Long-Horizon Visual Planning with Goal-Conditioned Hierarchical Predictors

Karl Pertsch [1*] **Oleh Rybkin** [2*] **Frederik Ebert** [3] **Chelsea Finn** [4] **Dinesh Jayaraman** [2] **Sergey Levine** [3]

Abstract

The ability to predict and plan into the future is fundamental for agents acting in the world. To reach a faraway goal, we predict trajectories at multiple timescales, first devising a coarse plan towards the goal and then gradually filling in details. In contrast, current learning approaches for visual prediction and planning fail on long-horizon tasks as they generate predictions (1) without considering goal information, and (2) at the finest temporal resolution, one step at a time. In this work we propose a framework for visual prediction and planning that is able to overcome both of these limitations. First, we formulate the problem of predicting *towards a goal* and propose the corresponding class of latent space goal-conditioned predictors (GCPs). GCPs significantly improve planning efficiency by constraining the search space to only those trajectories that reach the goal. Further, we show how GCPs can be naturally formulated as hierarchical models that, given two observations, predict an observation between them, and by recursively subdividing each part of the trajectory generate complete sequences. This divide-and-conquer strategy is effective at long-term prediction, and enables us to design an effective hierarchical planning algorithm that operates in a coarse-to-fine manner. By using both goal-conditioning and hierarchical prediction, GCPs enable us to solve visual planning tasks with much longer horizon than previously possible.

1. Introduction

Intelligent agents aiming to solve long-horizon tasks reason about the future, make predictions, and plan accordingly. Several recent approaches [7, 34, 10, 33, 23, 12] employ

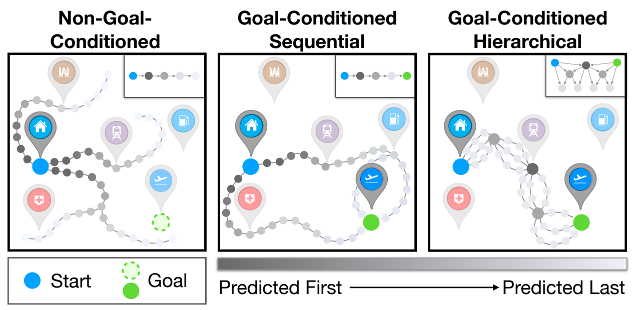

Figure 1. When planning towards faraway goals, we propose to condition the prediction of candidate trajectories on the goal, which significantly reduces the search space of possible trajectories (**left** vs. **middle**) and enables hierarchical planning approaches that break a long-horizon task into a series of short-horizon tasks by placing subgoals (**right**).

powerful predictive models [9, 1, 11, 20] to enable agents to predict and plan in complex environments directly from visual observations, without needing to engineer a state estimator. To plan a sequence of actions, these approaches usually use the predictive model to generate candidate rollouts starting from the current state and then search for the sequence that best reaches the goal using a cost function (see Fig. 1, left). However, such approaches do not scale to complex long-horizon tasks [7]. Imagine the task of planning a route from your home to the airport. The above approaches would attempt to model all possible routes starting at home and then search for those that ended up at the airport. For long-horizon problems, the number of possible trajectories grows very large, making extensive search infeasible.

In contrast, we propose a planning agent that only considers trajectories that start at home and end at the airport, i.e., makes predictions with the goal in mind. This approach both reduces prediction complexity as a simpler trajectory distribution needs to be modeled, and significantly reduces the planning search space, as depicted in Fig. 1 (center). Indeed, we can produce a feasible plan simply as a single forward pass of the generative model, and can further refine it to find the optimal plan through iterative optimization.

However, modeling this distribution becomes challenging for long time horizons even with goal-conditioned predictors. A naive method inspired by sequential predictive approaches would predict future trajectories at a fixed frequency, one step at a time — the equivalent of starting to

[*]Equal contribution [1]University of Southern California [2]University of Pennsylvania [3]UC Berkeley [4]Stanford University.

Presented at the Generalization in RL workshop, ICML 2020, Copyright 2020 by the authors. Project website: `sites.google.com/view/video-gcp`

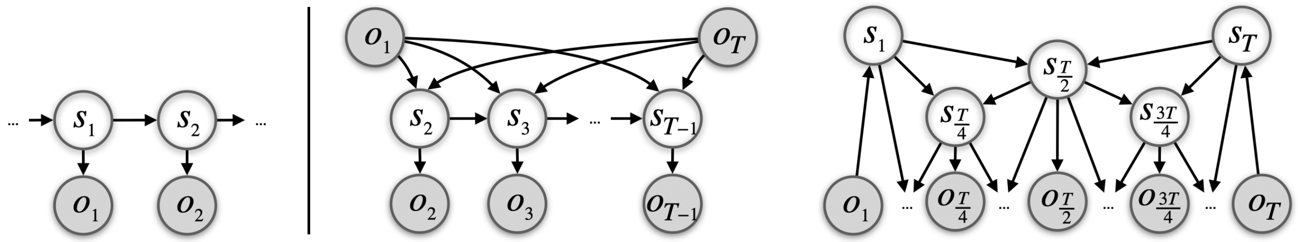

Figure 2. Graphical models for state-space sequence generation: forward prediction (left) and the proposed goal-conditioned predictors (GCPs). Shaded circles denote observations, white circles denote unobserved latent states. Center: a sequential goal-conditioned predictor with structure similar to forward prediction. Right: a hierarchical goal-conditioned predictor that recursively applies an infilling operator to generate the full sequence. All our models leverage stochastic latent states in order to handle complex high-dimensional observations.

plan the route to the airport by predicting the very first footsteps. This can lead to large accumulating errors. Moreover, the optimization problem of finding the best trajectory remains challenging. The sequential planning approaches are unable to focus on large important decisions as most samples are spent optimizing local variation in the trajectory. To alleviate both shortcomings, we propose to predict an a tree-structured way, starting with a coarse trajectory and recursively filling in finer and finer details. This is achieved by recursive application of a single module that is trained to answer: given two states, what is a state that occurs between them? This hierarchical prediction model is effective at long-term prediction and further enables us to design an efficient long-horizon planning approach by employing a coarse-to-fine trajectory optimization scheme.

In summary, the contributions of this work are as follows. First, we propose a framework for goal-conditioned prediction and planning that is able to scale to visual observations by using a latent state model. Second, we extend this framework to hierarchical prediction and planning, which improves both efficiency and performance through the coarse-to-fine strategy and effective parallelization. We further extend this method to modeling the temporal variation in subtask structure. Evaluated on a complex visual navigation task, our method scales better than alternative approaches, allowing effective control on tasks longer than possible with prior visual planning methods.

2. Goal-Conditioned Prediction

In this section, we formalize the goal-condition prediction problem, and propose models for goal-conditioned prediction, including both auto-regressive models and tree-structured models. To define the goal-conditioned prediction problem, consider a sequence of observations $[o_1, o_2, ...o_T]$ of length T. Standard forward prediction approaches (Fig 2, left) observe the first k observations and synthesize the rest of the sequence. That is, they model $p(o_{k+1}, o_{k+2}, \ldots o_{T-1}|o_1, o_2, \ldots o_k)$. Instead, we would like our goal-conditioned predictors to produce intermediate observations given the first and last elements in the sequence (Fig 2, center and right). In other words, they must model $p(o_2, o_3, \ldots o_{T-1}|o_1, o_T)$. A naive design for goal-conditioned prediction based on forward auto-regressive models (GCP-sequential, shown in Fig 2, center) predicts latent state representations sequentially in chronological order, from the start to the end, with the prediction at each point in time conditioned on the first and final observations as well as the previous latent state. In the following we propose a better goal-conditioned predictors that is able to scale to very long sequences.

2.1. Goal-Conditioned Prediction by Recursive Infilling

In order to scale goal-conditioned prediction to longer time horizons we now design a tree-structured GCP model that is both more efficient and more effective than the naive sequential predictor.

Suppose that we have an intermediate state prediction operator $p(s_t|\text{pa}(t))$ that produces an intermediate latent state s_t halfway in time between its two parent states pa(t). Then, consider the following alternative process for goal-conditioned prediction depicted in Fig 2 (right): at the beginning, the observed first and last observation are encoded into the latent state space as s_1 and s_T, and the prediction operator $p(s_t|\text{pa}(t))$ generates $s_{T/2}$. The same operator may now be applied to two new sets of parents $(s_1, s_{T/2})$ and $(s_{T/2}, s_T)$. As this process continues recursively, the intermediate prediction operator fills in more and more temporal detail until the full sequence is synthesized.

We call this model GCP-tree, since it has a tree-like shape where each predicted state is dependent on its left and right parents, starting with the start and the goal. GCP-tree factorizes the goal-conditioned sequence generation as:

$$p(o_2, o_3, \ldots o_{T-1}|o_1, o_T) = \int p(s_1|o_1)p(s_T|o_T) \prod_{t=2}^{T-1} p(o_t|s_t)p(s_t|\text{pa}(t))ds_{2:T-1}. \quad (1)$$

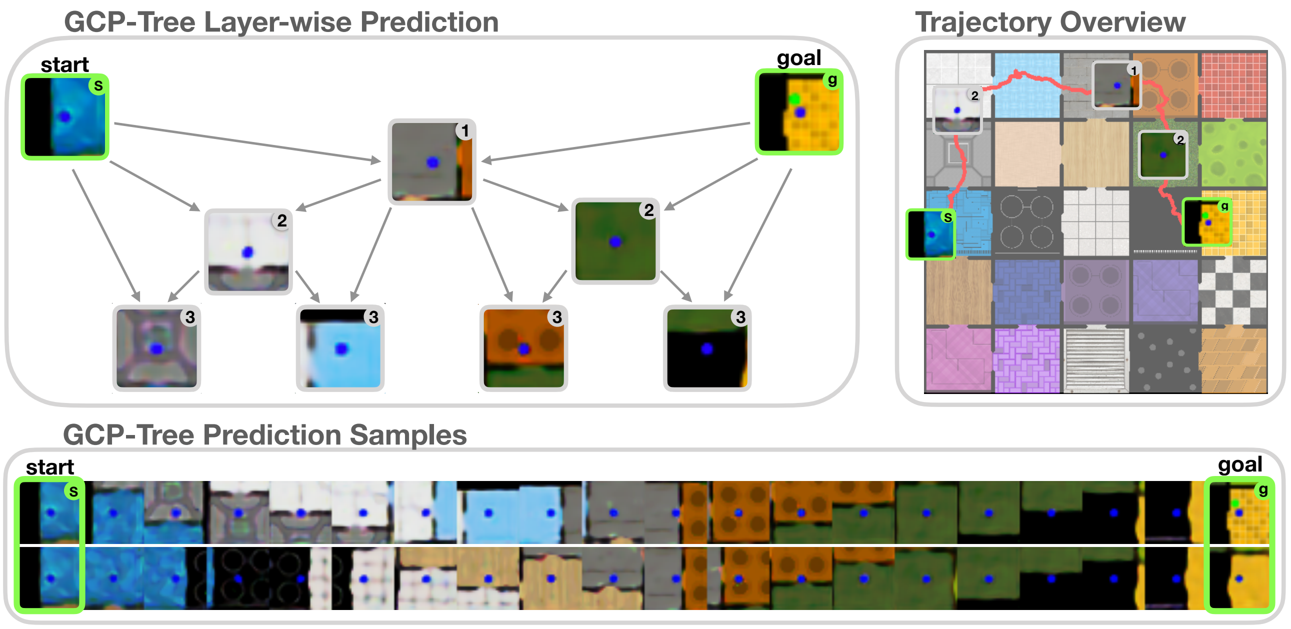

Figure 3. Samples from GCP-tree on the 25-room data. **Left**: hierarchical prediction process. At each layer, the infilling operator is applied between every two frames, producing a sequence with a finer and finer temporal resolution (three layers out of eight are shown). **Right**: visualization of the trajectory on the map together with a plan execution (see Section 4.2). **Bottom**: two image sequences sampled given the same start and goal (subsampled to 20 frames for visualization). Our model leverages stochastic latent states that enable modeling multimodal trajectories. See additional video results on the supplementary website `sites.google.com/view/video-gcp`.

2.2. Latent Variable Models for GCP

We have so far described the latent state s_t as being a monolithic random variable. However, an appropriate design of s_t is crucial for good performance: a purely deterministic s_t might not be able to model the variation in the data, while a purely stochastic s_t might lead to optimization challenges. Following prior work [3, 11], we therefore divide s_t into h_t and z_t, i.e. $s_t = (h_t, z_t)$, where h_t is the deterministic memory state of a recurrent neural network, and z_t is a stochastic per-time step latent variable. To optimize the resulting model, we leverage amortized variational inference [18, 26] with an approximate posterior $q(\tilde{z}|o_{1:T})$, where $\tilde{z} = z_{2:T-1}$. The deterministic state h_t does not require inference since it can simply be computed from the observed data o_1, o_T. The training objective is the following evidence lower bound on the log-likelihood of the sequence:

$$\begin{aligned}\ln p(o_{2:T-1}|o_{1,T}) \\ \geq \mathbb{E}_{q(z_{2:T-1}|x)}\left[\ln p(o_{2:T-1}|o_{1,T}, z_{2:T-1})\right] - \\ \beta KL\left[q(z|o_{1:T}) \,||\, p(z_{2:T-1}|o_{1,T})\right]. \qquad (2)\end{aligned}$$

3. Planning & Control with Goal-Conditioned Prediction

The GCP model can be directly applied to control problems since, given a goal, it can produce realistic trajectories for reaching that goal. However, in many cases our objective is to reach the goal *in a specific way*. For instance, we might want to spend the least amount of time or energy required to reach the goal. In those cases, explicit planning is required to obtain a trajectory from the model that optimizes a user-provided cost function $\mathcal{C}(o_t, \ldots, o_{t'})$. In GCPs, planning is performed over the latent variables z that determine *which* trajectory between start and goal is predicted: $\min_z \mathcal{C}(g(o_t, o_T, z))$, where g is the GCP model. We propose to use the cross-entropy method (CEM, [27]) for optimization, which has proven effective in prior work on visual MPC [7, 23, 24, 25]. We train a neural network estimator for the expected cost via supervised learning by randomly sampling two observations from a training trajectory and evaluating the true cost on the connecting trajectory segment $\mathcal{C}(o_t, \ldots, o_{t'})$ to obtain the target value. Once a trajectory is planned, we infer the actions necessary to execute it using a learned inverse model (see Appendix F).

Goal-conditioned hierarchical planning. Instead of optimizing the full trajectory at once, the hierarchical structure of the GCP-tree model allows us to design a more efficient, hierarchical planning scheme in which the trajectories between start and goal are optimized in a coarse-to-fine manner. The procedure is detailed in Algorithm 1. We initialize the plan to consist of only start and goal observation. Then our approach recursively adds new subgoals to the plan, leading to a more and more detailed trajectory. Concretely, we proceed by optimizing the latent variables of the

Table 1. Image-based control performance on navigation tasks

METHOD	9-ROOM NAV		25-ROOM NAV	
	SUCC.	COST	SUCC.	COST
GCBC	45%	139.75	7%	402.48
VF [7]	84%	128.00	26%	362.82
OURS	**93%**	**34.34**	**82%**	**158.06**
OURS (FLAT)	**94%**	36.00	79%	181.02
OURS (SEQUENTIAL)	91%	50.02	14%	391.99

GCP-tree model $g(o_t, o_T, z)$ layer by layer: in every step we sample M candidate latents per subgoal in the current layer and pick the corresponding subgoal that minimizes the total cost with respect to both its parents. The best subgoal gets inserted into the plan between its parents and the procedure recurses.

4. Experimental Evaluation

The aim of our experiments is to study the following questions: (1) Are the proposed visual goal-conditioned predictors able to effectively predict goal-directed long-horizon trajectories? (2) Is the proposed goal-conditioned hierarchical planning method able to solve long-horizon visual control tasks?

4.1. Goal-Conditioned Video Prediction

We compare the GCP models to prior interpolation methods in see Tab. 2. We observe that this prior work fails to learn meaningful long-term dynamics, and instead blend between start and goal image or predict physically implausible changes in the scene. In contrast, GCP-sequential and GCP-tree, equipped with powerful latent variable models, learn to predict rich scene dynamics between distant start and goal frames Furthmore, n the longer Human 3.6M and 25-room datasets, the GCP-tree model significantly outperforms the GCP-sequential model. Qualitatively, we observe that the sequential model struggles to take into account the goal information on the longer sequences, as this requires modeling long-term dependencies, while the hierarchical model is able to naturally incorporate the goal information in the recursive infilling process.

4.2. Visual Goal-Conditioned Planning and Control

Next, we evaluate our hierarchical goal-conditioned planning approach (see Section 3) on long-horizon visual control tasks. We test our method on a challenging image-based navigation task in the 9 and 25-rooms environments described in Section 4.1. Given the current image observation the agent is tasked to reach the goal, defined by a goal image, on the shortest possible path. We evaluate in both the 9-room and the 25-room layout with 100 task instances each. Successful task execution involves crossing up to three and up to 10 rooms respectively, requiring planning over horizons of several hundred time steps, much longer than in previous visual planning methods [6, 7].

We compare hierarchical planning with GCP to visual foresight (VF, Ebert et al. [7]), which optimizes rollouts from an action-conditioned forward prediction model via CEM[27]. We adopt the improvements to the sampling and CEM procedure introduced in Nagabandi et al. [23]. We also compare to goal-conditioned behavioral cloning (GCBC, [5]) as a “planning-free" approach for learning goal-reaching from example goal-reaching behavior.

In Table 1, we report the average success rate of reaching the goal room, as well as the average cost, which corresponds to the trajectory length.[1] VF performs well on the easy task set, which requires planning horizons similar to prior work on VF, but struggles on the longer tasks as the search space becomes large. The BC method is not able to model the complexity of the training data and fails to solve these environments. In contrast, our approach performs well even on the long-horizon task set.

We compare different planning approaches in Fig. 5. We find that samples from the forward prediction model in VF have low probability of reaching long-horizon goals. Using GCPs with a non-hierarchical planning scheme similar to [7, 23] (GCP-Flat) requires optimization over a large set of possible trajectories between start and goal and can struggle to find a plan with low cost. In contrast, our hierarchical planning approach finds plans with low cost by breaking the long-horizon task into shorter subtasks through multiple recursions of subgoal planning. Using GCP-sequential instead of GCP-tree for sampling performs well on short tasks, but struggles to scale to longer tasks (see Table 1), highlighting the importance of hierarchical prediction.

5. Discussion

We present two models for goal-conditioned prediction: a standard sequential architecture and a hierarchical tree-structured variant, where the latter either splits the sequence into equal parts at each level of the tree, or into variable-length chunks via an adaptive binding mechanism. We further propose an efficient hierarchical planning approach based on the tree-structure model. All variants of our method outperform prior video interpolation methods, and the hierarchical variants substantially outperform the sequential model and prior visual MPC approaches on a long-horizon image-based navigation task. Additionally, the adaptive binding model can discover bottleneck subgoals.

[1]Since reporting length for failed cases would skew the results towards methods that produce short, unsuccessful trajectories, we report a constant large length for failed trajectories.

References

[1] Lars Buesing, Theophane Weber, Sébastien Racanière, S. M. Ali Eslami, Danilo Jimenez Rezende, David P. Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, and Daan Wierstra. Learning and querying fast generative models for reinforcement learning. *arXiv:1802.03006*, 2018.

[2] Maxime Chevalier-Boisvert. gym-miniworld environment for openai gym. `https://github.com/maximecb/gym-miniworld`, 2018.

[3] E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *Proceedings of International Conference on Machine Learning (ICML)*, 2018.

[4] Emily Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, pages 4417–4426, 2017.

[5] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. In *NeurIPS*, 2019.

[6] Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *Conference on Robotic Learning (CoRL)*, 2017.

[7] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv:1812.00568*, 2018.

[8] Linxi Fan, Yuke Zhu, Jiren Zhu, Zihua Liu, Orien Zeng, Anchit Gupta, Joan Creus-Costa, Silvio Savarese, and Li Fei-Fei. Surreal: Open-source reinforcement learning framework and robot manipulation benchmark. In *Conference on Robot Learning*, 2018.

[9] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2016.

[10] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Proceedings of Neural Information Processing Systems (NeurIPS)*. 2018.

[11] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *Proceedings of International Conference on Machine Learning (ICML)*, 2019.

[12] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *ICLR 2020*, 2020.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[15] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[16] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4): 566–580, 1996.

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[18] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014.

[19] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In *Advances in Neural Information Processing Systems*, pages 8733–8744, 2018.

[20] A. X. Lee, R. Zhang, F. Ebert, P. Abbeel, C. Finn, and S. Levine. Stochastic adversarial video prediction. *arXiv:1804.01523*, abs/1804.01523, 2018.

[21] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[22] Ziwei Liu, Raymond A Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4463–4471, 2017.

[23] Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. *Conference on Robot Learning (CoRL)*, 2019.

[24] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *NeurIPS*, 2019.

[25] Karl Pertsch, Oleh Rybkin, Jingyun Yang, Shenghao Zhou, Kosta Derpanis, Joseph Lim, Kostas Daniilidis, and Andrew Jaegle. Keyin: Keyframing for visual planning. *Conference on Learning for Dynamics and Control*, 2020.

[26] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014.

[27] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer-Verlag New York, 2004.

[28] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[29] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[30] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[31] Nevan Wichers, Ruben Villegas, Dumitru Erhan, and Honglak Lee. Hierarchical long-term video prediction without supervision. *ICML*, 2018.

[32] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[33] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *Robotics: Science and Systems (RSS)*, 2019.

[34] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J Johnson, and Sergey Levine. Solar: deep structured representations for model-based reinforcement learning. *ICLR 2019*, 2019.

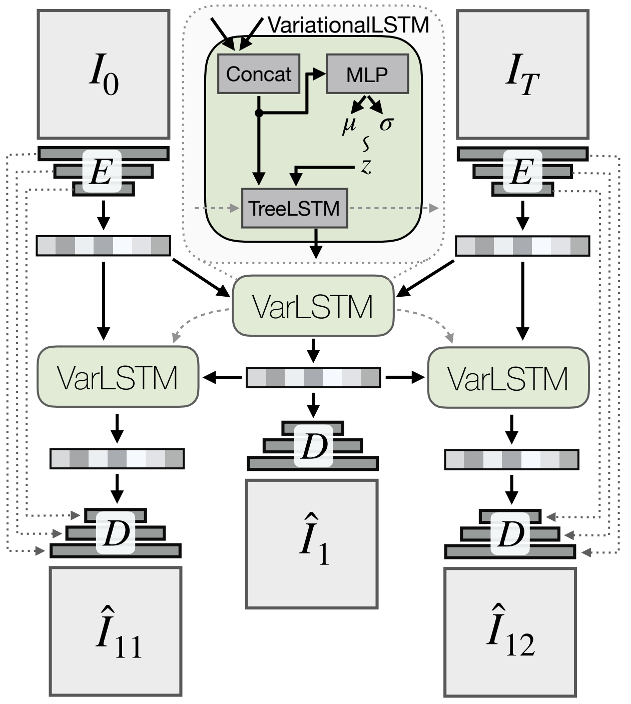

Figure 4. Architecture for two-layer hierarchical goal-conditioned predictor (GCP). Skip connections to first node's decoder omitted for clarity.

A. Architecture

We describe how GCP models can be instantiated with deep neural networks to predict sequences of high-dimensional observations $o_{1:T}$, such as videos. The prior $p(z_t|\text{pa}(t))$ is a diagonal Gaussian whose parameters are predicted with a multi-layer perceptron (MLP). The deterministic state predictor $p(h_t|z_t, \text{pa}(t))$ is implemented as an LSTM [13]. We found that using TreeLSTM [28] as the backbone of the hierarchical predictor significantly improved performance over vanilla recurrent architectures. We condition the recurrent predictor on the start and goal observations encoded through a convolutional encoder $e_t = E(o_t)$. The decoding distribution $p(o_t|s_t)$ is predicted by a convolutional decoder with input features $\hat{e}_t$ and skip-connections from the encoder [30, 3]. The parameters of the diagonal Gaussian posterior distribution for each node, $q(z_t|o_t, \text{pa}(t))$, are predicted given the corresponding observation and parent nodes with another MLP.

We use a convolutional encoder and decoder similar to the standard DCGAN discriminator and generator architecture respectively. The latent variables z_n as well as e_n are 32-dimensional. All hidden layers in the Multi-Layer Perceptron have 32 neurons. We add skip-connections from the encoder activations from the first image to the decoder for all images. For the inference network implemented with attention, we found it beneficial to use a 2-layer 1D temporal convolutional network that adds temporal context into the latent vectors e_t before attention. For the recursive predictor that predicts e_n, we found it crucial for the stability of the training to activate e_n with hyperbolic tangent (tanh), and use group normalization [32]. We observed that without this, the magnitude of activations can explode in the lower levels of the tree and conjecture that this is due to recursive application of the same network. We found that batch normalization [14] does not work as well as group normalization for the recursive predictor and conjecture that this is due to the activation distributions being non-i.i.d. for different levels of the tree. We use batch normalization in the convolutional encoder and decoder, and use local per-image batch statistics at test time. Fig. 4 gives a schematic overview of the recursive prediction architecture.

Hyperparameters. For each method and dataset, we performed a manual sweep of the hyperparameter β in the range from $1\text{e}{-0}$ to $1\text{e}{-4}$. The convolutional encoder and decoder both have five layers. We use the Rectified Adam optimizer [21, 17] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$, batch size of 16 for GCP-sequential and 4 for GCP-tree, and a learning rate of $2\text{e}{-4}$. On each dataset, we trained each network for the same number of epochs on a single high-end NVIDIA GPU.

B. Goal-conditioned planning and control

Algorithm 1 Goal-Conditioned Hierarchical Planning

```
1: Inputs: Hierarchical goal-conditioned predictor $g$, current & goal observation $o_t, o_T$, cost function $\hat{\mathcal{C}}$
2: Initialize plan: $P = [o_t, o_T]$
3: for $d = 1...D$ do                                   ▷ iterate depth of hierarchy
4:     for $n = 0...|P| - 1$ do
5:         $\mathbf{z} \sim \mathcal{N}(0, I)$                  ▷ sample M subgoal latents
6:         $\mathbf{o}_{\text{sg}} = g(P[n], P[n+1], \mathbf{z})$        ▷ predict subgoals
7:         $o_{d,n} = \arg\min_{o \in \mathbf{o}_{\text{sg}}} \hat{\mathcal{C}}(P[n], o) + \hat{\mathcal{C}}(o, P[n+1])$
8:         INSERT$(P, o_{d,n})$                       ▷ insert best subgoal in plan
9:     end for
10: end for
11: return $P$
```

Algorithm 2 Goal-Conditioned Control

```
1: Inputs: Goal-conditioned predictor $g$, inverse model $f_{\text{inv}}(o_t, o_{t+1})$, goal observation $o_T$, cost function $\mathcal{C}$, planning routine PLAN$(\cdot)$
2: while not done do
3:     $\hat{o}_t ... \hat{o}_T = \text{PLAN}(g, o_t, o_T, \mathcal{C})$
4:     for $i = 1...n_{\text{replan}}$ do
5:         $a_t = f_{\text{inv}}(o_t, \hat{o}_{t+i})$
6:         Execute action $a_t$.
7:     end for
8: end while
```

C. Additional results

We include additional qualitative and quantitative results here as well as at the supplementary website: `sites.google.com/view/video-gcp`.

D. Experimental Setup

Most commonly used video datasets in the literature depict relatively short motions, making them poorly suited for studying long-horizon prediction capability. We therefore evaluate on one standard dataset, and two synthetic datasets that we designed specifically for evaluating long-horizon prediction. The pick&place dataset contains videos of a simulated Sawyer robot arm placing objects into a bin. Training trajectories contain up to 80 frames at 64×64 px and are collected using a simple rule-based policy. The *Navigation* data consists of videos of an agent navigating a simulated environment with multiple rooms: we evaluate versions with 9-room and 25-room layouts, both of which use 32×32 px agent-centric topdown image observations, with up to 100 and 200 frame sequences, respectively. We collect example trajectories that reach goals in a randomized, suboptimal manner, providing a very diverse set of trajectories (details are in App. E). We further evaluate on the real-world Human 3.6M video dataset [15], predicting 64×64 px frames at full frequency of 50Hz up to 10 seconds in the future to show the scalability of our method. This is in contrast to prior work which evaluated on subsampled sequences shorter than 100 frames (see [4, 3, 31]). Architecture and hyperparameters are detailed in Appendix, Section A.

E. Data processing and generation

For training GCPs we use a dataset of example agent goal-reaching behavior. Below we describe how we collect those examples on the pick&place and navigation tasks and the details of the Human3.6M dataset.

pick&place. We generate the pick&place dataset using the RoboSuite framework [8] that is based on the Mujoco physics simulator [29]. We generate example goal-reaching trajectories by placing two objects at random locations on the table and using a rule-based policy to move them into the box that is located at a fixed position on the right of the workspace. We sample the object type randomly from a set of two possible object types, bread and can, with replacement.

Human 3.6M. For the Human 3.6 dataset, we downsample the original videos to 64 by 64 resolution. We obtain videos of length of roughly 800 to 1600 frames, which we randomly crop in time to 500-frame sequences. We split the Human 3.6 into training, validation and test set by correspondingly 95%, 5% and 5% of the data.

Navigation. For the navigation task the agent is asked to plan and execute a path between a given 2D start and goal position. The environment is simulated using the Gym-Miniworld framework [2]. We collect goal-reaching examples by randomly sampling start and goal positions in the 2D maze and plan trajectories using the Probabilistic Roadmap (PRM, Kavraki et al. [16]) planner. The navigation problem is designed such that multiple possible room sequences can be traversed to reach from start to goal for any start and goal combination. During planning we sample one possible room sequence at random, but constrain the selection to only such sequences that do not visit any room more than once, i.e. that do not have loops. This together with the random sampling of waypoints of the PRM algorithm leads to collected examples of goal reaching behavior with substantial suboptimality. We show an example trajectory distribution from the data in Fig. 10. While GCPs support training on sequences of variable length we need to set an upper bound on the length of trajectories to bound the required depth of the hierarchical predictive model and allow for efficient batch computation (e.g. at most 200 frames for the 25-room environment). If plans from the PRM planner exceed this threshold we subsample them to the maximum lenght using spline interpolation before executing them in the environment. The training data consists of 10,000 and 23,700 sequences for the 9-room and the 25-room task respectively, which we split at a ration of 99%, 1%, 1% into training, validation and test.

F. Planning Experimental Setup

For planning with GCPs we use the model architectures described in Section A trained on the navigation data described in Section E. The hyperparameters for the hierarchical planning experiments are listed in Table 3. We keep the hyperparameters constant across both 9-room and 25-room tasks except for the maximum episode length which we increase to 400 steps for the 25-room task. Note that the cost function is only used at training time to train the cost estimator described in Section 3, which we use to estimate all costs during planning.

To infer the actions necessary to execute a given plan, we train a separate inverse model $a_t = f_{\text{inv}}(o_t, o_{t+1})$ that infers the action a_t which leads from observation o_t to o_{t+1}. We train the inverse model with action labels from the training dataset and, in practice, input predicted feature vectors $\hat{e}_t$ instead of the decoded observations to not be affected by potential inaccuracies in the decoding process. We use a simple 3-layer MLP with 128 hidden units in each layer to instantiate f_{inv}. At every time step the current observation along with the next observation from the plan is passed

Table 2. Long-term prediction performance of the goal-conditioned predictors compared to prior work on video interpolation.

DATASET	PICK&PLACE		HUMAN 3.6M		9 ROOMS NAV		25 ROOMS NAV	
METHOD	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
GCP-TREE	**34.34**	**0.965**	**28.34**	**0.928**	**13.83**	**0.288**	**12.88**	**0.279**
GCP-SEQUENTIAL	**34.45**	**0.965**	27.57	0.924	12.91	0.213	11.61	0.209
DVF [22]	26.15	0.858	26.74	0.922	11.678	0.22	11.34	0.172
CIGAN [19]	21.16	0.613	16.89	0.453	11.96	0.222	9.91	0.150

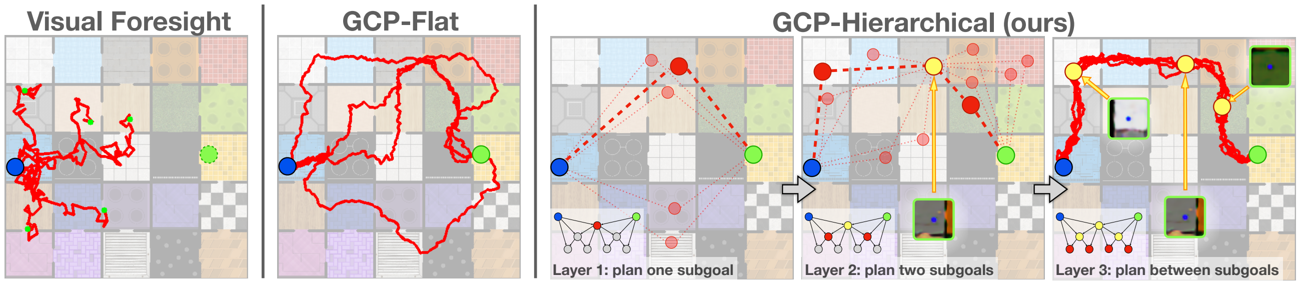
Visual Foresight
GCP-Flat
GCP-Hierarchical (ours)
Layer 1: plan one subgoal
Layer 2: plan two subgoals
Layer 3: plan between subgoals

Figure 5. Comparison between planning methods. Trajectories (red) sampled while planning from start (blue) to goal (green). All methods predict image trajectories, which are shown as 2d states for visualization. **Left**: visual MPC [7] with forward predictor, **middle**: non-hierarchical planning with goal-conditioned predictor (GCP), **right**: hierarchical planning with GCP (ours) recursively optimizes subgoals (yellow/red) in a coarse-to-fine manner and finally plans short trajectories between the subgoals. Goal-conditioning ensures that trajectories reach the long-horizon goal, while hierarchical planning decomposes the task into shorter segments which are easier to optimize.

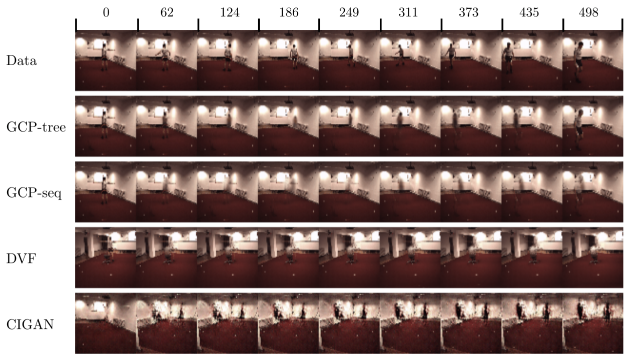

Figure 6. Predictions on Human 3.6M. We see that the GCP models are able to faithfully capture the human trajectory. The optical flow-based method (DVF) captures the background but fails to generate complex motion needed for long-term goal-conditioned prediction. Causal InfoGan also struggles to capture the structure of these long sequences and produce implausible interpolations. Full qualitative results are on the supplementary website: `sites.google.com/view/gcp-hier/home`.

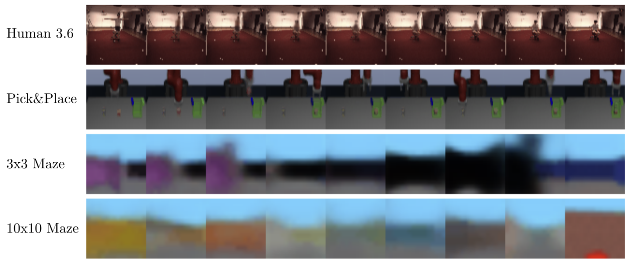

Figure 7. Prior samples from GCP-tree on the four datasets: Human 3.6, pick&place, 3x3 Maze and 10x10 Maze. Each sequence is subsampled to 9 frames. Full qualitative results are on the supplementary website: `sites.google.com/view/gcp-hier/home`.

to the inverse model and the predicted action is executed. We found it crucial to perform such closed-loop control to avoid accumulating errors that posed a central problem when inferring the actions for the whole plan once and then executing them open-loop.

We separately tuned the hyperparameters for the visual foresight baseline and found that substantially more samples are required to achieve good performance, even on the shorter 9-room tasks. Specifically, we perform three iterations of CEM with a batch size of 500 samples each. For sampling and refitting of action distributions we follow the procedure described in [23]. We use a planning horizon of 50 steps and replan after the current plan is executed. We cannot use the cost function from Table 3 for this baseline as it leads to degenerate solutions: in constrast to GCPs, VF searches over the space of *all* trajectories, not only those that reach the goal. Therefore, the VF planner could minimize the trajectory length cost used for the GCP models by predicting trajectories in which the agent does not move. We instead use a cost function that measures whether the predicted trajectory reached the goal by computing the L2 distance between the final predicted observation of the trajectory and

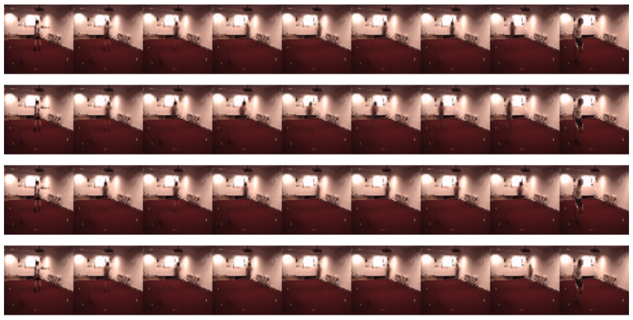

Figure 8. Prior samples from GCP-tree on the Human 3.6M dataset. Each row is a different prior sample conditioned on the same information.

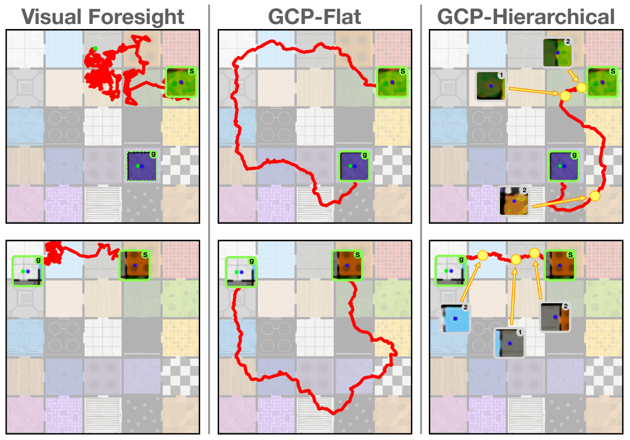

Figure 9. Comparison of visual planning & control approaches. Execution traces of Visual Foresight (**left**), GCP-tree with non-hierarchical planning (**middle**) and GCP-tree with hierarchical planning (**right**) on two 25-room navigation tasks. Visualized are start and goal observation for all approaches as well as predicted subgoals for hierarchical planning. Both GCP-based approaches can reach faraway goals reliably, but GCP with hierarchical planning finds shorter trajectories to the goal.

the goal observation.

We run all experiments on a single NVIDIA V100 GPU and find that we need approximately 30mins / 1h to evaluate all 100 task instances on the 9-room and 25-room tasks respectively when using the hierarchical GCP planning. The VF evaluation requires many more model rollouts and therefore increases the runtime by a factor of approximately five, even though we increase the model rollout batch size by a factor of 20 for VF to parallelize trajectory sampling as much as possible.

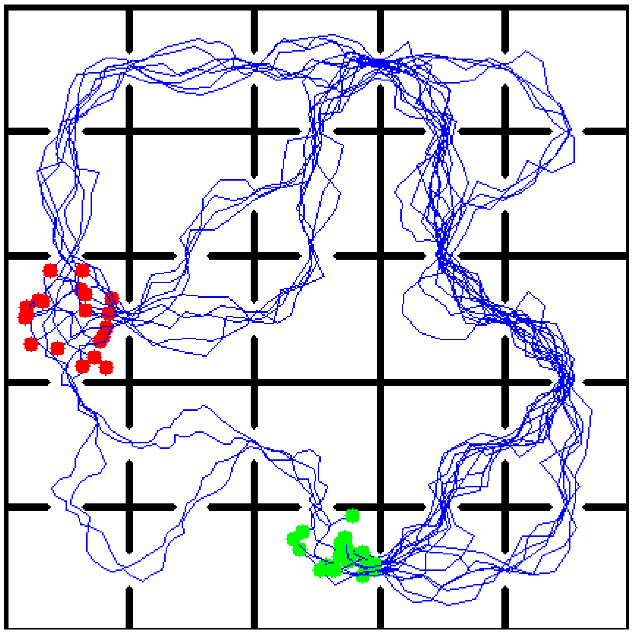

Figure 10. Example trajectory distributions between fixed start (red) and goal (green) rooms on the 25-room navigation task. The example goal-reaching behavior is highly suboptimal, with both strong multimodality in the space of possible solutions as well as low-level noise in each individual trajectory.

Table 3. Hyperparameters for hierarchical planning with GCPs on 9-room and 25-room navigation tasks.

Hierarchical Planning Parameters	
Hierarchical planning layers (D)	2
Samples per subgoal (M)	10
Final Segment Optimization	
Sequence samples per Segment	5
General Parameters	
Max. episode steps	200 / 400
Cost function	$\sum_{t=0}^{T-1}(x_{t+1} - x_t)^2$