Self-Supervised Visual Planning with Temporal Skip Connections

Frederik Ebert[1, 2], Chelsea Finn[1], Alex X. Lee[1], and Sergey Levine[1]

[1]Department of Electrical Engineering and Computer Sciences, UC Berkeley, United States
[2]Department of Computer Science, Technical University of Munich, Germany
{febert,cbfinn,alexlee_gk,svlevine}@berkeley.edu

Abstract: In order to autonomously learn wide repertoires of complex skills, robots must be able to learn from their own autonomously collected data, without human supervision. One learning signal that is always available for autonomously collected data is prediction. If a robot can learn to predict the future, it can use this predictive model to take actions to produce desired outcomes, such as moving an object to a particular location. However, in complex open-world scenarios, designing a representation for prediction is difficult. In this work, we instead aim to enable self-supervised robot learning through direct video prediction: instead of attempting to design a good representation, we directly predict what the robot will see next, and then use this model to achieve desired goals. A key challenge in video prediction for robotic manipulation is handling complex spatial arrangements such as occlusions. To that end, we introduce a video prediction model that can keep track of objects through occlusion by incorporating temporal skip-connections. Together with a novel planning criterion and action space formulation, we demonstrate that this model substantially outperforms prior work on video prediction-based control. Our results show manipulation of objects not seen during training, handling multiple objects, and pushing objects around obstructions. These results represent a significant advance in the range and complexity of skills that can be performed entirely with self-supervised robot learning.

Keywords: deep learning, video prediction, manipulation, model-based reinforcement learning

1 Introduction

A key bottleneck in enabling robots to autonomously learn a wide range of skills is the need for human involvement, for example, through hand-specifying reward functions, providing demonstrations, or resetting and arranging the environment for episodic learning. Learning action-conditioned predictive models of the environment is a promising approach for self-supervised robot learning as it allows the robot to learn entirely on its own from autonomously collected data. In order for a robot to be able to predict what will happen in response to its actions, it needs a representation of the environment that is suitable for prediction. In complex, open-world environments, it is difficult to construct a concise and sufficient representation for prediction. At a high level, the number and types of objects in the scene might change substantially trial to trial, making it impossible to provide a single fixed-size representation. At a low level, the type of information that is important about each object might change from object to object: the motion of a rigid box might depend only on its shape and friction coefficient, while the motion of a deformable stuffed animal might depend on a variety of other
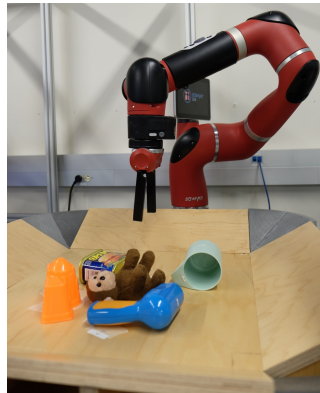

Figure 1: The robot learns to move new objects from self-supervised experience.

factors which are hard to determine by hand. Instead of engineering representations for different object types and object scenes, we can instead directly predict the robot's sensory observations. The rationale behind this is that, if the robot is able to predict future observations, it has acquired a sufficient understanding of the environment that it can leverage for planning actions.

A major challenge in directly predicting visual observations for robotic control lies in deducing the spatial arrangement of objects from ambiguous observations. For example, when one object passes in front of another, the robot must remember that the occluded object still persists. In human visual perception, this is referred to as object permanence, and is known to take several months to emerge in infants [1]. When the robot is commanded to manipulate an object which can become occluded during a manipulation, it must be able to accurately predict how that object will respond to an occlusion. In this work, we propose a visual predictive model for robotic control that can reason about spatial arrangements of objects in 3D, while using only monocular images and without providing any form of camera calibration. Previous learning methods have proposed to explicitly model object motion in image-space [2, 3] or to explicitly model 3D motion using point cloud measurements from depth cameras [4], and lacked the capability to maintain information about objects which are occluded during the predicted motion. We propose a simple model that does not require explicitly deducing full 3D structure in the scene, but does provide effective handling of occlusions by storing the appearance of occluded objects in memory. Furthermore, unlike prior prediction methods [4, 5] our model does not require any additional, external systems for providing point-to-point correspondences between frames of different time-steps.

The technical contribution of this work is three-fold. First, we present a video prediction method that can more accurately maintain object permanence through occlusions, by incorporating temporal skip connections. Second, we propose a planning objective for control through video prediction that leads to significantly improved long-term planning performance, including planning through occlusions, when compared to prior work [2]. Finally, we propose a mechanism for planning with both discrete and continuous actions with video prediction models. Our evaluation demonstrations that these components can be combined to enable a learned video prediction model to perform a range of real-world pushing tasks. Our experiments include manipulation of previously unseen objects, handling multiple objects, pushing objects around obstructions, and moving the arm around and over other obstacle-objects, representing a significant advance in the range and complexity of skills that can be acquired through entirely self-supervised learning.

2 Related Work

Large-Scale, Self-Supervised Robotic Learning. Large-scale robotic data collection has been explored in a number of recent works. Oberlin and Tellex [6] proposed to collect object scans using multiple robots. Several prior works have focused on autonomous data collection for individual skills, such as grasping [7, 8] or obstacle avoidance [9, 10]. In contrast to these methods, our approach learns predictive models that can be used to perform a variety of manipulations, and does not require a success measure or reward function during data collection. Several prior approaches have also sought to learn inverse or forward models from raw sensory data without any supervision [11, 2]. While these methods demonstrated effective generalization to new objects, they were limited in the complexity of tasks and time-scale at which these tasks could be performed. The method proposed by Agrawal et al. [11] was able to plan single pokes, and then greedily execute multiple pokes in sequence. The method of Finn and Levine [2] performed long-horizon planning, but was only effective for short motions. In our comparisons to this method, we demonstrate a substantial improvement in the length and complexity of manipulations that can be performed with our models.

Sensory Prediction Models. Action-conditioned video prediction has been explored in the context of synthetic video game images [12, 13] and robotic manipulation [14, 15, 16], and video prediction without actions has been studied for unstructured videos [17, 18, 19] and driving [20, 3].

Several works have sought to use more complex distributions for future images, for example by using autoregressive models [16, 21]. While this often produces sharp predictions, the resulting models are extremely demanding computationally, and have not been applied to real-world robotic control. In this work, we extend video prediction methods that are based on predicting a transformation from the previous image [15, 3]. Prior work has also sought to predict motion directly in 3D, using 3D

point clouds obtained from a depth camera [4], requiring point-to-point correspondences over time, which makes it hard to apply to previously unseen objects. Our predictive model is effective for a wide range of real-world object manipulations and does not require 3D depth sensing or point-to-point correspondences between frames. Prior work has also proposed to plan through learned models via differentiation, though not with visual inputs [22]. We instead use a stochastic, sampling-based planning method [23, 2], which we extend to handle a mixture of continuous and discrete actions.

3 Preliminaries

In this section, we define our image-based robotic control problem, present a formulation of visual model predictive control (visual MPC) over pixel motion, and summarize prior video prediction models based on image transformation.

3.1 Visual Model Predictive Control

Our visual MPC problem formulation follows the problem statement outlined in prior work [2]. We assume that the user defines a goal for the robot in terms of pixel motion: given an image from the robot's camera, the user can choose one or more pixels in the image, and choose a destination where each pixel should be moved. For example, the user might select a pixel on an object and ask the robot to move it 10 cm to the left. Formally, the user specifies P source pixel locations $d_0^{(1)}, \ldots, d_0^{(P)}$ in the initial image I_0, and P goal locations $g^{(1)}, \ldots, g^{(P)}$. The source and goal pixel locations are denoted by the coordinates $(x_d^{(i)}, y_d^{(i)})$ and $(x_g^{(i)}, y_g^{(i)})$. Given a goal, the robot plans for a sequence of actions $\mathbf{a}_{1:T}$ over T time steps, where T is the planning horizon. The problem is formulated as the minimization of a cost function c which depends on the predicted pixel positions $d_t^{(j)}$. The planner makes use of a learned model that predicts pixel motion. Given a distribution over pixel positions $P_{t_0, d^{(i)}} \in \mathbb{R}^{H \times W}, \sum_{H,W} P_{t_0, d^{(i)}} = 1$ at time $t = 0$, the model predicts distributions over its positions $P_{t, d^{(i)}}$ at time $t \in \{1, \ldots, T\}$. To achieve the best results with imperfect models, the actions can be iteratively replanned at each real-world time step $\tau \in \{0, \ldots, \tau_{max}\}$ following the framework of model-predictive control (MPC): at each real-world step τ, the model is used to plan T steps into the future, and the first action of the plan is executed. At the first real-world time step $\tau = 0$, the distribution $P_{t=0, d^{(i)}}$ is initialized as 1 at the location of the designated pixel and zero elsewhere. In subsequent steps ($\tau > 0$), the 1-step ahead prediction of the previous step is used to initialize $P_{t=0, d^{(i)}}$. The cost c is a function of $P_{t, d^{(i)}}$ at all predicted time steps $0...T$. Prior work proposed to use the negative log-probability of placing each pixel at its goal location as the cost [2]. We will discuss in Section 5 how performance can be improved substantially by instead using expected distances. Planning is performed using the cross-entropy method (CEM), a gradient-free optimization procedure that consists of iteratively resampling action sequences and refitting Gaussian distributions to the actions with the best predicted cost. Further details can be found in prior work [2].

3.2 Video Prediction via Pixel Transformations

Visual MPC requires a model that can effectively predict the motion of the selected pixels $d_0^{(1)}, \ldots, d_0^{(P)}$ up to T steps into the future. In this work, we extend the model proposed in [15], where this flow prediction capability emerges implicitly, and therefore no external pixel motion supervision is required. Future images are generated by transforming past observations. The model uses stacked convolutional LSTMs that predict a collection of pixel transformations at each time step, with corresponding composition masks. In this model, the previous image I_t is transformed by N separate transformations, and all of the transformed images $\tilde{I}_t^{(i)}$ are composited together according to weights obtained from the predicted masks. Intuitively, each transformation corresponds to the motion of a different object in the scene, and each mask corresponds to the spatial extents of that object. Let us denote the N transformed images as $\tilde{I}_t^{(1)}, \ldots, \tilde{I}_t^{(N)}$, and the predicted masks as $\mathbf{M}_1, \ldots \mathbf{M}_N$, where each 1-channel mask is the same size as the image. The next image prediction is then computed by compositing the images together using the masks: $\hat{I}_{t+1} = \sum_{i=1}^{N} \tilde{I}_t^{(i)} \mathbf{M}_i$. To predict multiple time steps into the future, the model is applied recursively. These transformations can be represented as convolutions, where each pixel in the transformed image is formed by ap-

plying a convolution kernel to the previous one. This method can represent a wide range of local transformations. When these convolution kernels are normalized, they can be interpreted as transition probabilities, allowing us to make probabilistic predictions about future locations of individual pixels. To predict the future positions of the designated pixels $d^{(i)}$, the same transformations which are used for the images are applied to $P_{t,d^{(i)}}$ such that $P_{t+1,d^{(i)}} = \frac{1}{P_s} \sum_{i=1}^{N} \tilde{P}_{t,d^{(i)}}^{(i)} \mathbf{M}_i$, where $\frac{1}{P_s}$ is a normalizing constant to ensure that $P_{t+1,d^{(i)}}$ adds up to 1 over the spatial dimension of the image. Since this prior predictive model outputs a single image at each time step, it is unable to track pixel positions through occlusions. Therefore, this model is only suitable for planning motions where the user-selected pixels are not occluded during the manipulation, which restricts its use in cluttered environments or with multiple selected pixels. In the next section, we discuss our proposed occlusion-aware model, which lifts this limitation by employing temporal skip connections.

4 Skip Connection Neural Advection Model

To enable effective tracking of objects through occlusions, we propose an extension to the dynamic neural advection (DNA) model [15] that incorporates temporal skip connections. This model uses a similar multilayer convolutional LSTM structure; however, the transformations are now conditioned on a context of previous images. In the most generic version, this involves conditioning the transformation at time t on all of the previous images $I_0, ...I_t$, though in practice we found that a greatly simplified version of this model performed just as well in practice. We will therefore first present the generic model, and then describe the practical simplifications. We refer to this model as the skip connection neural advection model (SNA), since it handles occlusions by copying pixels from prior images in the history such that when a pixel is occluded (e.g., by the robot arm or by another object) it can still reappear later in the sequence. When predicting the next image $\hat{I}_{t+1}$, the generic SNA model transforms each image in the history according to a different transformation and with different masks to produce $\hat{I}_{t+1}$ (see Figure 10 in the appendix):

$$\hat{I}_{t+1} = \sum_{j=t-T}^{t} \sum_{i=1}^{N} \mathbf{M}_{i,j} \tilde{I}_j^{(i)} \tag{1}$$

In the case where $t < T$, negative values of j simply reuse the first image in the sequence. This generic formulation can be computationally expensive, since the number of masks and transformations scales with $T \times N$. A more tractable model, which we found works comparably well in practice in our robotic manipulation setting, assumes that occluded objects are typically static throughout the prediction horizon. This assumption allows us to dispense the intermediate transformations and only provide a skip connection from the very first image in the sequence, which is also the only real image, since all of the subsequent images are predicted by the model:

$$\hat{I}_{t+1} = I_0 \mathbf{M}_{N+1} + \sum_{i=1}^{N} \tilde{I}_t^{(i)} \mathbf{M}_i \tag{2}$$

This model only needs to output $N + 1$ masks. We observed similar prediction performance when using a transformed initial image $\tilde{I}_0$ in place of I_0, and therefore used the simplified model in Equation (2) in all of our experiments. We provide an example of the model recovering from occlusion in Figure 5. In the figure, the arm moves in front of the designated pixel, marked in blue in Figure 3. The graphs in Figure 4 show the predicted probability of the designated pixel, which is stationary during the entire motion, being at its original position at each step. Precisely when the arm occludes the designated pixel, the pixel's probability of being at this point decreases. This indicates that the model is 'unsure' where this pixel is. When the arm unoccludes the designated pixel, it should become visible again, and the probability of the designated pixel being at its original position should go up. In the case of the DNA model and its variants [15], the probability mass does not increase after the object reappears. This is because the DNA model cannot recover information about object that it has 'overwritten' during its predictions. Consequently, for these models, the probability stays low and *moves with the arm*, which causes the model to believe that the occluded pixel also moves with the arm. We identified this as one of the major causes of planning failure when using the prior models. By contrast, in the case of our SNA model, the probability of the correct pixel position increases again rapidly right after the arm unoccludes the object. Furthermore, the probability of the unoccluded object's position becomes increasingly sharp at its original position as the arm moves further away.
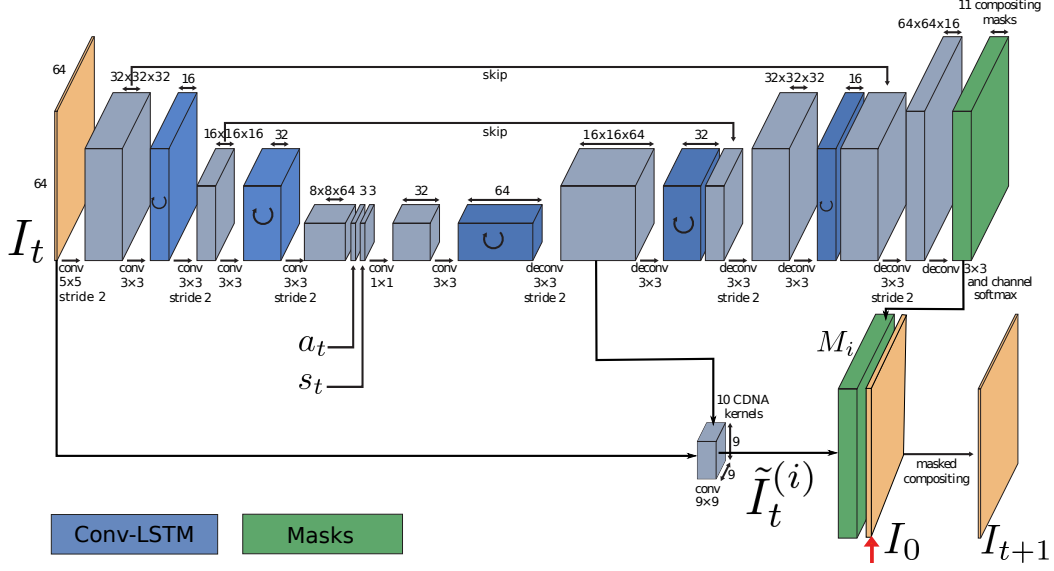
Figure 2: Simplified SNA model based on Equation (2). The red arrow indicates where the image from the first time step I_0 is concatenated with the transformed images $\tilde{I}_t^{(i)}$ multiplying each channel with a separate mask to produce the predicted frame for step $t + 1$.
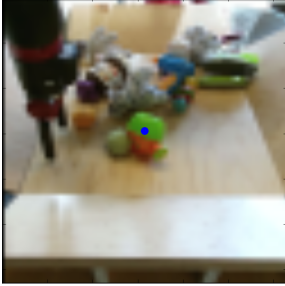

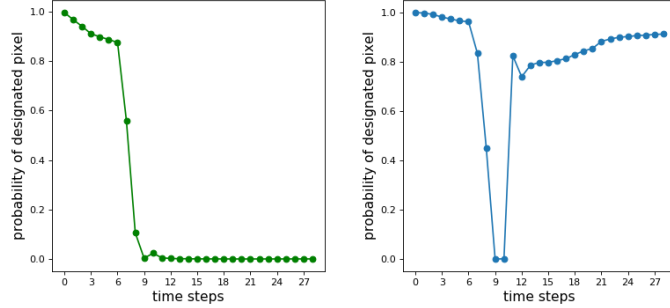Figure 3: The blue dot indicates the designated pixel

Figure 4: Predicted probability $P_{d^{(0)}}(t)$ of the designated pixel being at the location of the blue dot indicated in Figure 3 for the DNA model (left) and the SNA model (right).

5 Visual MPC with Pixel Distance Costs

The choice of objective function for visual MPC has a large impact on the performance of the method. Intuitively, as the model's predictions get more uncertain further into the future, the planner relies more and more on the cost function to provide a reasonable estimate of the overall distance to the goal. Prior work used the probability of the chosen pixel(s) reaching their goal position(s) after T time steps, as discussed in Section 3.1. When the trajectories needed to reach the goal are long, the objective provides relatively little information about the progress towards the goal, since most of the predicted probabilities will have a value close to zero at the goal-pixel. In this work, we propose a cost function that still makes use of the uncertainty estimates about the pixel position, but provides a substantially smoother planning objective, resulting in improved performance for more complex, longer-horizon tasks. A straightforward choice of smooth cost function in deterministic settings is the Euclidean distance between the current and desired location of the desired pixel(s), given by $\|d_{t'} - g\|_2$. Since our video prediction model produces a distribution over the pixel location at each time step, we can use the expected value of this distance as a cost function, summed over the entire

5
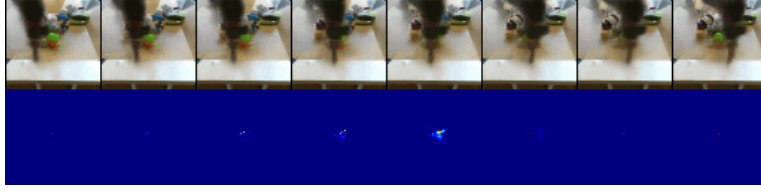
(a) Skip connection neural advection (SNA) does not erase or move objects in the background

(b) Standard DNA [2] exhibits undesirable movement of the distribution $P_{d^{(0)}}(t)$ and erases the background

Figure 5: Top rows: Predicted images of arm moving *in front of* green object with designated pixel (as indicated in Figure 3). Bottom rows: Predicted probability distributions $P_{d^{(0)}}(t)$ of designated pixel obtained by repeatedly applying transformations.

Figure 6: Integrating actions in the vertical direction enables moving over obstacles. This results in more natural and shorter paths. Top row: executed trajectory, left: red diamond indicates designated pixel, green dot indicates goal, 2nd and 3rd row: predictions and probability of designated pixel at time step 3.

horizon T:

$$c_{t+1:t+T}(g) = c(I_{t-1:t}, \mathbf{x}_{t-1:t}, \mathbf{a}_{t:t+T}, d_t, g)$$
$$= \sum_{t'=t+1,\ldots,t+T} \mathbb{E}_{d_{t'} \sim P_{t',d^{(i)}}} [\|d_{t'} - g\|_2] \tag{3}$$

where the expectation can be computed by summing over all of the positions in each predicted image, and the cost corresponds to a (element-wise) Hadamard product of the pixel location probability map and the distances between each pixel position and the goal. This cost function encourages the movement of the designated objects in the right direction for each step of the execution, regardless of whether the g position can be reached within T time steps. For multi-objective tasks with multiple designated pixels $d^{(i)}$ the costs are summed to together weighting them equally. Although the use of well-shaped cost functions in MPC has been explored extensively in prior work [24], the combination of cost shaping and visual MPC has not been studied extensively.

6 Sampling-Based MPC with Continuous and Discrete Actions

The choice of action representation for visual MPC has a significant effect on the performance of the resulting controller. The action representation must allow the planner sufficient freedom to maneuver the arm to perform a wide variety of tasks, while constraining sufficiently so as to create a tractable search space. We found that a particularly well-suited action representation for tabletop manipulation can be constructed by combining continuous and discrete actions, in contrast to prior work that used only continuous end-effector motion vectors as actions [2]. The actions consist of the horizontal motion of the end-effector, as well as a discrete "lift" action that allows the robot to command the end-effector to lift off the table vertically. The discrete lifting action can take on N values (4 in our implementation) that specify for how many time steps the robot

	seen		unseen	
	distance mean, std.	improvement mean, std.	dist. mean, std.	improvement mean, std.
random actions	29.4 ± 3.2	-0.85 ± 2.3	N/A	N/A
DNA with distance metric as in [2]	25.9 ± 12.2	5.2± 13.7	24.6±10.6	2.1±12.4
DNA with our exp. dist. metric eqn. 3	**9.2 ±7.4**	**15. ± 11.5**	**17.5 ± 10.2**	**8.3 ± 11.8**
SNA exp. dist. metric eqn. 3 (ours)	13.0 ± 5.6	12.4 ± 8.8	18.18 ± 9.5	7.7 ± 10.5

Table 1: Results of the pushing benchmark on 20 different object/goal configurations. Units are pixels in the 64x64 images.

should lift the end-effector off the table. Unlike with continuous vertical motion commands, these discrete commands result in the end-effector staying off the table for multiple time steps even during random data collection. In order to incorporate this hybrid action space into the stochastic CEM-based optimization in visual MPC, we sample real-valued parameters for the discrete action, and then round them to the nearest valid integer to obtain discrete actions. As usual with CEM, we iteratively refit a multivariate Gaussian distribution to the best performing samples (which in our case are chosen to be in the 90th percentile of samples), treating the entire action vector as if it was continuous. Figure 6 shows an example of a situation where the discrete vertical motion component of the action is used by our model to lift the gripper over the object in order to push it from the opposite side.

7   Experiments

Our experimental evaluation compares the proposed occlusion-aware SNA video prediction model, as well as the improved cost function for planning, with a previously proposed model based on dynamic neural advection (DNA) [2]. We use a Sawyer robot, shown in Figure 1, to push a variety of objects in a tabletop setting. In the appendix, we include a details on hyperparameters, analysis of sample complexity, and a discussion of robustness and limitations. We evaluate long pushes and multi-objective tasks where one object must be pushed without disturbing another. The supplementary video and links to the code and data are available at `https://sites.google.com/view/sna-visual-mpc`

Figure 7 shows an example task for the pushing benchmark. We collected 20 trajectories with 3 novel objects and 1 training object. Table 1 shows the results for the pushing benchmark. The column *distance* refers to the mean distance between the goal pixel and the designated pixel at the final time-step. The column *improvement* indicates how much the designated pixel of the objects could be moved closer to their goal (or further away for negative values) compared to the starting location. The true locations of the designated pixels after pushing were annotated by a human.

Figure 7: Pushing task. The designated pixel (red diamond) needs to be pushed to the green circle.

The results in Table 1 show that our proposed planning cost in Equation (3) substantially outperforms the planning cost used in prior work [2]. The performance of the SNA model in these experiments is comparable to the prior DNA model [2] when both use the new planning cost, since this task does not involve any occlusions. Although the DNA model has a slightly better mean distance compared to SNA, it is well within the standard deviation, suggesting that the difference is not significant.

To examine how well each approach can handle occlusions, we devised a second task that requires the robot to push one object, while keeping another object stationary. When the stationary object is in the way, the robot must move the goal object around it, as shown in Figure 8 on the left. While doing this, the gripper may occlude the stationary object, and the task can only be performed successfully if the model can make accurate predictions through this occlusion. These tasks are specified by picking one pixel on the target object, and one on the obstacle. The obstacle is commanded to
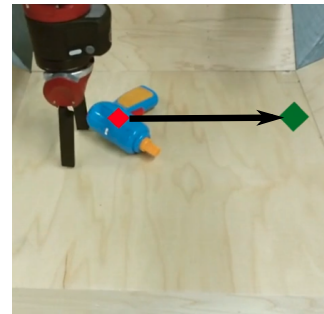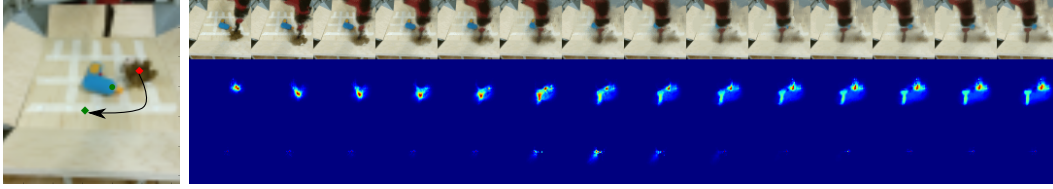
Figure 8: Left: Task setup with green dot marking the obstacle. Right, first row: the predicted frames generated by SNA. Second row: the probability distribution of the designated pixel on the *moving* object (brown stuffed animal). Note that part of the distribution shifts down and left, which is the indicated goal. Third row: the probability distribution of the designated pixel on the obstacle-object (blue power drill). Although the distribution increases in entropy during the occlusion (in the middle), it then recovers and remains on its original position.

	seen		unseen	
	moved imp. mean, std.	stationary imp. mean, std.	moved imp. mean, std.	stationary imp. mean, std.
Dynamic Neural Advection, DNA [2]	3.5 ±4.94	-1.4 ± 0.39	0.83 ±1.13	-1.1 ± 0.9
Skip Con. Neural Advection, SNA (ours)	**8.16 ± 2.9**	**-0.9 ±0.7**	**10.6 ± 3.7**	**-1.5 ± 0.9**

Table 2: Results for multi-objective pushing on 8 object/goal configurations with 2 seen and 2 novel objects. Values indicate improvement in distance from starting position, higher is better. Units are pixels in the 64x64 images.

remain stationary, while the target object destination location is chosen on the other side of the obstacle.

We used four different object arrangements, with two training objects and two objects that were unseen during training. We found that, in most of the cases, the SNA model was able to find a valid trajectory, while the prior DNA model was mostly unable to find a solution. Figure 8 shows an example of the SNA model successfully predicting the position of the obstacle through an occlusion and finding a trajectory that avoids the obstacle. These findings are reflected by our quantitative results shown in Table 2, indicating the importance of temporal skip connections.

8 Discussion and Future Work

We showed that visual predictive models trained entirely with videos from random pushing motions can be leveraged to build a model-predictive control scheme that is able to solve a wide range multi-objective pushing tasks in spite of occlusions. We also demonstrated that we can combine both discrete and continuous actions in an action-conditioned video prediction framework to perform more complex behaviors, such as lifting the gripper to move over objects.

Although our method achieves significant improvement over prior work, it does have a number of limitations. The behaviors in our experiments are relatively short. In principle, the visual MPC approach can allow the robot to repeatedly retry the task until it succeeds, but the ability to retry is limited by the model's ability to track the target pixel: the tracking deteriorates over time, and although our model achieves substantially better tracking through occlusions than prior work, repeated occlusions still cause it to lose track. Improving the quality of visual tracking of the designated pixels may allow the system to retry the task until it succeeds. More complex behaviors, such as picking and placing (e.g., to arrange a table setting), may also be difficult to learn with only randomly collected data. We expect that more goal-directed data collection would substantially improve the model's ability to perform complex tasks. Furthermore, better predictive models that incorporate hierarchical structure or reason at variable time scales would further improve the capabilities of visual MPC to carry out temporally extended tasks. Fortunately, as video prediction methods continue to improve, we expect methods such as ours to further improve in their capability.

Acknowledgments

References

[1] R. S. Bogartz, J. L. Shinskey, and T. Schilling. Object permanence in five-and-a-half month old infants? *Infancy*, 1, 2000.

[2] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[3] B. De Brabandere, X. Jia, T. Tuytelaars, and L. Van Gool. Dynamic filter networks. In *Neural Information Processing Systems (NIPS)*, 2016.

[4] A. Byravan and D. Fox. Se3-nets: Learning rigid body motion using deep neural networks. *arXiv preprint arXiv:1606.02378*, 2016.

[5] J. Walker, A. Gupta, and M. Hebert. Dense optical flow prediction from a static image. In *International Conference on Computer Vision (ICCV)*, 2015.

[6] J. Oberlin and S. Tellex. Autonomously acquiring instance-based object models from experience. In *International Symposium on Robotics Research*, 2015.

[7] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *International Conference on Robotics and Automation (ICRA)*, 2016.

[8] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *International Journal of Robotics Research (IJRR)*, 2016.

[9] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

[10] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. *arXiv preprint arXiv:1704.05588*, 2017.

[11] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, 2016.

[12] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, 2015.

[13] S. Chiappa, S. Racaniere, D. Wierstra, and S. Mohamed. Recurrent environment simulators. *arXiv preprint arXiv:1704.02254*, 2017.

[14] B. Boots, A. Byravan, and D. Fox. Learning predictive models of a depth camera & manipulator from raw execution traces. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.

[15] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *Neural Information Processing Systems (NIPS)*, 2016.

[16] N. Kalchbrenner, A. v. d. Oord, K. Simonyan, I. Danihelka, O. Vinyals, A. Graves, and K. Kavukcuoglu. Video pixel networks. *arXiv preprint arXiv:1610.00527*, 2016.

[17] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *International Conference on Learning Representations (ICLR)*, 2016.

[18] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, 2015.

[19] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, 2016.

[20] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *International Conference on Learning Representations (ICLR)*, 2017.

[21] S. Reed, A. v. d. Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, D. Belov, and N. de Freitas. Parallel multiscale autoregressive density estimation. *arXiv preprint arXiv:1703.03664*, 2017.

[22] I. Lenz and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In *In RSS*. Citeseer, 2015.

[23] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[24] G. Li, B. Lennox, and Z. Ding. Infinite horizon model predictive control for tracking problems. 1, June 2005. ISSN 1948-3449. doi:10.1109/ICCA.2005.1528173.

[25] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Appendix

Hyperparameters

The video prediction network was trained on sequences of 15 steps taken from 44,000 trajectories of length 30 by randomly shifting the 15-step long window thus providing a form of data augmentation. The model was trained on 66,000 iterations with a minibatch-size of 32, using a learing rate of 0.001 with the Adam optimizer [25]. For visual MPC we use $n_{iter} = 3$ CEM-iterations. At every CEM iteration $M = 200$ action sequences are sampled and the best $K = 10$ samples are selected, fitting a Gaussian to these examples. Then new actions are sampled according to the fitted distribution and the procedure is repeated for n_{iter} iterations.

Prototyping in Simulation

The visual MPC algorithm has been prototyped and tested on a simple block pushing task simulated in the MuJoCo physics engine. The code has been made available in the same repository. Using the simulator, training data could be collected orders of magnitude faster. Furthermore a benchmark was set up which does not require any manual reset or manual labeling of the objects' positions. The main downside of simulation however is that the complexity of both real-world dynamics and real-world visual scenes cannot be matched. (More realistic simulators exist, but require very large computational resources and large amounts of hand-engineering for setup).

Dependence of Model Performance on Amount of Training Data

We tested the effect of using different amounts of training data and compare the results visually, see Figure 9. The complete size of the data set is 44,000 trajectories. For this evaluation all models were trained for 76k iterations. The quality of the predictions for the probability distribution of the designated pixel vary widely with different amounts of training data and the quality of these predictions is crucial for the performance of visual MPC. The fact that performance still improves with very large amounts of training data suggests that the model has sufficient expressive power to leverage this amount of experience. In the future we will investigate the effect of using even more data and evaluate the effect of a larger number of different training objects.

Failure Cases and Limitations

Visual MPC fails when objects have vastly different appearance than those used in training. For example when objects are much bigger than any of the training objects these methods tend to perform poorly. Also we observed failure for one object with a very bright color which did not occur in the training set. However the model usually generalizes well to novel objects of similar size and appearance as the training objects. Apart from that we observed two main failure modes: The first one occurs when the probability mass of the designated pixel, $P_{t,d^{(i)}}$ drifts away from the object of interest (as discussed in the Section 8) since there is no feedback correcting the position of the designated pixel. To alleviate this problem a tracker can be integrated into the system enabling visual MPC to keep retrying when object behave differently than expected.

The second failure mode occurs when visual MPC does not find an action sequence which moves the designated pixel closer to the goal. This can happen, when the goalpoint for pushing the object is far away and it is very unlikely to sample a trajectory moving closer to the goal. In some cases the problem can be avoided simply by increasing the number of samples used when applying CEM. However this slows down the planning process. In order to enable more temporally extended action sequences a potential solution could be to increase the efficiency of the sampling process e.g. by introducing macro-actions which consist of a sequence of actions (learning macro-actions is out of the scope of this paper).
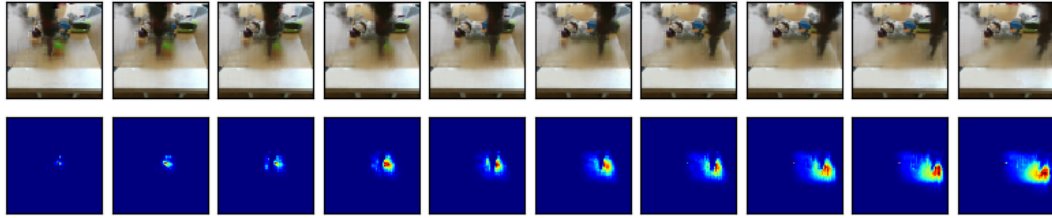
Robustness of the Model

We did not find any negative influence of clutter in the workspace, as long as the object to be moved has a free way path to its goal. When an obstacle is on the path it can be marked by a second designated pixel to avoid collision using an additional cost function. In this case the planner usually
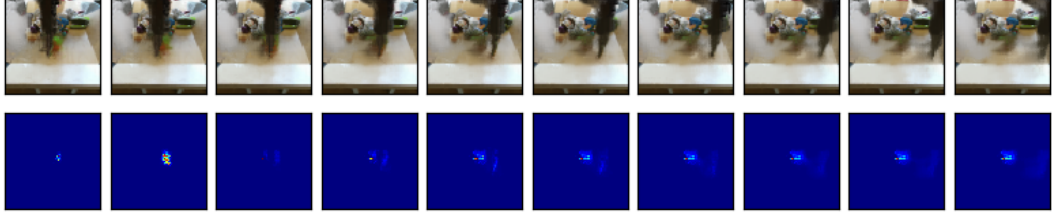
(a) using 5% of data
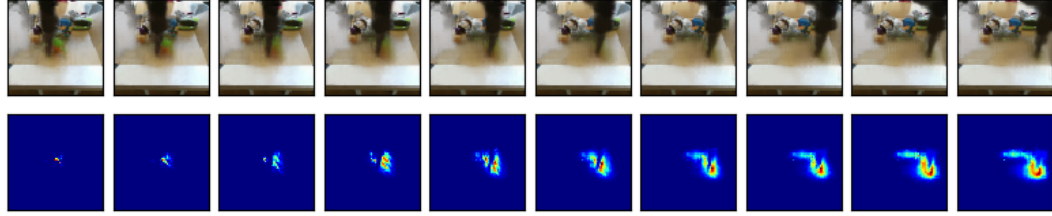
(b) using 10% of data

(c) using 100% of data

Figure 9: Top rows: Predicted images. Bottom rows: Predicted probability distributions of designated pixel on the green object, marked in Figure 3. When using only 5% of the data (of 44k trajectories in total), the model is not able to predict object movement at all (in fact the arm passes through the objects) and the distribution remains on the initial position of the object. For 30% percent the object distribution appears much more smeared than with 100% of the data.

manages to find a way around the obstacle. However visual MPC with the current type of model is not yet capable of reasoning by itself that it has to push and object around obstacles when it is not marked explicitly. The reason is that the model has large uncertainty when multiple collisions occur (i.e. when the arm pushes object 1 and object 1 collides with object 2).

Our model is robust to small changes of the viewpoint (in the order of several centimetres in translation and several degrees in orientation). A likely reason is that during data collection the camera has been slightly displaced in position. Robustness could be improved by training a model from several viewpoints at the same time. As the prediction problem also becomes harder in this way more data might be required.

General skip connection neural advection model

We show a diagram of the general skip connection neural advection model in Figure 10

Alternative Model Architectures

Instead of copying pixels from the first image of the sequence I_0, the pixels in the first image can also be transformed and then merged together with the transformed pixels from the previous time step as

$$\hat{I}_{t+1} = \tilde{I}_0 \mathbf{M}_{N+1} + \sum_{i=1}^{N} \tilde{I}_t^{(i)} \mathbf{M}_i \tag{4}$$
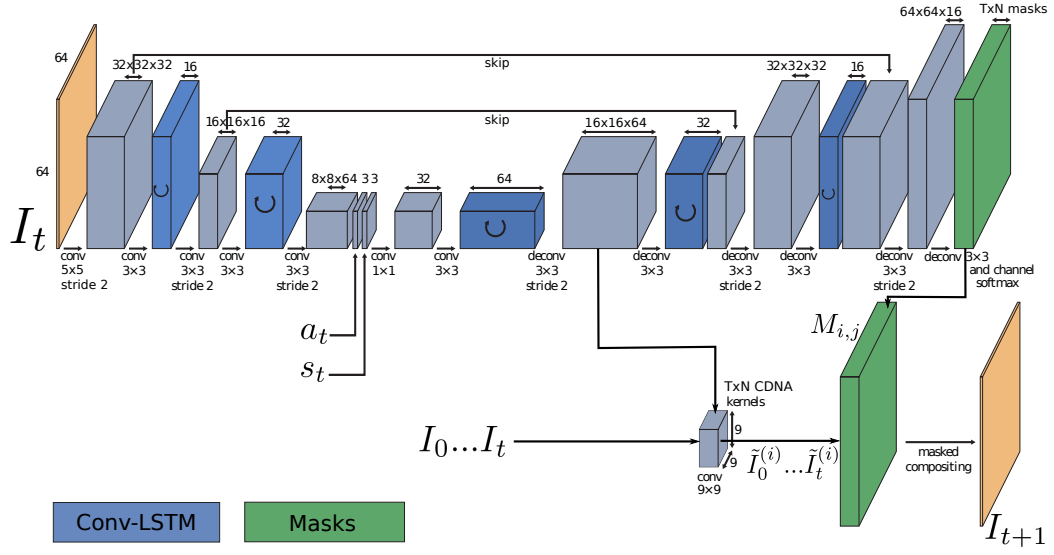
Figure 10: General SNA model based on Equation (1).

where $\tilde{I}_0$ is the transformed background. We tested this model on various pushing task and found that performance is comparable to the SNA model.