# Motion Planner Guided Visuomotor Policy Learning

Venkata Pradeep Kadubandi, Gautam Salhotra, Gaurav S. Sukhatme, Peter Englert

*Abstract*— **We learn a visuomotor policy from a motion planner where the input of the policy and planner are from different domains. The domain of the planner is a low-dimensional representation of the environment that consists of object poses and shapes whereas the policy observes a high-dimensional visual representation of the environment as input. The goal is to learn a visuomotor policy that imitates the behavior generated by a motion planner. In the training phase, the robot has access to the low-dimensional environment representation, while at inference time only the visual representation is observed. We first train a behavioral cloning policy in the low-dimensional environment representation and an autoencoder in the visual domain. Then, we combine both models into a single policy and fine-tune it on a low amount of demonstration data. In simulated experiments, we demonstrate the effectiveness of our approach and compare it to prior work.**

## I. Introduction

There has been a lot of interest in learning robot control policies directly from raw sensory image inputs [1]–[4]. One approach is to train the perception system jointly with the control policy in an end-to-end fashion. Another common approach has been to use representation learning techniques like autoencoders [2], [3] in an unsupervised fashion to learn latent representations of high-dimensional sensory inputs and use them in control policies. Training deep neural networks is at the core of these methods, which has various limitations. For instance, they require large amounts of training data and collecting such data is impractical in real world environments. Simulators can be an effective tool to gather large amounts of data, and typically also provide additional information about the robot and environment. For example, a low-dimensional state representation that consists of the shapes and positions of objects in an environment. When such a low-dimensional environment state is available, sampling-based motion planning algorithms or trajectory optimization techniques are able to generate smooth collision-free paths towards a goal state. Motion planning and trajectory optimization, however, rely on a known environment model and also do not work with high-dimensional raw sensory inputs like images.

In this work, we explore the question: Can we use a motion planner as an expert to guide visuomotor policy learning? The planner works in a domain where it has access to a low-dimensional environment representation specifying object poses and shapes. We use such a motion planner to generate a large amount of data for a given task. Then, we apply behavioral cloning on this planner data to learn a policy that can infer robotic actions from low-dimensional environment states. Our goal is to transfer this policy to a visuomotor policy that only takes a high-dimensional image as input. To aid learning such a policy, we first learn an image autoencoder to learn a latent environment representation that captures the key information of an environment. Therefore, we assume to have a separate dataset of environment images. For this dataset, we do not assume to have access to the low-dimensional environment state that corresponds to an image observation, which is usually much harder and impractical to obtain. For the third and final step, we assume to have few demonstrations of actions as ground truth along with image observations of the environment. This type of data is typically hard to acquire and our goal is to work with much less demonstration data compared to planner data or data for training the image autoencoder. We use the demonstration data to train an end-to-end policy in visual domain by combining the image encoder with the low-dimensional policy into a visuomotor policy (see Figure 1). Since the encoder typically learns a higher dimensional latent representation than the low-dimensional environment state, we will learn a mapping from the latent representation to environment state as part of this step.

We show preliminary results of two experiments in simulation. In the first one, a point robot has to reach various goals in a planar environment. The second experiment consists of a robot arm with three joints that has to reach a goal configuration from various initial configurations. The experiments show that our approach is able to learn effective policies from a low amount of demonstrations.

## II. Method

We consider the problem of policy search where the goal is to learn a policy $u_t = \pi(s_t)$ which outputs the actions $u_t$ for an agent based on the state $s_t$. The environment is a Markov decision process whose state evolves according to unknown non-linear dynamics. We assume that the robot state $x_t$, such as joint positions and velocities, is always available as input to the policy. For the environment, we consider two kinds of state representations: 1) a high-dimensional raw sensory state $I_t$ like an image from a camera mounted on the agent; 2) a more structured low-dimensional state $y_t$, for example a specification of the pose and shape of the goal and obstacles. Table I summarizes the terminology we use throughout this paper. The goal of our approach is to learn an end-to-end policy that maps high-dimensional images to actions. We construct this policy from an autoencoder and a

The authors are with the Computer Science Department, Viterbi School of Engineering, University of Southern California, Los Angeles. kaduband|salhotra|gaurav|penglert@usc.edu

Gaurav Sukhatme holds concurrent appointments as a Professor at USC and as an Amazon Scholar. This paper describes work performed at USC and is not associated with Amazon.

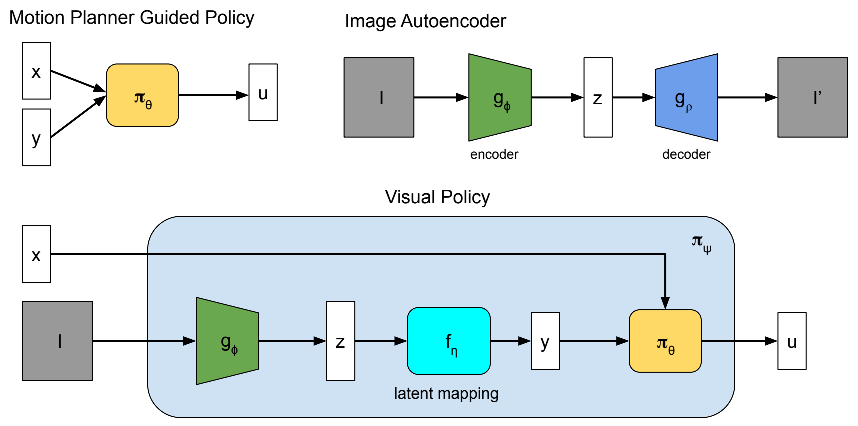

Fig. 1. Planner Guided Visual Learning. Top left shows behavior cloning from motion planner. Top right shows the image autoencoder. Bottom diagram shows how we combine them both to train visuomotor policy. Matching colors represent reusing neural network weights from one stage to another.

policy learned via behavioral cloning from a motion planner. A visual depiction of the method is shown in Figure 1. In the following sections, we explain the models and training scheme of our approach.

### A. Datasets and Objectives

We assume to have access to a motion planner that can generate an action trajectory $u_{0:T}$ for a given initial robot state $x_0$, initial environment state $y_0$, and final environment state $y_T$. We sample various inputs to the planner and execute the planned path in a simulator. The resulting state, action, and environment trajectories are stored in the *planner dataset*

$$\mathcal{D}_{\text{plan}} = \{(x_k, u_k, y_k)\}_{k=1\ldots M}$$

This data is used to train a policy using behavioral cloning by treating the motion planner as an expert. We call this motion planner guided policy. More details can be found in Section II-B.

In the visual domain, we assume to have a few demonstrations available where we know the corresponding action $u_t$ for an image $I_t$. This can be obtained by running the robot in the real world, taking a record of image observations $I_{0:T}$, actions $u_{0:T}$, and the reached robot states $x_{0:T}$. These trajectories are stored in the *demonstration dataset*

$$\mathcal{D}_{\text{demo}} = \{(x_k, u_k, I_k)\}_{k=1\ldots N}$$

Note that the amount of datapoints is assumed to be lower than in the planner dataset. We use the demonstration dataset to train the final policy that outputs an action for a given image. The details of this training are outlined in Section II-D.

In addition to the demonstration data in visual domain and the planner data in simulation, we assume to have access to a large amount of image observations. Here we do not assume that a specific goal or action is associated with the image observation. Our assumption is that such unlabeled data is much easier to generate. We use this data to learn a latent state representation of the high-dimensional image using an autoencoder. We call this *autoencoder dataset*

TABLE I
SUMMARY OF NOTATION USED IN THIS ARTICLE.

| **Symbol** | **Description** |
|---|---|
| $x_t$ | Robot state (position, velocity) |
| $y_t$ | Low-dimensional environment state including obstacle poses/shapes |
| $I_t$ | Image observation of environment |
| $u_t$ | Robot action |
| $z_t$ | Intermediate latent representation of high-dimensional image. This is typically lower dimensional than $I_t$, but higher dimensional than $y_t$. |
| $z_t = g_\phi^{\text{enc}}(I_t)$ | Image encoder parameterized by $\phi$ |
| $I_t = g_\rho^{\text{dec}}(z_t)$ | Image decoder parameterized by $\rho$ |
| $u_t = \pi_\theta^{\text{mp}}(x_t, y_t)$ | Behavioral cloning policy parametrized by $\theta$ |
| $u_t = \pi_\psi^{\text{img}}(x_t, I_t)$ | Visuomotor policy parameterized by $\psi$ |
| $y_t = f_\eta^{\text{latent}}(z_t)$ | Latent mapping parametrized by $\eta$ |

$$\mathcal{D}_{\text{auto}} = \{I_k\}_{k=1\ldots L} \; L \gg N$$

Details of training the autoencoder are outlined in Section II-C.

Our objective is to learn a policy that maps a high-dimensional image to an action, i.e., to learn a function $u_t = \pi_\psi^{\text{img}}(x_t, I_t)$. The complete algorithm is outlined in Algorithm 1. We use the planner data to learn a policy $u_t = \pi_\theta^{\text{mp}}(x_t, y_t)$ via behavioral cloning. An autoencoder $z_t = g_\phi^{\text{enc}}(I_t)$ is trained on $\mathcal{D}_{\text{auto}}$ such that the reconstruction loss is minimized. Finally, we combine two models into the policy $\pi_\psi^{\text{img}}$ and add a mapping from latent state $z_t$ to environment state $y_t$ i.e., $y_t = f_\eta^{\text{latent}}(z_t)$ that we fine-tune on the demonstration dataset. The output of our algorithm is an end-to-end policy that works with raw sensory image observations $u_t = \pi_\psi^{\text{img}}(x_t, I_t)$. In the following sections, we elaborate on the details of each of these steps.

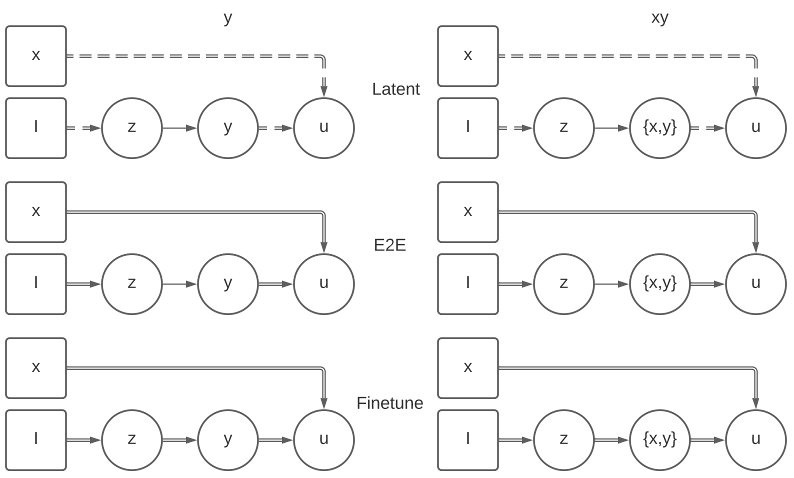

Fig. 2. Experiment Variations for combining and training $g_\phi^{\text{enc}}$, $f_\eta^{\text{latent}}$, $\pi_\theta^{\text{mp}}$ to produce a $\pi_\psi^{\text{img}}$. Broken arrow represents pre-trained frozen weights. Double arrow represents pre-trained and finetuned parameters. Rows represent variations in training procedure. In left column, the latent learning predicts $y_t$ from $z_t$. In right column, we predict $(x_t, y_t)$ i.e., robot state is also predicted as part of environment state.

### B. Motion planner Policy

Using the planner dataset $\mathcal{D}_{\text{plan}}$, we learn a policy that maps a low-dimensional environment state to an action i,e., $u_t = \pi_\theta^{\text{mp}}(x_t, y_t)$. Our key insight here is that we can obtain ground truth data for learning such a policy by running existing sampling-based motion planning algorithms and collecting data for robot state, action, and environment state. The loss function guiding this learning is given by:

$$\mathcal{L}_{\text{mp}}(\theta) = \sum_{(x_t,y_t,u_t)\in\mathcal{D}_{\text{plan}}} ||\pi_\theta^{\text{mp}}(x_t, y_t) - u_t||^2 \ . \qquad (1)$$

### C. Autoencoder

In order to learn a meaningful latent representation that reduces the dimensionality of the image, we train an autoencoder $z_t = g_\phi^{\text{enc}}(I_t)$ . In this step, we only use a reconstruction loss as the cue for representational learning

$$\mathcal{L}_{\text{auto}}(\rho, \phi) = \sum_{I_t\in\mathcal{D}_{\text{auto}}} ||g_\rho^{\text{dec}}(g_\phi^{\text{enc}}(I_t)) - I_t||^2 \ . \qquad (2)$$

In subsequent steps, when we train the image policy network end-to-end, we are fine-tuning the learned latent representation in a direction more suitable for solving the visuomotor task.

```
Algorithm 1: Planner Guided Visual Learning
  Function(PGVL)
  Result: Visual policy $\pi_\psi^{\text{img}}$
  Input: $\mathcal{D}_{\text{plan}}$, $\mathcal{D}_{\text{demo}}$, $\mathcal{D}_{\text{auto}}$
  Begin
      Train $\pi_\theta^{\text{mp}}$ with $\mathcal{D}_{\text{plan}}$ on $\mathcal{L}_{\text{mp}}(\theta)$ (Equation 1)
      Train autoencoder models $g_\phi^{\text{enc}}$ and $g_\rho^{\text{dec}}$ with $\mathcal{D}_{\text{auto}}$
        on $\mathcal{L}_{\text{auto}}(\rho, \phi)$ (Equation 2)
      Initialize $\pi_\psi^{\text{img}}$ with $g_\phi^{\text{enc}}$ and $\pi_\theta^{\text{mp}}$ (Equation 3)
      Fine-tune $\pi_\psi^{\text{img}}$ with $\mathcal{D}_{\text{demo}}$ on $\mathcal{L}_{\text{img}}$ (Equation 4)
  End
```

### D. Learning Latent Mapping

We combine the learned encoder $g_\phi^{\text{enc}}$ and policy $\pi_\theta^{\text{mp}}$ into a visual policy

$$\pi_\psi^{\text{img}}(x_t, I_t) = \pi_\theta^{\text{mp}}(x_t, f_\eta^{\text{latent}}(g_\phi^{\text{enc}}(I_t))) \qquad (3)$$

where the two models are connected with a latent mapping $y_t = f_\eta^{\text{latent}}(z_t)$. This gives us an end-to-end policy to work with images. We typically freeze the weights of the pre-trained encoder $\phi$ and low-dimensional policy $\theta$. Using the demonstration data, we train the sub-neural network that maps from the latent representation $z_t$ to the input of low-dimensional policy $y_t$ on the loss function

$$\mathcal{L}_{\text{img}}(\psi) = \sum_{(x_t,u_t,I_t)\in\mathcal{D}_{\text{demo}}} ||\pi_\psi^{\text{img}}(x_t, I_t) - u_t||^2 \ . \qquad (4)$$

## III. Experiments

We evaluate our method on two different tasks: The first task is a point robot in a planar 2D grid environment in which the robot and goal are initialized at random locations within the grid and the robot has a discrete action space. The second experiment is the MuJoCO reacher, which is a robot arm with three degrees of freedom. There, the robot has a continuous action space and has to move from random initial configurations to a desired goal configuration.

**Variations**: We considered a few different variations for combining the two networks and training procedures, which are outlined in Figure 2. For the training procedure, we considered the variations:

- *Latent/y*: Training the latent mapping from $z_t$ to $y_t$.
- *E2E/y*: Training end-to-end by not freezing the encoder and motion planner policy network parameters.

- *Finetune/y*: Training the latent mapping by freezing other parameters and then fine-tuning the network end-to-end by unfreezing all parameters.

In all the above settings, the latent state maps only to the environment state $y_t$. In another option we considered, the latent mapping predicts $(x_t, y_t)$ instead of $y_t$, which we denote as *Latent/xy*, *E2E/xy*, and *Finetune/xy*.

For comparison, we considered two baselines:

- *Behavioral Cloning from Images*: Training the visual policy directly from images in a supervised fashion. We use this as baseline in the MuJoCo Reacher experiment.
- *Environment Decoder with Motion Planner Policy*: Here, we assume to have a labeled dataset of environment state $y_t$ and the image observation $I_t$. We train the image decoder directly to predict $y_t$ from $I_t$. The prediction from this network is used as input to the low-dimensional policy to predict $u_t$. This baseline is used in the grid experiment.

We used different criteria to compare the different methods. In the evaluation, we applied the learned policy to the environment. We considered rollouts that exactly match the length of the ground truth trajectory and also rollouts with a fixed horizon (a sufficiently large number is chosen to be higher than all ground truth trajectories). While the first choice tells us how well the policy mimics the expert, the second choice signifies whether the agent stays at the goal if it was reached. We evaluate the following critera: goal deviation at the end of trajectory (how far is the reached goal from expected goal), trajectory error (the distance between robot and environment states of the ground truth trajectory and the rollout trajectory), and policy error (the distance between actions chosen by our agent and the ground truth actions). Note that when the rollout is longer than the ground truth, the latter two criteria consider only the first portion of it that matches in length to the ground truth. We also report the success rate of the rollouts that ended near the goal state (determined using a specific threshold value for goal deviation based on experimental setting).

## A. Point Robot in Planar Grid

For the point robot experiment, the image observation is a $32 \times 32$ grayscale image depicting a birds-eye view where the robot is a square centered at the robot position and the goal is another square centered at the desired location on the grid. The robot state $x_t$ and environment state $y_t$ are two-dimensional positions on the grid. The positions are determined based on dividing the grid into $32 \times 32$ cells. The robot's actions is also two-dimensional, which allow the robot to step along eight directions to neighboring grid. Note that this is a discrete action space. To generate ground truth planner data, we randomly sampled various environment and robot states and used a greedy planner, which produces a plan that moves the robot to the goal. A sample environment and a trajectory is shown in Figure 3. A trajectory is generated by first moving the robot to align diagonally to the goal and then moving along the appropriate diagonal direction to reach the goal.

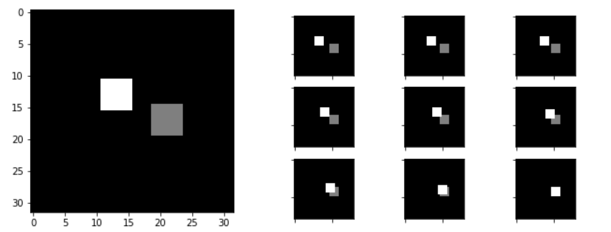

Fig. 3. Grid Environment and Sample Trajectory

In this experiment, we assume that both the image $I_t$ and corresponding environment state $y_t$ are available. So, both the plan and demo dataset are of the form:

$$\mathcal{D}_{\text{plan/demo}} = \{(x_k, u_k, y_k, I_k)\}_{i=1\dots N}$$

We generated 10000 trajectories for the training dataset and the total number of samples is $N = 130000$. We generated a separate test dataset with 500 trajectories containing $N' = 6500$ samples.

In the baseline method, we train an image encoder to learn a mapping from $I_t$ directly to $y_t$ given by:

$$\mathcal{L}_{\text{demo}}(\tau) = \sum_{(I_t, y_t) \in \mathcal{D}_{\text{demo}}} ||g_\tau^{\text{dec}}(I_t) - y_t||^2$$

The end-to-end policy for the baseline is then formed as

$$\pi_\psi^{\text{img}}(x_t, I_t) = \pi_\theta^{\text{mp}}(x_t, g_\tau^{\text{dec}}(I_t)) \ .$$

Table II shows the results we have observed (the experimental variations which are not shown did not produce decent results).

## B. MuJoCo Reacher

In this experiment, we used a robot arm with three rotational joints that is simulated in MuJoCo. The robot starts at a random initial configuration and the goal of the task is to reach a goal configuration. Thus both $x_t$ and $y_t$ are three-dimensional. The action $u_t$ represents a small desired change in the joint configuration at a time step and is also three-dimensional. For this experimental setup, the action space is continuous with joint limits of 0.15 radians for all the joints. A sample trajectory is shown in Figure 4.

We created different datasets for this experiment: a dataset with uniformly random sampled start and goal configurations; a fixed start position across the dataset; a fixed goal position across the dataset; Different image observation sizes among $(256, 256)$, $(128, 128)$ or $(64, 64)$. We observed good results for the the baseline method with fixed goal position and an image size of $(64, 64)$. The results are only shown for this dataset.

For this dataset, both $\mathcal{D}_{\text{plan}}$ and $\mathcal{D}_{\text{demo}}$ training sets contain 900 trajectories with 30000 samples. We used a separate test set with 100 trajectories and 3000 samples. To train the image autoencoder, we used a bigger dataset $\mathcal{D}_{\text{auto}}$ with 100000 images.

In this experiment, we used behavioral cloning on images as the baseline. Table III presents the results we observed from the baseline run and two variations of our method.

TABLE II
COMPARISON OF VARIOUS PERFORMANCE METRICS ACROSS GRID ENVIRONMENT RUNS.

| **Method/Metric** | **Policy Accuracy % (Avg)** | **Policy Accuracy % (Worst)** | **Trajectory Loss (Avg)** | **Goal Deviation (Avg)** | **Goal Deviation (Worst)** | **Success Rate (in Reaching goal)** |
|---|---|---|---|---|---|---|
| Environment Decoder with Motion Planner Policy (baseline) | 99.38 | 60 | 4.0e-3 | 2.5e-2 | 1.0 | 96.0% |
| PGVL: Latent/y | 88.64 | 0 | 2.5e-1 | 9.0e-1 | 24.5 | 60.8% |
| PGVL: Finetune/y | 99.18 | 60 | 1.2e-2 | **1.6e-2** | 1.0 | **97.6%** |
| PGVL: Latent/xy | 95.84 | 0 | 7.7e-2 | 1.2e-1 | 1.5 | 79.2% |
| PGVL: Finetune/xy | 99.06 | 0 | 1.3e-2 | **1.3e-2** | 1.0 | **98.6%** |

TABLE III
COMPARISON OF VARIOUS PERFORMANCE METRICS ACROSS REACHER RUNS.

| **Method/Metric** | **Goal Loss (Avg)** | **Goal Loss (Highest)** | **Trajectory Loss (Avg)** | **Policy Loss (Avg)** | **Success Rate (in Reaching goal)** |
|---|---|---|---|---|---|
| Behavioral cloning from Images (baseline) | 3.0e-2 | 8.3e-2 | 6.1e-2 | 9.2e-3 | 100% |
| PGVL: E2E/y | **5.1e-3** | **2.1e-2** | 5.5e-2 | 9.3e-3 | 100% |
| PGVL: E2E/xy | **1.6e-2** | **3.1e-2** | 7.9e-2 | 1.2e-2 | 100% |

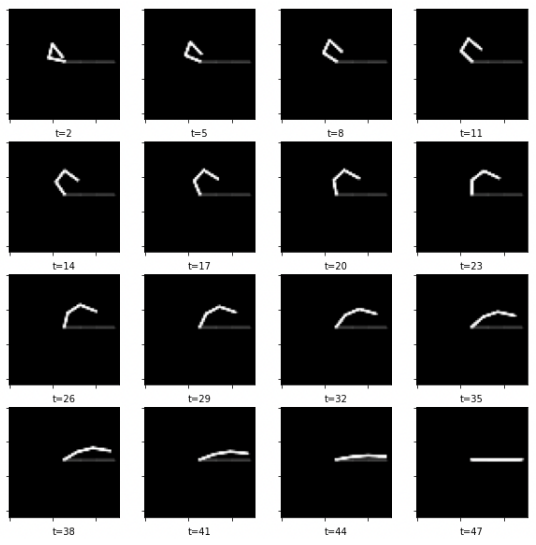

Fig. 4. A sub sample of Reacher Environment Sample Trajectory. The actual trajectory is 48 steps.

The image autoencoder is trained for 50 epochs. The motion planner guided policy is trained for 100 epochs. Afterwards, we fine-tuned the end-to-end network for 10 epochs and the resulting performance is similar to the baseline method. We compare different runs in the box plot in Figure 5 of goal deviations across the entire test set for different experiments. We observed that our approach has a better final goal deviation and reaches the goal closer across different test trajectories.

## IV. RELATED WORK

Controlling robots directly from raw sensory vision input has been an active area of research. Model-free Reinforcement Learning (RL) methods have been applied to this problem [1], [5], [6] for various tasks such as dexterous manipulation [7], [8], pick-and-place tasks [9], [10], etc.

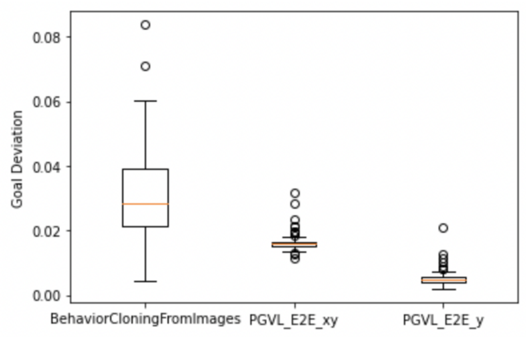

Fig. 5. Reacher Goal Deviation Box Plot Comparison.

Another approach is behavioral cloning [4], [11] where a policy is directly learned from expert demonstrations in a supervised end-to-end fashion. It is one of the baseline methods we used in our experiments. However these end-to-end learning methods often require a large number of samples and do not learn meaningful latent representations towards solving the problem. We use behavioral cloning as part of our method. However, we do not directly use it on images, but instead use the trajectory data generated from a motion planning algorithm as expert demonstrations to learn a policy that works with low-dimensional environment states.

Several works also take the route of learning latent representations towards solving the control problem. Embed to control [2] is closely related to our work in learning latent representations. A key difference is that they do not learn a policy directly and instead use a trajectory optimization algorithm in the latent space to infer optimal actions. The approach in [3] follows a different route of using self-supervised correspondence as the cue for learning intermediate representations. Our work uses motion planning algorithms to guide the policy learning towards learning latent representations.

Motion planning [12]–[17] can plan collision-free paths to a goal location in cluttered environments, using explicit representations of the robot and environment. Prominent techniques in sampling-based motion planning include probabilistic roadmaps [12]–[14] and rapidly-exploring random trees [15]–[17]. However, these methods typically work with low-dimensional state representations and cannot by applied directly to raw sensory input.

Combining motion planning algorithms with robot learning has been attempted in several works before. The standard methodology along this route is to break the problem down into parts that can and cannot be solved by the planner. Based on the generated motion plan, RL is used to learn the part that the planner cannot handle [18]–[24].

## V. Conclusion

In this work, we have shown that effective end-to-end policies from raw sensory inputs can be learned by augmenting the latent representation of an autoencoder with a policy that imitates a motion planner. Our experiments show that such policies match in quality with existing end-to-end learning methods and achieve a high goal accuracy.

In the current work, we assume that the underlying distribution governing the task where we apply motion planning and the goal task is the same. Extending our work to apply to tasks with different underlying data distributions is future work that could have important implications for sim-to-real transfer.

## References

[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *CoRR*, vol. abs/1504.00702, 2015.

[2] M. Watter, J. T. Springenberg, J. Boedecker, and M. A. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," *CoRR*, vol. abs/1506.07365, 2015.

[3] P. R. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *CoRR*, vol. abs/1909.06933, 2019.

[4] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[5] S. Gu, E. Holly, T. P. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation," *CoRR*, vol. abs/1610.00633, 2016.

[6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018.

[7] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *arXiv preprint arXiv:1808.00177*, 2018.

[8] A. Rajeswaran, V. Kumar, A. Gupta, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *CoRR*, vol. abs/1709.10087, 2017.

[9] Y. Zhu, Z. Wang, J. Merel, A. A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," *CoRR*, vol. abs/1802.09564, 2018.

[10] J. Yamada, Y. Lee, G. Salhotra, K. Pertsch, M. Pflueger, G. S. Sukhatme, J. J. Lim, and P. Englert, "Motion planner augmented reinforcement learning for obstructed environments," in *Conference on Robot Learning*, 2020.

[11] D. A. Pomerleau, "ALVINN: An Autonomous Land Vehicle in a Neural Network," in *Advances in Neural Information Processing System*, 1989.

[12] N. Amato and Y. Wu, "A randomized roadmap method for path and manipulation planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 1, 1996, pp. 113–120 vol.1.

[13] L. Kavraki and J. Latombe, "Randomized preprocessing of configuration for fast path planning," *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pp. 2138–2145 vol.3, 1994.

[14] M. Overmars, "A random approach to motion planning," Department of Computer Science, Utrecht University, Tech. Rep. RUU-CS-92-32, 1992.

[15] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Department, Iowa State University, Tech. Rep. TR 98-11, 1998.

[16] S. M. LaValle, J. J. Kuffner, B. Donald *et al.*, *Algorithmic and computational robotics: new directions.* AK Peters, 2001, ch. Rapidly-exploring random trees: Progress and prospects, pp. 293–308.

[17] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[18] P. Englert and M. Toussaint, "Learning manipulation skills from a single demonstration," *International Journal of Robotics Research*, vol. 37, no. 1, pp. 137–154, 2018.

[19] R. Vuga, B. Nemec, and A. Ude, "Enhanced Policy Adaptation Through Directed Explorative Learning," *International Journal of Humanoid Robotics*, vol. 12, no. 3, 2015.

[20] S. P. Singh, A. G. Barto, R. Grupen, and C. Connolly, "Robust reinforcement learning in motion planning," 1994, pp. 655–662.

[21] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, 2019.

[22] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese, "Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation," *CoRR*, vol. abs/2008.07792, 2020.

[23] T. Jurgenson and A. Tamar, "Harnessing reinforcement learning for neural motion planning," *CoRR*, vol. abs/1906.00214, 2019.

[24] B. Ichter and M. Pavone, "Robot motion planning in learned latent spaces," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2407–2414, 2019.