equations, \( \nabla_\theta L \) is solved.

The optimization on \( k_{p2p} \) is summarized in Algorithm 1.

D. From Skill Kernel to Skills

In this paper, we consider a skill is simply the combination of several skill kernels, namely kernel ensembles. There should be more advanced ways to construct a skill from different kernels, though not discussed here.

E. From Skill to Control

Given an image from camera, each skill kernel will select out several alternative association instances thus generate control error vectors. For example, the point-to-point \( k_{p2p} \) will output \( p \) vectors with structure \([err_x, err_y]\), which can be used on controllers, e.g., feature based visual servoing or UVS\(^9\) (uncalibrated visual servoing [32]).

IV. Experiments

A. Quantitative Evaluation

We firstly evaluate what types of skills that the learned inference behavior is capable of. Four types are tested (Fig. 4): Sorting skill represents a regular setting; Insertion is for skills that need line-to-line constraint; Folding is for manipulation with deformable objects; Screw skill represents types that have low image textures. Each skill is evaluated on videos that show human performing the same task but with random behaviors. The objective is to infer the correct geometric feature associations that can define the demonstrated skill.

Then we test if the learned behavior from human demonstration video directly generalizes to robot hand. In our test, we also change the background table, as well as randomly arrange the target pose (Fig. 7A).

Lastly, we keep testing on the robot, but within 4 other scenarios (Fig. 7B-E) w.r.t. moving camera, occlusion, object running out of camera’s field of view and illumination changes.

8For example in a ‘peg-in-hole’ skill, point-to-point kernel should be first used to coarsely move to the target, while line-to-line kernel best fits in the final alignment actions. Their relationship is not a simple combination.

9M. Gridseht et al. 2016 [13] explores using more geometric features (lines, conics) and skill kernels in UVS control.

Baseline: To our best knowledge, there are no existing methods that learn geometric feature associations by watching human demonstration. However, for comparison, we hand designed a baseline on the Sorting skill. The baseline needs human to manually select 10 pairs of feature points and initialize 20 trackers. Each pair has one point on object and the other on the target. All the 10 pairs simultaneously define the same skill (Fig. 5). So this baseline is quite robust. In evaluation, as long as one pair still defines the skill, we mark the baseline a successful trial.

Metric: We evaluate on each video frame and calculate the accuracy of inferences. For the baseline, when it fails on one frame, it can’t be resumed unless human hand select the features again. So we only report success or failure on the final result. For our method, failures can be automatically corrected in successive frames. While our method can output \( p \) inferred associations, we only pick the top one to evaluate.

1) Training: For each skill, we evaluate the point-to-point kernel using SIFT and ORB features respectively. For the Insertion skill, we add the line-to-line kernel using LBD [15] line descriptor. All kernels have the same graph layer size=512 with hidden state dimension=512 and \( p=10 \) alternatives (III-C.1). In training, we set the regularizer coefficient \( \lambda = 0.1 \), and human factor \( \sigma_0 = 0.55 \). Each kernel with different descriptors are trained individually. The training curve of

Fig. 5: The hand designed baseline requires human to specifically select 10 pairs of feature points to define the demonstrated skill.

Fig. 6: Results on the four skills.

Fig. 7: A: Experimental setup on the manipulator. B: We change the camera pose in evaluation by rotation and a random displacement.
Fig. 8: Training curve of the sorting skill. Top row: using SIFT descriptors. Bottom row: using ORB descriptors. A: loss value over iterations. B: RSW curve. C: control error output from selected associations over steps. D: reward output from selected associations over steps. Results show that using SIFT descriptors exhibits smoother learning. The association instance selections of both are stabilized when the loss converges and output control error stabilizes at the same level.

Fig. 9: Given one human demonstration video, we evaluate the learned behavior on 5 scenarios. A: using robot hand with a different background and random target pose; B: projective variance due to camera pose change; C: occlusion; D: object out of camera’s viewfield; E: illumination change. For each scenario, we detect all feature points and use colored line to mark the select-out associations. The top one is marked red and the bar next to it indicates the estimation confidence. Only the association with confidence greater than 10% is displayed. We observe that the learned inference behavior tends to select fixed association instances while showing the flexibility of selecting alternatives when fixed ones are not observable. We also observe the accuracy is highly related to the capability of SIFT descriptor. It reaches high accuracy under projective variances (B), however, fails under illumination changes (E).

Sorting skill is shown in Fig.8 as an example.

2) Results:

a) On different skills: Results (Fig 6) on the 4 skills show our method is capable of the Sorting and Insertion skill but performs moderately in folding and Screw skills. In experiments, we observed that when both object and target have rich textures, results are better. One reason is the use of SIFT or ORB that are local descriptors dependent on textures. We expect further improvement by using other local feature descriptors [16] [33]. We also find the more features that can be fed into the skill kernel, the better accuracy it performs. Due to our hardware GPU limitation, we can only test using a small number (60 in average) of features.

b) On various conditions: Fig. 9 lists results on various conditions. Generally, i) our method exhibits adaptive behaviors that when some associations are occluded, the selection of others will make it up; ii) our method exhibits robust behavior that the failure between frames doesn’t affect successive frames since it directly selects the feature association on each frame. Instead, the baseline depends on the initialization of trackers and continuous tracking.

Though results on the robot manipulator show our method can output the correct selections of geometric feature associations which can be directly used in controllers eg., visual servoing or UVS [13], due to resource limitations, we didn’t test with a plug-in controller. We leave this as our future work.

V. CONCLUSION

We propose a general method to infer association relationship between geometric features by watching human demonstration. While showing some promising results, there are issues that need to be further investigated. 1) Consistent control error output: while the result shows that our method tends to select a fixed set of associations, it can’t guarantee the selection consistency. One possible solution is to add constraints between frames. 2) Other local feature descriptors [16] [33] are worth trying for a better generalization ability. 3) The generalization to point cloud geometric primitive needs to be further studied.
VI. SUPPLEMENTARY MATERIALS

A. Conditions when the cost function is a constant

We prove that when \( p(r_{tj}) \) is a regular normal distribution with domain \([-\infty, \infty]\), the cost function Eq. (11) in our paper is a constant which is related to human factor \( r_0^2 \).

Firstly, let’s review the cost function in Eq. (11):

\[
L = \arg \max_{\theta} \sum r_{tj}^* - \log Z_t
\]

where \( Z_t \) 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 \( r_{tj} \sim N(r_0^*, \sigma_0) \), where \( r_0^* \) is reward from the selected action \( a_{tj}^* \) that is observed in the demonstration. So \( Z_t \) can be written as:

\[
Z_t = \mathbb{E}_{p(r_{tj})}[exp(r_{tj})]
\]

(17)

Considering a \([-\infty, \infty]\) domain of \( r_{tj} \), we have:

\[
Z_t = \int_{-\infty}^{\infty} exp(r_{tj}) p(r_{tj}) dr_{tj}
\]

(18)

where \( p(r_{tj}) = N(r_{tj}\mid r_0^*, \sigma_0) \), Eq. (3) can be rewritten as:

\[
Z_t = \frac{1}{\sqrt{2\pi}\sigma_0} \int_{-\infty}^{\infty} exp(- \frac{1}{2\sigma_0^2} r_{tj}^2 + \frac{r_0^*}{\sigma_0^2} + 1)r_{tj}
\]

\[
- \frac{1}{2\sigma_0^2} r_{tj}^2) dr_{tj}
\]

(19)

Now \( Z_t \) has a standard form as a Gaussian integral, which is tractable in practice[34]:

\[
\int_{-\infty}^{\infty} k exp(-fx^2 + gx + h)dx = k\sqrt{\frac{\pi}{f}} exp\left(\frac{g^2}{4f} + h\right)
\]

(20)

So, we have:

\[
Z_t = exp(r_0^* + \frac{\sigma_0^2}{2})
\]

(21)

As a result, \( r_0^* \) is neutralized in the cost function, Eq. (1) can be rewritten as:

\[
L = \arg \max_{\theta} \sum -\frac{\sigma_0^2}{2}
\]

(22)

which is now a constant related to human factor \( \sigma_0 \).

\( \sigma_0 \) is determined by human demonstrator’s confidence level \( \alpha \).

B. Cost function with truncated normal distribution

We empirically calculate the cost values given different \( r_0^* \) and \( \sigma_0 \) (Fig. 2). A Monte Carlo estimator with a sampling size=2000 is used for computation. Results show the cost value overall increases as \( r_0^* \) grows, however the slope is different. Lower \( \sigma_0 \) outputs a smaller gradient for learning the reward function while higher \( \sigma_0 \) outputs a larger one.

Intuitively, a lower \( \sigma_0 \) 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 \( \sigma_0 \) means human demonstrator is not so confident in demonstration. So the demonstration samples have more randomness compared to smaller \( \sigma_0 \) demonstrations. Any updates in the resulting \( r_0^* \) should have more value in learning.

Fig. 11: Cost function values with different \( \sigma_0 \) and \( r_0^* \).

REFERENCES

[1] K. Ikeuchi and T. Suehiro, “Toward an assembly plan from observation. I. task recognition with polyhedral objects,” IEEE transactions on robotics and automation, vol. 10, no. 3, pp. 368–385, 1994.
[2] Y. Kuniyoshi, M. Inaba, and H. Inoue, “Learning by watching: Extracting reusable task knowledge from visual observation of human performance,” IEEE transactions on robotics and automation, vol. 10, no. 6, pp. 799–822, 1994.
[3] S. R. Ahmadzadeh, A. Paikan, F. Mastrogiavini, L. Natale, P. Kormushev, and D. G. Caldwell, “Learning symbolic representations of actions from human demonstrations,” in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 3801–3808.
[4] Y. Yang, Y. Li, C. Fermuller, and Y. Aloimonos, “Robot learning manipulation action plans by ‘watching’ unconstrained videos from the world wide web,” in Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.
[5] N. Shukla, C. Xiong, and S.-C. Zhu, “A unified framework for humanoid robot knowledge transfer,” in 2015 AAAI Fall Symposium Series, 2015.
[6] C. Xiong, N. Shukla, W. Xiong, and S.-C. Zhu, “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.
[7] R. Dillmann, M. Kaiser, and A. Ude, “Acquisition of elementary robot skills from human demonstration,” in International symposium on intelligent robotics systems. Citeseer, 1995, pp. 185–192.
[8] J. Kober, K. Mülling, O. Krammer, C. H. Lampert, B. Schölkopf, and J. Peters, “Movement templates for learning of hitting and batting,” in Robotics and Automation (ICRA), 2010 IEEE International Conference on. IEEE, 2010, pp. 833–838.
[9] J. Jin, L. Petrich, M. Dehghan, Z. Zhang, and M. Jagersand, “Robot eye-hand coordination learning by watching human demonstrations: a task function approximation approach,” arXiv preprint arXiv:1810.00159, 2018.

[10] G. Schectter, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solorowjow, and S. Levine, “Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards,” arXiv preprint arXiv:1906.05841, 2019.

[11] S. Hutchinson, G. D. Hager, and P. I. Corke, “A tutorial on visual servo control,” IEEE transactions on robotics and automation, vol. 12, no. 5, pp. 651–670, 1996.

[12] Z. Dodds, G. D. Hager, A. S. Morse, and J. P. Hespanha, “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.

[13] M. Gridseth, O. Ramirez, C. P. Quintero, and M. Jagersand, “ViTa: Visual task specification interface for manipulation with uncalibrated visual servoing,” Proceedings - IEEE International Conference on Robotics and Automation, vol. 2016-June, pp. 3434–3440, 2016.

[14] Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, Q. Bateux, E. Marchand, J. Leitner, F. Chaumette, P. C. V. Ser, Q. Bateux, and E. Marchand, “Visual servoing from deep neural networks,” arXiv preprint arXiv:1705.08940, 2017.

[15] L. Zhang and R. Koch, “An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency,” Journal of Visual Communication and Image Representation, vol. 24, no. 7, pp. 794–805, 2013.

[16] S. A. Winder and M. Brown, “Learning local image descriptors,” in 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2007, pp. 1–8.

[17] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine, “One-shot imitation from observing humans via domain-adaptive meta-learning,” arXiv preprint arXiv:1802.01557, 2018.

[18] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” Twenty-first international conference on Machine learning - ICML ’04, p. 1, 2004.

[19] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, “Time-contrastive networks: Self-supervised learning from video,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1134–1141.

[20] R. Dillmann, “Teaching and learning of robot tasks via observation of human performance,” Robotics and Autonomous Systems, vol. 47, no. 2-3, pp. 109–116, 2004.

[21] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, “Neural task programming: Learning to generalize across hierarchical tasks,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1–8.

[22] G. Konidaris, S. Künnersma, R. Grupen, and A. Barto, “Robot learning from demonstration by constructing skill trees,” The International Journal of Robotics Research, vol. 31, no. 3, pp. 360–375, 2012.

[23] G. Konidaris, S. Künnersma, R. Grupen, and A. Barto, “Reinforcement learning from demonstration,” Journal of Machine Learning Research, vol. 14, no. 1, pp. 305–338, 2013.

[24] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, “Time-contrastive networks: Self-supervised learning from video,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1134–1141.

[25] R. Dillmann, “Teaching and learning of robot tasks via observation of human performance,” Robotics and Autonomous Systems, vol. 47, no. 2-3, pp. 109–116, 2004.

[26] D. Xu, S. Nair, Y. Zhu, J. Gao, A. Garg, L. Fei-Fei, and S. Savarese, “Neural task programming: Learning to generalize across hierarchical tasks,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1–8.

[27] G. Konidaris, S. Künnersma, R. Grupen, and A. Barto, “Robot learning from demonstration by constructing skill trees,” The International Journal of Robotics Research, vol. 31, no. 3, pp. 360–375, 2012.