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]
[1] USC [2] UPenn [3] UC Berkeley [4] Stanford University

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 optimizes trajectories in a coarse-to-fine manner. We show that 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 [11, 69, 18, 67, 41, 20] employ powerful predictive models [15, 4, 19, 36] to enable agents to predict and plan in complex environments directly from visual sensory observations, without needing to engineer a state estimator. To plan a sequence of actions, these approaches usually use the predictive model to generate candidate roll-outs 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

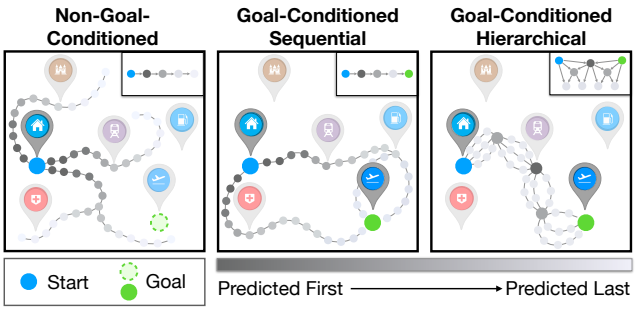

Figure 1: When planning towards faraway goals, we propose to condition the prediction of candidate trajectories on the goal, which significantly reduces the search space of possible trajectories (**left** vs. **middle**) and enables hierarchical planning approaches that break a long-horizon task into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

[*] Equal contribution. Ordering determined by a coin flip. Project page: `orybkin.github.io/video-gcp`

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

approaches do not scale to complex long-horizon tasks [11]. 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 search space for finding the best route, 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 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.

Hierarchical prediction models naturally lend themselves to modeling the hierarchical structure present in many long-horizon tasks by breaking them into their constituent steps. However such procedural steps do not all occur on a regularly spaced schedule or last for equal lengths of time. Therefore, we further propose a version of our model based on a novel probabilistic formulation of dynamic time warping [55] that allows the model to select which frames to generate at each level in the tree, enabling flexible placement of intermediate predictions.

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 Related Work

Video interpolation. We propose a latent variable goal-conditioned prediction model that is able to handle high-dimensional image observations. This resembles prior work on video-interpolation where given a start and a goal image, images are filled in between. So far such work has focused on short-term interpolation, often using models based on optical flow [39, 26, 45, 46]. Recent work has proposed neural network models that predict images directly, but this work still evaluates on short-horizon prediction [64]. The models introduced in our work by contrast scale to video sequences of up to 500 time steps modelling complex distributions that exhibit multi-modality.

Visual planning and control. Most existing visual planning methods [14, 49, 67, 11, 19] use model predictive control, computing plans forward in time by sampling state or action sequences. This quickly becomes computationally intractable for longer horizon problems, as the search complexity grows exponentially with the number of time steps [11]. Instead, we propose a method for goal-conditioned hierarchical planning, which is able to effectively scale to long horizons as it both reduces the search space and performs more efficient hierarchical optimization. Ichter and Pavone [22] also perform goal-conditioned planning by constraining the search space to trajectories that end at the goal, however, the method is only validated on low-dimensional states. In this paper, we leverage latent state-space goal-conditioned predictors that scale to visual inputs and further improve the planning by using a hierarchical divide-and-conquer scheme. Other types of goal-conditioned control include inverse models and goal-conditioned imitative models [48, 68, 60, 57]. However, these methods rely

on imitation learning and are limited to settings where high-quality demonstrations are available. In contrast, our goal-conditioned planning and control method is able to optimize the trajectory it executes, and does not require optimal training data.

Hierarchical planning. While hierarchical planning has been extensively explored in symbolic AI [54, 34, 28], these approaches are unable to cope with raw (e.g., image-based) observations, limiting their ability to solve diverse real-world tasks. Instead, we propose an approach that learns to perform hierarchical planning directly in terms of sensory observations, purely from data. Since our method does not require human-designed specification of tasks and environment, it is applicable in general settings where trajectory data can be collected. Recently, a number of different hierarchical planning approaches have been introduced [25, 50, 13, 43, 30, 44] that only work well with one or two layers of abstraction due to the architectural design or computational bottlenecks. Some of the few hierarchical planning approaches that have been shown to work with more than two layers of abstraction use tree-structured models [5, 27, 47]. However these models have not been shown to scale to high-dimensional spaces such as images. While also using a tree-structured model similar to our method, Chen et al. [5] make the assumption that the map in the physical workspace is known. To the best of our knowledge the hierarchical planning algorithm introduced here is the first algorithm that uses a variable number of abstraction layers while scaling to high-dimensional inputs such as images.

3 Goal-Conditioned Prediction

In this section, we formalize the goal-condition prediction problem, and propose several 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 $p(o_{k+1}, o_{k+2}, \dots o_{T-1}|o_1, o_2, \dots o_k)$. 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 $p(o_2, o_3, \dots o_{T-1}|o_1, o_T)$. We propose several designs for goal-conditioned predictors that operate in learned compact state spaces for scalability and accuracy.

3.1 Goal-Conditioned Sequential Prediction

We first present a naive design for goal-conditioned prediction based on forward auto-regressive models. Autoregressive models operating directly on observations scale poorly in terms of computational efficiency and predictive performance [8, 4, 19]. We design a latent state-space model (GCP-sequential, shown in Fig 2, center) that predicts in a latent space represented by a random variable s_t and then decodes the observations with a decoder $p(o_t|s_t)$. The latent state s_t is used to allow handling partially observable settings. The likelihood of this model factorizes as follows:

$$p(o_2, o_3, \dots o_{T-1}|o_1, o_T) = \int p(o_2|s_2)p(s_2|o_1, o_T) \prod_{t=3}^{T-1} p(o_t|s_t)p(s_t|s_{t-1}, o_1, o_T)ds_{2:T-1}. \quad (1)$$

We show in Sec 3.4 that this model is simple to implement, and can build directly on previously proposed auto-regressive sequence prediction models. However, its computational complexity scales with the sequence length, as every state must be produced in sequence. As we show empirically, this approach also struggles with modeling longer sequences due to compounding errors, and is prone to ignoring the goal information on these longer sequences as very long-term dependencies have to be modeled when predicting the second observation from the first observation and the goal.

3.2 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 $\text{pa}(t)$. 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

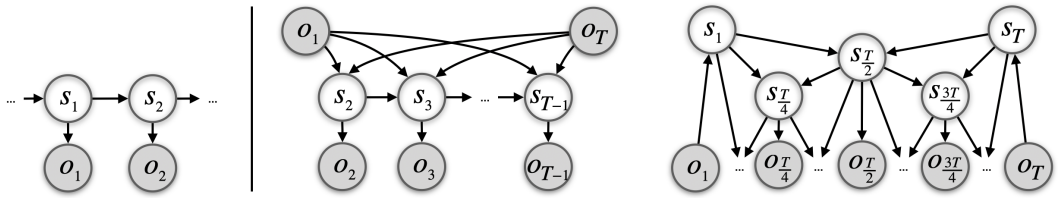

Figure 2: Graphical models for state-space sequence generation: forward prediction (left) and the proposed goal-conditioned predictors (GCPs). Shaded circles denote observations, white circles denote unobserved latent states. Center: a sequential goal-conditioned predictor with structure similar to forward 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.

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[1] 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 problem as:

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

Adaptive binding. We have thus far described the intermediate prediction operator as always generating the state that occurs halfway in time between its two parents. While this is a simple and effective scheme, it may not correspond to the natural hierarchical structure in the sequence. For example, in the navigation example from the introduction, we might prefer the first split to correspond to visiting the bank, which partitions the prediction problem into two largely independent halves. We then design a version of GCP-tree that allows the intermediate state predictor to select which of the several states between the parents to predict, each time it is applied. In other words, the predicted state might *bind* to one of many observations in the sequence. In this more versatile model, we represent the time steps of the tree nodes with discrete latent variable w that selects which nodes bind to which observations: $p(o_t|s_{1:N}, w_t) = p(o_t|s_{w_t})$. We can then express the prediction problem as:

$$p(o_{2:T-1}|o_1, o_T) = \int p(s_1|o_1)p(s_N|o_T) \prod_n p(s_n|\text{pa}(n)) \prod_{t=2}^{T-1} p(o_t|s_{1:N}, w_t)p(w_t)ds_{1:N}dw_{2:T-1}.$$

Appendix F shows an efficient inference procedure for w based on a probabilistic version of dynamic time warping [55].

3.3 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 [8, 19], 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 [32, 51] 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:

$$\ln p(o_{2:T-1}|o_{1,T}) \geq \mathbb{E}_{q(\tilde{z})} \left[\ln p(o_{2:T-1}|o_{1,T}, \tilde{z})\right] - \text{KL}\left(q(\tilde{z}) \,||\, p(\tilde{z}|o_{1,T})\right). \tag{3}$$

[1]The structure of the generation graph closely mimics a graph-theoretic tree, but every node has two parents instead of one.

3.4 Architectures for Goal-Conditioned Prediction

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 [21]. 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 [62, 8]. In line with recent work [53], we found that learning a calibrated decoder is important for good performance, and we use the discrete logistics mixture as the decoding distribution [56]. 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. For a more detailed description of the architectural parameters we refer to Appendix C.

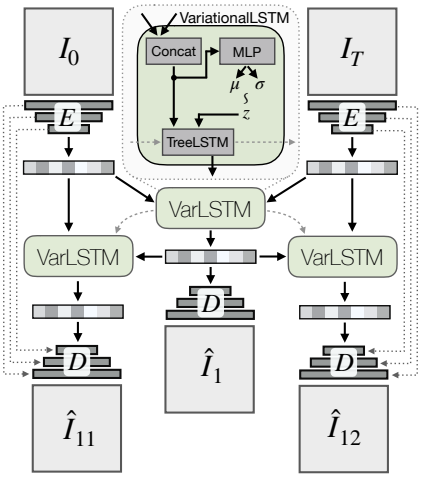

Figure 3: Architecture for two-layer hierarchical goal-conditioned predictor (GCP). Skip connections to first node's decoder omitted for clarity.

4 Planning & Control with Goal-Conditioned Prediction

In the previous section, we described an approach to goal-conditioned sequence prediction or GCP. 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 $\mathcal{C}(o_t, \dots, o_{t'})$. 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, [52]) for optimization, which has proven effective in prior work on visual MPC [11, 41, 44, 50]. Once a trajectory is planned, we infer the actions necessary to execute it using a learned inverse model (see Appendix, Section E).

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 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.

Algorithm 1 Goal-Conditioned Hierarchical Planning

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

Cost function. Evaluating the true cost function $\mathcal{C}(o_t, \dots, o_{t'})$ would require unrolling the full prediction tree. For more efficient, *hierarchical* planning, we instead want to evaluate the *expected* cost $\hat{\mathcal{C}}(o_t, o_{t'}) = \mathbb{E}_{(o_t,\dots,o_{t'}) \sim \mathcal{D}}\ \mathcal{C}(o_t, \dots, o_{t'})$ of a trajectory between two observations under the training data distribution $\mathcal{D}$. This allows us to estimate the cost of a trajectory passing through a given subgoal as the sum of the pairwise cost estimates to both its parents, without predicting all its children. 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 $\mathcal{C}(o_t, \dots, o_{t'})$ to obtain the target value.

Table 1: Long-term prediction performance of the goal-conditioned predictors compared to prior work on video interpolation. Additional evaluation on FVD / LPIPS [61, 70] in Appendix, Table 5.

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 [39]	26.15	0.858	26.74	0.922	11.678	0.22	11.34	0.172
CIGAN [35]	21.16	0.613	16.89	0.453	11.96	0.222	9.91	0.150

5 Experimental Evaluation

The aim of our experiments is to study the following questions: (1) Are the proposed GCPs able to effectively predict goal-directed trajectories in the image space and scale to long time horizons? (2) Is the proposed goal-conditioned hierarchical planning method able to solve long-horizon visual control tasks? (3) Does the version of GCP with adaptive binding find high-level events in the trajectories?

5.1 Goal-Conditioned Video Prediction

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 top-down 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. D)[2]. We further evaluate on the real-world Human 3.6M video dataset [24], 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 [9, 8, 65]). Architecture and hyperparameters are detailed in Appendix C.

Table 2: Ablation of prediction performance on pick&place

METHOD	PSNR	SSIM
TREE	34.34	0.965
TREE W/O SKIPS	32.64	0.955
TREE W/O LSTM	31.44	0.947

In Tab. 1, we compare the GCP models to a state-of-the-art deep video interpolation method, DVF [39],[5] as well as a method for goal-conditioned generation of visual plans by interpolation in a learned latent space, CIGAN [35]. Following the standard procedure for evaluation of stochastic prediction models, we report top-of-100 peak signal-to-noise ratio (PSNR) and structural similarity metric (SSIM). We observe that the interpolation methods fail 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 (see qualitative results in Fig. 4 and for all methods on the project website[6]).

Table 3: GCP runtime on 16×16 px H3.6M sequences in sec/training batch[4]

SEQ LENGTH	100	500	1000
GCP-SEQ	1.49	8.44	17.6
GCP-TREE	**0.55**	**1.66**	**2.77**
SPEED-UP	$\times 2.7$	$\times 5.1$	$\times 6.4$

On 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

[2] Our model can even learn to plan using data collected with completely random actions (see supp. section G)

[4] We use 16×16 px to fit the longest sequences on a single NVIDIA V100 GPU, we expect the results to translate to larger resolutions on GPUs with larger memory.

[5] While DVF has an official trained model, we re-train DVF on each dataset for better performance.

[6] See additional video results on the supplementary website `orybkin.github.io/video-gcp`

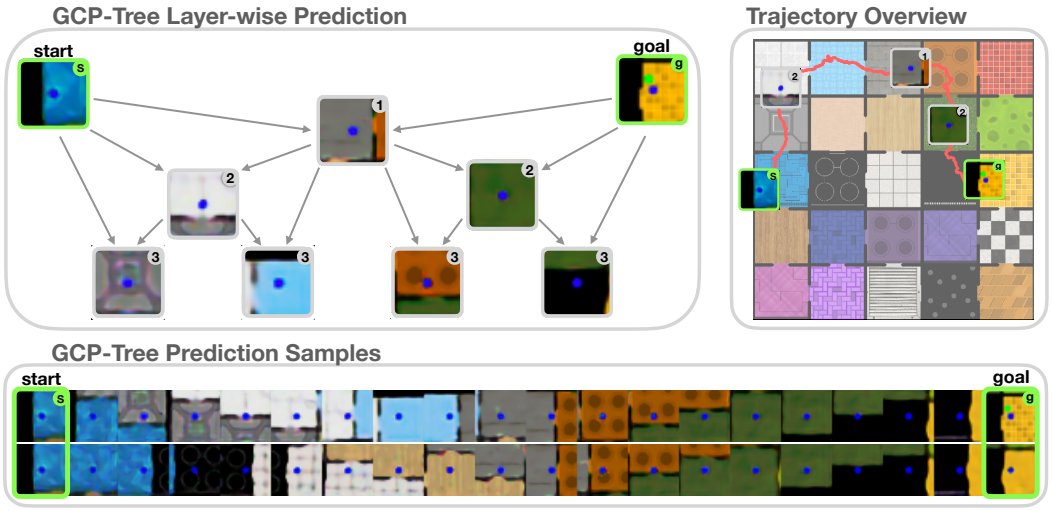

Figure 4: Samples from GCP-tree on the 25-room data. **Left**: hierarchical prediction process. At each layer, the infilling operator is applied between every two frames, producing a sequence with a finer and finer temporal resolution. Three layers out of eight are shown. **Right**: visualization of the trajectory on the map together with a plan execution (see Section 5.2). **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.

dependencies, while the hierarchical model is able to naturally incorporate the goal information in the recursive infilling process. Additionally, the hierarchical structure of GCP-tree enables substantially faster runtimes (see Table 3). We present an ablation study for GCP-tree in Tab. 2, showing that both the skip connections and the recurrence in the predictive module contribute to good performance.

5.2 Visual Goal-Conditioned Planning and Control

Next, we evaluate our hierarchical goal-conditioned planning approach (see Section 4) 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 5.1. 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 [10, 11].

Table 4: Visual control performance on navigation tasks

METHOD	9-ROOM NAV		25-ROOM NAV	
	SUCCESS	COST	SUCCESS	COST
GC BC [42]	45%	139.75	7%	402.48
VF [11]	84%	128.00	26%	362.82
OURS	**93%**	**34.34**	**82%**	**158.06**
GCP-FLAT	**94%**	36.00	79%	181.02
GCP-SEQUENTIAL	91%	50.02	14%	391.99

We compare hierarchical planning with GCP to visual foresight (VF, Ebert et al. [11]), which optimizes rollouts from an action-conditioned forward prediction model via CEM [52]. We adopt the improvements to the sampling and CEM procedure introduced in Nagabandi et al. [41]. We also compare to goal-conditioned behavioral cloning (GC BC, [42]) as a "planning-free" approach for learning goal-reaching from example goal-reaching behavior.

In Table 4, we report the average success rate of reaching the goal room, as well as the average cost, which corresponds to the trajectory length.[7] 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.

[7] 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.

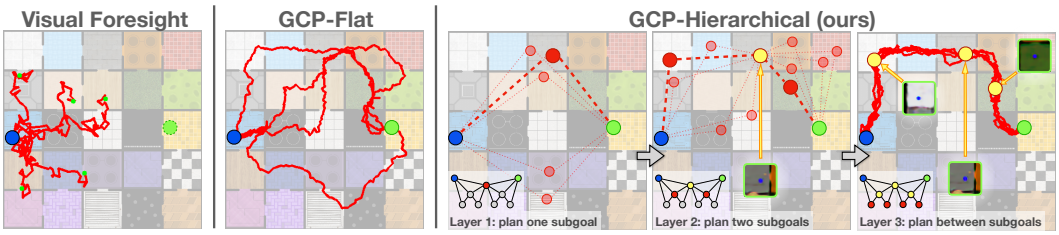

Figure 5: Comparison between planning methods. Trajectories (red) sampled while planning from start (blue) to goal (green). All methods predict image trajectories, which are shown as 2d states for visualization. **Left**: visual MPC [11] with forward predictor, **middle**: non-hierarchical planning with 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.

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 [11, 41] (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 4), highlighting the importance of the hierarchical prediction model.

5.3 Bottleneck Discovery

We qualitatively evaluate the ability of GCP-tree *with adaptive binding* (see Section 3.2) to learn the bottleneck structure in the robotic pick&place dataset. We increase the reconstruction loss of the nodes in the first two layers of the tree 50 times, forcing these nodes to bind to the frames for which the prediction is the most confident, the bottlenecks (see experimental details in Appendix F).

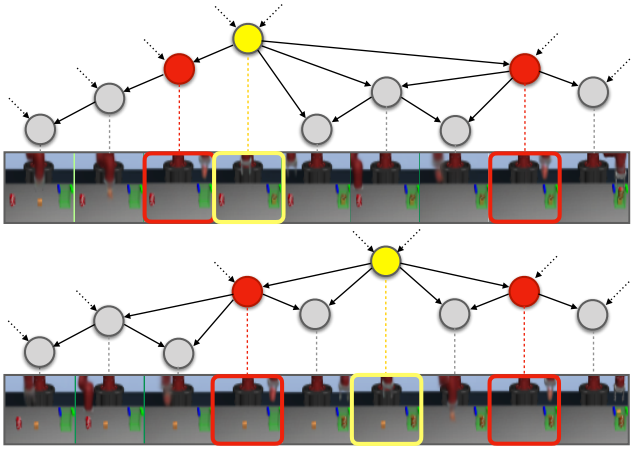

Figure 6: Bottleneck discovery on pick&place. Discovered tree structure with adaptive binding: nodes from the first two layers (yellow/red) bind to semantically consistent bottlenecks across sequences, e.g. in which the robot is about to drop the object into the bin.

In Fig. 6, we see that this structural prior causes the model to bind the top nodes to frames that represent semantic bottlenecks, e.g. when the robot is about to drop the object in the bin. We found that all three top layer nodes specialize on binding to distinctive bottlenecks, leading to diverse predicted tree structures. We did not observe that adaptive binding improves the quality of predictions on our datasets, though the ability to discover meaningful bottlenecks may itself be useful [3, 33, 16].

6 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.

Broader Impact

We proposed a method for visual prediction and planning that is able to solve long-horizon tasks autonomously. This method may have a broader impact on capabilities of robots performing tasks such as autonomous navigation or object manipulation, and may be applicable in settings such as navigation of zones dangerous for humans, search and rescue, as well as warehouse robotics applications. While the method, and in general all planning and reinforcement learning methods, may be applied to a variety of settings, including those with questionable ethical motivation, we are optimistic of the general positive impact of future autonomous robotic systems, especially in the areas described above.

Another ethical consideration is that, since the model is able to produce long videos targeted to a particular goal, it might be used to produce fake videos of people performing a certain action, and provides a degree of control about that action through the specification of the goal image. This might enable forging fake videos targeted at specific persons. However, recent research has shown that most current methods for generating fake videos are easily detectable, both by people and automatic detection methods [17, 1, 37].

Acknowledgements

We thank Suraj Nair, Thanard Kurutach, and Aviral Kumar for fruitful discussions. We would like to thank Ben Eysenbach, Ayush Jain and two anonymous internal reviewers for feedback on an earlier version of the paper, Shenghao Zhou for discussion and help with preliminary evaluation of the method, and Kristian Hartikainen for discussion and software development tips. This research was supported through the following grants: NSF-IIP-1439681 (I/UCRC), NSF-IIS-1703319, NSF MRI 1626008, ARL RCTA W911NF-10-2-0016, ONR N00014-17-1-2093, ARL DCIST CRA W911NF-17-2-0181, the DARPA-SRC C-BRIC, and by Honda Research Institute. KP and OR were visitors at UC Berkeley while conducting this research.

References

[1] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 38–45, 2019.

[2] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *Proceedings of International Conference on Learning Representations (ICLR)*, 2015.

[3] Andrew G. Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(1-2):41–77, January 2003. ISSN 0924-6703. doi: 10.1023/A:1022140919877. URL `https://doi.org/10.1023/A:1022140919877`.

[4] 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.

[5] Binghong Chen, Bo Dai, and Le Song. Learning to plan via neural exploration-exploitation trees. *CoRR*, abs/1903.00070, 2019. URL `http://arxiv.org/abs/1903.00070`.

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

[7] Marco Cuturi and Mathieu Blondel. Soft-dtw: a differentiable loss function for time-series. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 894–903. JMLR. org, 2017.

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

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

[10] 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.

[11] 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.

[12] 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.

[13] Kuan Fang, Yuke Zhu, Animesh Garg, Silvio Savarese, and Li Fei-Fei. Dynamics learning with cascaded variational inference for multi-step manipulation. *CoRL 2019*, 2019.

[14] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2017.

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

[16] Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Yoshua Bengio, and Sergey Levine. Infobot: Transfer and exploration via the information bottleneck. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[17] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

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

[19] 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.

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

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

[22] Brian Ichter and Marco Pavone. Robot motion planning in learned latent spaces. *CoRR*, abs/1807.10366, 2018. URL `http://arxiv.org/abs/1807.10366`.

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

[24] 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.

[25] Dinesh Jayaraman, Frederik Ebert, Alexey Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[26] Huaizu Jiang, Deqing Sun, Varun Jampani, Ming-Hsuan Yang, Erik Learned-Miller, and Jan Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[27] Tom Jurgenson, Or Avner, Edward Groshev, and Aviv Tamar. Sub-goal trees–a framework for goal-based reinforcement learning. *ICML*, 2020.

[28] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical planning in the now. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[29] 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.

[30] Taesup Kim, Sungjin Ahn, and Yoshua Bengio. Variational temporal abstraction. In *Advances in Neural Information Processing Systems 32*, pages 11566–11575. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/9332-variational-temporal-abstraction.pdf`.

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

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

[33] Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compositional imitation learning: Explaining and executing one task at a time. *ICML*, 2019.

[34] Craig A Knoblock. Learning abstraction hierarchies for problem solving. In *AAAI*, pages 923–928, 1990.

[35] 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.

[36] 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.

[37] Yuezun Li and Siwei Lyu. Exposing deepfake videos by detecting face warping artifacts. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[38] 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.

[39] 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.

[40] Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

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

[42] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2146–2153. IEEE, 2017.

[43] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *arXiv preprint arXiv:1909.05829*, 2019.

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

[45] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive convolution. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[46] Simon Niklaus, Long Mai, and Feng Liu. Video frame interpolation via adaptive separable convolution. In *IEEE International Conference on Computer Vision*, 2017.

[47] Giambattista Parascandolo, Lars Buesing, Josh Merel, Leonard Hasenclever, John Aslanides, Jessica B Hamrick, Nicolas Heess, Alexander Neitz, and Theophane Weber. Divide-and-conquer monte carlo tree search for goal-directed planning. *arXiv preprint arXiv:2004.11410*, 2020.

[48] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 2050–2053, 2018.

[49] Chris Paxton, Yotam Barnoy, Kapil D. Katyal, Raman Arora, and Gregory D. Hager. Visual robot task planning. *CoRR*, abs/1804.00062, 2018. URL `http://arxiv.org/abs/1804.00062`.

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

[51] 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.

[52] 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.

[53] Oleh Rybkin, Kostas Daniilidis, and Sergey Levine. Simple and effective vae training with calibrated decoders, 2020.

[54] Earl D Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial intelligence*, 5(2): 115–135, 1974.

[55] Hiroaki Sakoe. Dynamic-programming approach to continuous speech recognition. In *1971 Proc. the International Congress of Acoustics, Budapest*, 1971.

[56] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

[57] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv:1912.04443*, 2019.

[58] 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.

[59] 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.

[60] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[61] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

[62] 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.

[63] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2018.

[64] Tsun-Hsuan Wang, Yen-Chi Cheng, Chieh Hubert Lin, Hwann-Tzong Chen, and Min Sun. Point-to-point video generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10491–10500, 2019.

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

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

[67] 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.

[68] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[69] 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.

[70] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

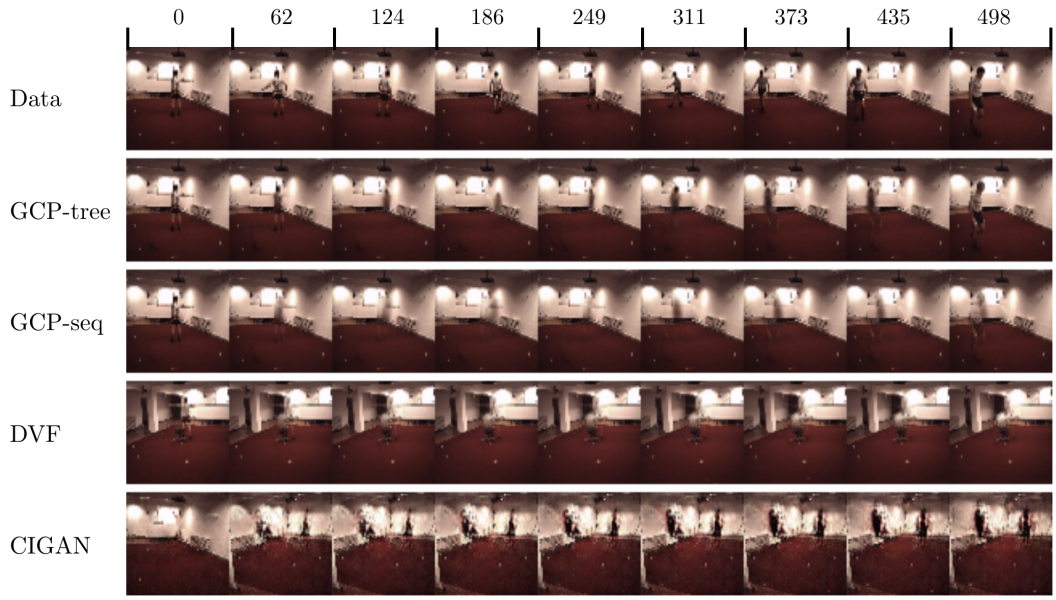

Figure 7: Predictions on Human 3.6M. We see that the GCP models are able to faithfully capture the human trajectory. The optical flow-based method (DVF) captures the background but fails to generate complex motion needed for long-term goal-conditioned prediction. Causal InfoGan also struggles to capture 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`.

Table 5: Prediction performance on perceptual metrics.

Dataset	Pick&Place		Human 3.6M		9-room Maze		25-room Maze	
Method	FVD	LPIPS	FVD	LPIPS	FVD	LPIPS	FVD	LPIPS
GCP-tree	430.3	**0.02**	**1314.3**	**0.05**	**655.50**	**0.174**	**413.31**	**0.168**
GCP-sequential	**328.9**	**0.02**	1541.8	**0.06**	860.04	0.214	638.95	0.238
DVF [39]	2879.9	0.06	1704.6	**0.05**	1320.34	0.231	1476.44	0.215
CIGAN [35]	3252.6	0.12	2528.5	0.17	1440.6	0.190	677.40	0.219

A Additional results

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

B Evidence lower bound (ELBO) derivation

We wish to optimize the likelihood of the sequence conditioned on the start and the goal frame $p(o_{2:T-1}|o_{1,T})$. However, due to the use of latent variable models, this likelihood is intractable, and we resort to variational inference to optimize it. Specifically, we introduce an approximate posterior network $q(z_{2:T-1}|o_{1:T})$, where that approximates the true posterior [32, 51]. The ELBO can be derived from the objective that consists of likelihood and a term that enforces that the approximate posterior matches the true posterior:

$$\begin{aligned}\ln p(o_{2:T-1}|o_{1,T}) &\geq \ln p(o_{2:T-1}|o_{1,T}) - \mathrm{KL}(q(z_{2:T-1}|o_{1:T})||p(z_{2:T-1}|o_{1:T})) \\ &= \mathbb{E}_{q(z_{2:T-1}|o_{1:T})}\left[\ln p(o_{2:T-1}|o_{1,T}, z_{2:T-1})\right] - \mathrm{KL}\left(q(z_{2:T-1}|o_{1:T})\right) \,||\, p(z_{2:T-1}|o_{1,T})\right), \quad (4)\end{aligned}$$

where the last equality is simply a rearrangement of terms.

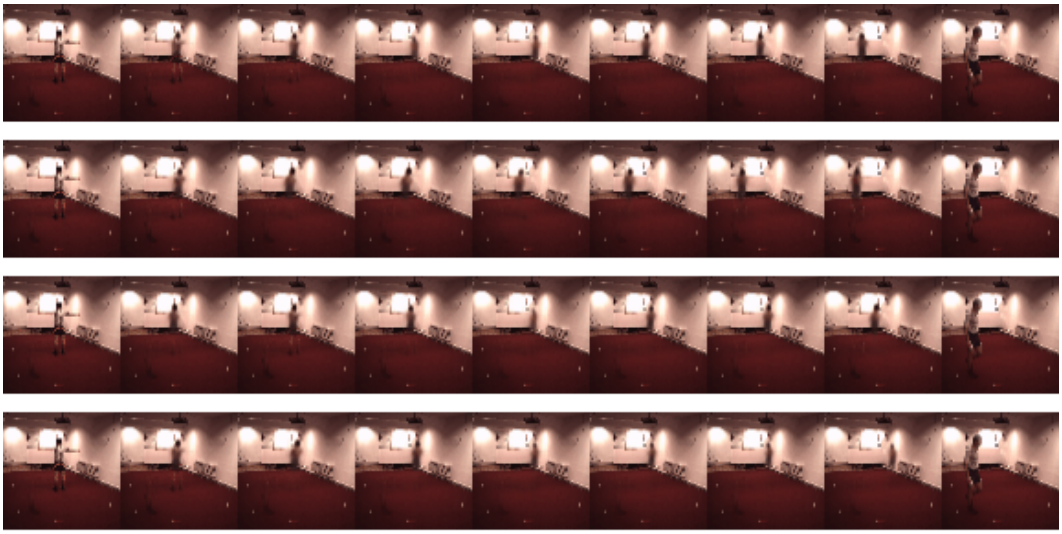

Figure 8: Prior samples from GCP-tree on the Human 3.6M dataset. Each row is a different prior sample conditioned on the same information.

Further, in order to efficiently parametrize these distributions, we factorize the distributions as follows according to the graphical model in Fig 2 (right) and Eq. 2:

$$p(o_{2:T-1}|o_{1,T}, z_{2:T-1}) = \prod_{t=2}^{T-1} p(o_t|o_{1,T}, z_t), \tag{5}$$

$$p(z_{2:T-1}|o_{1,T}) = \prod_{t=2}^{T-1}, \tag{6}$$

$$q(z_{2:T-1}|o_{1:T}) = \prod_{t=2}^{T-1} q(z_t|o_t, \text{pa}(t)). \tag{7}$$

We therefore require the following distributions to define our model: $p(o_t|o_{1,T}, z_t)$, $p(z_t|\text{pa}(t))$, $q(z_t|o_t, \text{pa}(t))$. The parameterization of these distributions is defined in Section 3.4. The parent operator $\text{pa}(t)$ returns the parent nodes of s_t according to the graphical model in Fig 2 (right). Using these factorized distributions, we can write out the ELBO in more detail as:

$$\ln p(o_{2:T-1}|o_{1,T}) \geq \mathbb{E}_{q(z_{2:T-1}|o_{1:T})} \sum_{t=2}^{T-1} \left[\ln p(o_t|o_{1,T}, z_t) - \text{KL}\left(q(z_t|o_t, \text{pa}(t)) \;||\; p(z_t|\text{pa}(t))\right)\right]. \tag{8}$$

C Architecture

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 256-dimensional. All hidden layers in the Multi-Layer Perceptron have 256 neurons. We add skip-connections from the encoder activations from the first image to the decoder for all images. For the inference network we found it beneficial to use a 2-layer 1D temporal convolutional network that adds temporal context into the latent vectors e_t. For the recursive predictor that predicts e_n, we use group normalization [66]. We found that batch normalization [23] 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. Further, for the simple RNN (without the LSTM architecture)

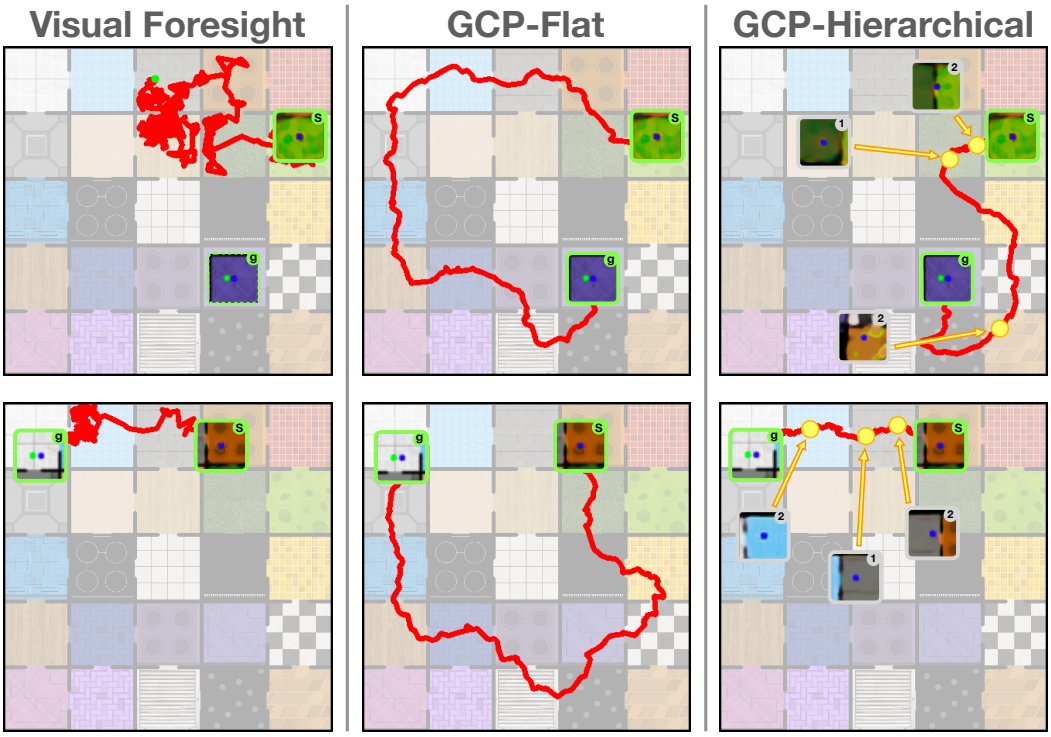

Figure 9: Comparison of visual planning & control approaches. Execution traces of Visual Foresight (**left**), GCP-tree with non-hierarchical planning (**middle**) and GCP-tree with hierarchical planning (**right**) on two 25-room navigation tasks. Visualized are start and goal observation for all approaches 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.

ablation of our tree model, we activate e_n with hyperbolic tangent (tanh). 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 using TreeLSTM [58] as the backbone of the hierarchical predictor significantly improved performance over vanilla recurrent architectures.

To increase the visual fidelity of the generated results when predicting images, we use a foreground-background generation procedure similar to [63]. The decoding distribution $p(o_t|s_t)$ is a mixture of discretized logistics [56], which we found to work better than alternative distributions. We use the mean of the decoding distribution as the prediction.

For the adaptive binding model, the frame o_t corresponding to the node s_n is not known before the s_n is produced. We therefore conditioned the inference distribution on the entire evidence sequence $o_{1:T}$ via the attention mechanism over the embeddings [2, 40]: $q(z_t) = \text{Att}(\text{enc}(o_{1:T}), \text{pa}(t))$. We reuse the same observation embeddings e_t for the attention mechanism values.

Hyperparameters. The convolutional encoder and decoder both have five layers. We use the Rectified Adam optimizer [38, 31] 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. Training took a day for all datasets except the 25-room dataset, where we train the models for 3 days.

D 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 [12] that is based on the Mujoco physics simulator [59]. 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.

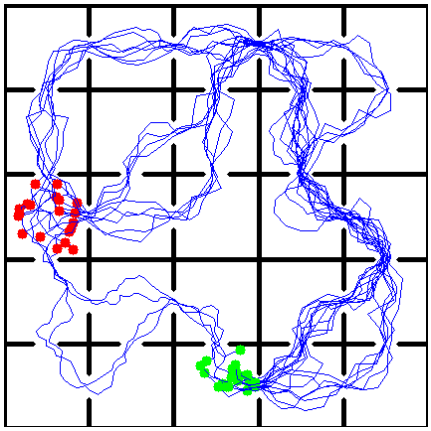

Figure 10: Example trajectory distributions between fixed start (red) and goal (green) rooms on the 25-room navigation task. The example goal-reaching behavior is highly suboptimal, with both strong multimodality in the space of possible solutions as well as low-level noise in each individual trajectory.

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 [6]. 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. [29]) 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.

E Planning Experimental Setup

For planning with GCPs we use the model architectures described in Section C trained on the navigation data described in Section D. The hyperparameters for the hierarchical planning experiments are listed in Table 6. 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 4, 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 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.

Table 6: 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	$\sum_{t=0}^{T-1}(x_{t+1} - x_t)^2$

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 [41]. We use a planning horizon of 50 steps and replan after the current plan is executed. We cannot use the cost function from Table 6 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 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.

F Adaptive Binding with Dynamic Programming

F.1 An efficient inference procedure

To optimize the model with adaptive binding, we perform variational inference on both w and z:

$$\log p(x) \geq \mathbb{E}_{q(z,w)}[p(x|w,z)] - D_{KL}(q(z|x)||p(z)) - D_{KL}(q(w|x,z)||p(w)). \tag{9}$$

To infer $q(w|x,z)$, we want to produce a distribution over possible alignments between the tree and the evidence sequence. Moreover, certain alignments, such as the ones that violate the ordering of the sequence are forbidden. We define such distribution over aligment matrices A via Dynamic Time Warping. We define the energy of an alignment matrix as the cost, and the following distribution over alignment matrices:

$$p(A|x,y) = \frac{1}{Z} e^{-A*c(x,z)},$$

where $Z = \mathbb{E}_A[e^{-A*c(x,z)}]$, and c is the MSE error between the ground truth frame x_t and the decoded frame associated with z_n. We are interested in computing marginal edge distributions $w = \mathbb{E}_A[A]$. Given these, we can compute the reconstruction error efficiently. We next show how to efficiently compute the marginal edge distributions.

Given two sequences $x_{0:T}, z_{0:N}$, denote the partition function of aligning two subsequences $x_{0:i}, z_{0:j}$ as $f_{i,j} = \sum_{A \in \mathcal{A}_{0:i,0:j}} e^{-A*c(x_{0:i},z_{0:j})}$. [7] shows that these can be computed efficiently as:

$$f_{i,j} = c(x_i, z_j) * (f_{i-1,j-1} + f_{i-1,j}).$$

Futhermore, denote the partition function of aligning $x_{i:T}, z_{j:N}$ as $b_{i,j} = \sum_{A \in \mathcal{A}_{i:T,j:N}} e^{-A * c(x_{i:T}, z_{j:N})}$. Analogously, we can compute it as:

$$b_{i,j} = c(x_i, z_j) * (b_{i+1,j+1} + b_{i+1,j}).$$

Proposition 1 *The total unnormalized density of all alignment matrices that include the edge (i, j) can be computed as $e_{i,j} = f_{i,j} * b_{i,j} / c(x_i, z_j) = c(x_i, z_j) * (f_{i-1,j-1} + f_{i-1,j}) * (b_{i+1,j+1} + b_{i+1,j})$. Moreover, the probability of the edge (i, j) can be computed as $w_{i,j} = e_{i,j}/Z$.*

Proposition 1 enables us to compute the expected reconstruction loss in quadratic time:

$$w * c(x, y).$$

F.2 Bottleneck Discovery Experimental Setup

In order to use the adaptive binding model to discover bottleneck frames that are easier to predict, we increase the reconstruction loss on those nodes as described in the main text. Specifically, we use Gaussian decoding distribution for this experiment, and set the variance of the decoding distribution for several top layers in the hierarchy to a fraction of the value for lower layers. This encourages the model to bind the frames that are easier to predict higher in the hierarchy as the low variance severely penalizes poor predictions. We found this simple variance re-weighting scheme effective at discovering bottleneck frames on several environments.

To generate the visualization of the discovered tree structure in Fig. 6 we evenly subsample the original 80-frame sequences and display those nodes that bound closest to the subsampled frames such that the resulting graph structure still forms a valid 2-connected tree. The variations in tree structure arise because the semantic bottlenecks which the nodes specialize on binding to appear at different time steps in the sequences due to variations in speed and initial position of the robot arm as well as initial placement of the objects.

G Training from Random Data

In the room navigation experiments we train our model with noisy trajectories that reach diverse goals with considerable suboptimality (see Fig. 10). To test whether our method can work with even more sub-optimal training data, we conduct preliminary experiments with completely random exploration data, and observe that our method still successfully solves navigation tasks in the 9-room environment (see Fig. 11). This suggests that the proposed method is scalable even to situations where no good planners exist that can be used for data collection.

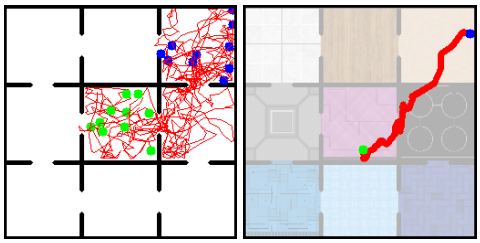

Figure 11: **Left**: random exploration data. **Right**: execution of our method trained on random data.

In Tab. 7, we compare the average trajectory length of training data and our method on both, the dataset used for the experiments in section 5.1 and the random action data. We find that planning with our method leads to substantially shorter trajectories, further showing the ability of our approach to improve upon low-quality training data.

Table 7: Average Trajectory Length. Planning with GCP finds shorter paths than the training distribution.

	ORIGINAL DATA	RANDOM DATA
TRAINING DATA	31.4	62.6
GCP-TREE (OURS)	**20.7**	**42.6**