Robustness via Retrying: Closed-Loop Robotic Manipulation with Self-Supervised Learning

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

[1]Department of Electrical Engineering and Computer Sciences, UC Berkeley, United States
{febert, sdasari, cbfinn,alexlee_gk,svlevine}@berkeley.edu

Abstract: Prediction is an appealing objective for self-supervised learning of behavioral skills, for example for autonomous robots. However, incorporating prediction of future sensory inputs into an end-to-end framework for decision making and control poses a number of major challenges. How should the predictive model be used? What happens when the predictions are inaccurate? In this paper, we tackle these questions by proposing a method for learning complex robotic skills from raw image observations, using only autonomously collected experience. We show that even an imperfect model can complete complex tasks if it can continuously retry, but this requires the model to not lose track of the objective (e.g., the object of interest). By incorporating learned registration into our method, we can enable a robot to continuously retry the task until it gets it right. We demonstrate that this idea can be combined with a video-prediction based controller to enable complex behaviors to be learned from scratch using only raw visual inputs, including grasping, repositioning objects, and non-prehensile manipulation. Our real-world experiments demonstrate a model trained with 160 robot hours of autonomously collected data is able to successfully master complex manipulation tasks with a wide range of objects not seen during training.

1 Introduction

Humans have the ability to learn complex skills (such as manipulating objects) through millions of interactions with their environment during their lifetime. These interactions enable us to acquire a general understanding of the physical world and, notably, do not require supervision beyond observation of one's own actions and their consequences. Hence, self-supervised learning through prediction is an appealing direction of research as it enables intelligent systems to leverage and learn from massive amounts of raw data to autonomously acquire general skills. Yet, developing self-supervised learning systems using predictive models of raw sensory inputs presents a number of challenges: planning needs to account for imperfections in the predictive model and the robot needs a grounded mechanism for evaluating predicted futures. How can we enable systems to plan to perform complex tasks from raw sensory observations, even when the predictions may often be inaccurate?
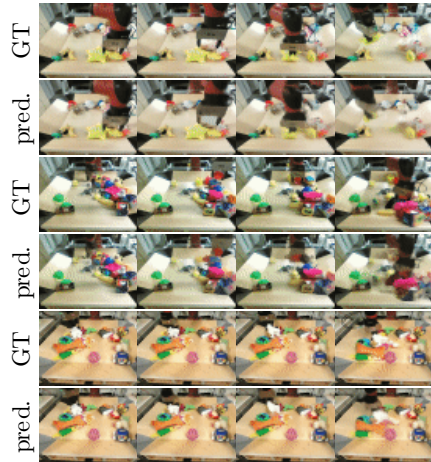

Figure 1: Ground truth and predictions from the model (only every 4 frames are shown). These examples show grasping, pushing, and simultaneous grasping and dragging.

Prior work on self-supervised robot learning has enabled robots to learn rudimentary, short-term manipulation skills such as grasping [1, 2], singulation [3], pushing [4, 5], poking [6], and other arm motions [7]. The question that we are concerned with in this work is: can self-supervised predictive models of raw visual observations be used to perform more complex and realistic tasks, especially tasks

that are temporally extended? While this might seem exceedingly challenging due to the difficulty of long-term prediction of images, we make the following observations that make it feasible to perform relatively long-horizon tasks. First, we observe that short-horizon planning can still give us a good indication of the potential for a plan to achieve a long-term goal, if provided with the right cost function. This fact has been known for decades, and is the basis for model-predictive control: iterative short-horizon replanning that aims to achieve long-term goals. Yet, maintaining an accurate estimate of the cost function throughout planning is crucial. For video prediction, we propose a cost function based on image registration, which we demonstrate can itself be learned without any human supervision using the same exact dataset as the one used to train the predictive model. The second idea that enables our method to perform long-horizon tasks is that, if the robot can always evaluate the goal, it can continuously retry, so that even flawed predictions allow for an eventual successful execution. We use the same learned registration model to enable this retrying behavior.

We demonstrate our method on the task of maneuvering unknown objects in a table-top setting using a robot manipulator. In order to autonomously learn to perform manipulation skills with high-fidelity, tasks need to be specified in way that allows for precision. To specify goals, we provide an image of the desired configuration. This provides a straight-forward and grounded mechanism for a human to provide a goal in the observation space of the robot. Building upon prior methods that use self-supervised predictive models for control [4, 5, 7], we develop a method that can plan actions with a video prediction model to achieve the desired state specified in the goal image. At each time step, our method uses a learned registration model to find correspondences between the current image and both the goal-image and the initial image. By doing so, we enable the robot to persistently attempt the task until completion. In contrast to the short-horizon pushing skills and arm motions demonstrated in prior work [4, 5, 7], we show that our video prediction model can be used to perform longer-term manipulations, autonomously choose to push and pick objects, and, when provided enough time, accomplish tasks significantly more consistently.

2 Related Work

Many self-supervised robot learning methods have focused on predicting the outcome of a particular event such as grasp success [1, 2, 3] or crashing [8, 9]. Consequently, these approaches can only recover a single policy that optimizes the probability of the predicted event. Instead, we aim to acquire a model that can be reused for multiple goals and behaviors. To do so, we build upon prior works that learn to predict the sequence of future observations, which can be used to optimize with respect to a variety of goals [4, 5, 7]. Unlike Byravan et al. [7], we demonstrate complex object manipulations with previously-unseen objects from RGB video. In contrast to prior self-supervised visual planning works [4, 5], we can perform substantially longer tasks, by using image registration with respect to a goal image. Goal observations have been previously used for specifying a reward function for robot learning systems [10, 11, 12, 13]. Unlike these methods, we use a learned registration to the goal image for measuring distance to the goal rather than distance in a learned feature space. Distances in unconstrained feature spaces can be arbitrary, while registration inherently measures how pixels should move and can therefore provide a calibrated distance metric with respect to the goal.

Another related robotics problem is visual servoing, where visual feedback is used to control a robot to reach a goal configuration [14, 15, 16]. Traditional approaches aim to minimize distances of feature points [17, 18, 19], or pixel intensities [20]. Learning and convolutional networks have also been incorporated into visual servoing [21, 22, 23, 2]. Unlike servoing approaches that use reactive control laws, we use multi-step predictive models to achieve temporally extended goals, while still using continuous visual feedback for retrying at every step. Furthermore, our method performs both prehensile and non-prehensile manipulation, while visual servoing typically assumes fully actuated control, often with a known Jacobian.

Model-predictive control (MPC) [24] has proven successful in a variety of robotic tasks [25, 26, 27, 28, 29]. MPC involves planning for controls with a short horizon and re-planning as the execution progresses. Thus, it provides the foundation of persistent retrying and the ability to overcome inaccuracies of the predictive model. However, as we later show, maintaining an accurate model of the *cost* used for planning throughout an episode is critical for success, and has prevented prior work on visual foresight [4, 5] from moving beyond short-term tasks. One of our primary contributions is a grounded mechanism for evaluating the planning cost of visual predictions, allowing persistent re-planning with video prediction models which enables longer-term manipulations.

It is also possible to use off-the-shelf trackers [30, 31, 32, 33] to address this issue. However, these trackers usually only have a limited capability to adapt to the domain they are applied to, and can lose track during occlusions. A key advantage of our learned registration approach, inspired by [34], is that we can obtain registration success metrics for every point in the image and every pair of images and therefore we can propose a weighting scheme for the planning costs according to the *uncertainty* of the individual registrations. Furthermore, since our method is completely self-supervised, it continues to improve as more data is collected by the robot.

3 Preliminaries

Our visual MPC problem formulation follows the problem statement outlined in prior work [4]. In this setting, an action-conditioned video prediction model g (typically represented by a deep neural network) is used to predict future camera observations $\hat{I}_{1:T} \in \mathbb{R}^{T \times H \times W \times 3}$ for a robot conditioned on a sequence of candidate actions $a_{1:T}$, (where T is the prediction horizon). This can be written as $\hat{I}_{1:T} = g(a_{1:T}, I_0)$, where I_0 is the starting frame from the current time-step. An optimization-based planner is then used to select the action sequence that results in an outcome that accomplishes a user-specified goal. This type of vision-based control is highly general, since it reasons over raw pixel observations without the requirement for a fixed-size state space, and has been demonstrated to generalize effectively to non-prehensile manipulation of previously unseen objects [4, 5].

Visual MPC assumes that the task can be defined in terms of pixel motion. Formally, in the initial image I_0 we define $i \in [0, ..n]$ source and goal pixel locations denoted by the coordinates $d_{0,i} \in \mathbb{N}^2$ and $d_{g,i} \in \mathbb{N}^2$. Given a goal, visual MPC plans for a sequence of actions $a_{1:T}$ over T time steps, where T is the planning horizon, to move the pixel at $d_{0,i}$ to $d_{g,i}$. If this pixel lies on top of an object, this corresponds to moving that entire object to a goal position. Note that this problem formulation resembles visual servoing, but is considerably more complex, since moving the object at d_0 might require complex non-prehensile or prehensile manipulation and the use of a deep video prediction model. The planning problem is formulated as the minimization of a cost function c, which in prior work [5] measures the distance between the predicted pixel positions $\hat{d}_\tau$ and the goal position d_g:

$$c_i = \sum_{\tau=1,...,T} \mathbb{E}_{\hat{d}_{\tau,i} \sim P_{\tau,i}} \left[\| \hat{d}_{\tau,i} - d_{g,i} \|_2 \right] \tag{1}$$

where $c_i \in \mathbb{R}$. The planner makes use of a learned model that predicts pixel motion. In this paper, we use a deterministic variant of the video prediction model architecture developed by Lee et al. [35], where the pixel transformations are specified by flow vectors. Given a distribution over pixel positions $P_{t_0,i} \in \mathbb{R}^{H \times W}, \sum_{H,W} P_{t_0,i} = 1$ at time $t = 0$, the model predicts distributions over its positions $P_{t,i}$ at time $t \in \{1, \ldots, T\}$. Planning is performed using the cross-entropy method (CEM) [36] which efficiently finds an action sequences where the cost c_i (i.e. the sum over time of the expected distances to the goal) is lowest. To achieve the best results with imperfect models, the actions sequence is replanned at each real-world time step $t \in \{0, ..., t_{max}\}$ following the framework of model-predictive control (MPC): at each real-world step t the first action of the best action sequence is executed. At the first real-world time step $t = 0$, the distribution $P_{\tau=0,i}$ is initialized as 1 at the location of the designated pixel and zero elsewhere. In prior work [5, 4], in subsequent steps ($t > 0$), the prediction of the previous step is used to initialize $P_{\tau=0,i}$. However this causes accumulating errors, often preventing the model from solving long-term tasks or responding to situation where the environment (e.g. objects) behave differently than expected. In effect, the model loses track of which object was designated in the initial image. In this work we propose a method for registering the current image to both a start and goal images, where the designated pixels are known, allowing the model to replan according to the most recent estimate of the objects position. Crucially, the method we propose for doing this is self-supervised using the same exact data that is used to train the video prediction model, allowing both the predictor and registration model to benefit from each episode of robot experience.

4 Retrying by Registration

While self-supervision enables training powerful models to predict raw sensor inputs (among other quantities), using these model for control (e.g., with model-predictive control) poses the challenge of defining an appropriate cost function. One naïve approach of formulating a cost function for video-prediction based control could be using the pixel-wise error between a *goal image* and the predicted

image. Goal images have the advantage that they are very general and do not make any domain specific assumptions. Minimizing this pixel-difference cost with respect to the action sequence passed into the model should result in a controller that tries to bring the system into the goal state. However there are a number of issues with this approach: first when objects in the image are far from the position in the goal image (e.g., they do not overlap) there is no gradient signal of the pixel-difference cost with respect to the actions. Second, due to the blurry predictions obtained from video-prediction model, the pixel-wise difference between the predictions and the goal image can become meaningless. Another possible approach would be to perform a registration between predicted video frames and the goal-image and using the average length of the warping vectors as a cost function for "closeness" to the goal image; however, registering blurry prediction to a sharp goal-image poses a similar challenge as before.

The main contribution of this work is a method for computing the planning cost based on image-to-image registration, which yields distances between tracked object and target location. As a result, the cost is well-shaped, allowing for efficient optimization.

Closed-loop video-prediction based control
When the robot interacts with an object, the position of that object may change in unexpected ways, and therefore it is crucial that the system can update its belief of where the target object is. Prior work on visual MPC lacked this capability. To enable the agent to "keep retrying" indefinitely until the task is accomplished we propose to use an image-to-image registration approach to find the target object in the current frame. We show that this tracking capability can be learned completely unsupervised from the data collected by the robot autonomously. As one might expect the self-supervised tracking approach works better for images that are closer to each other and sometimes fails when the entities in the image are too far apart. To increase

Figure 2: Closed loop control is achieved by registering the current image I_t globally to the first frame I_0 and the goal image I_g. In this example registration to I_0 succeeds while registration to I_g fails since the object in I_g is too far away.

accuracy and robustness in addition to the starting image our method can register to a goal-image. The goal image is optional and can be provided by a human or through demonstration.
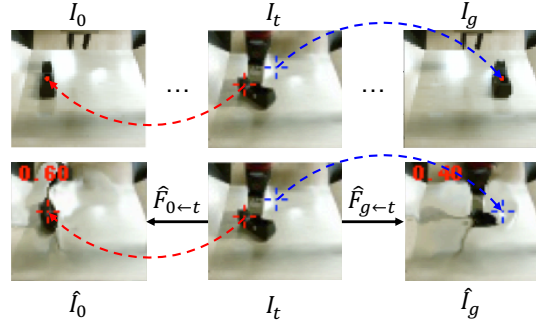
4.1 Test time procedure

We will first describe the architecture at test time (see Figure 3). The start and goal images I_0 and I_g are passed into the registration network R which is trained to return flow maps $\hat{F}_{0\leftarrow t} \in \mathbb{R}^{H\times W\times 2}$. The flow map is a vector field in $\mathbb{R}^2$ which describes the relative motion for every pixel between two frames.

$$\hat{F}_{0\leftarrow t} = R(I_t, I_0) \qquad\qquad \hat{F}_{g\leftarrow t} = R(I_t, I_g) \qquad (2)$$

The flow map $\hat{F}_{0\leftarrow t}$ can be used to warp the image of the current time step t to the start image I_0, and $\hat{F}_{g\leftarrow t}$ can be used to warp from I_t to I_g (see Figure 2 for an illustration):

$$\hat{I}_0 = \hat{F}_{0\leftarrow t} \diamond I_t \qquad\qquad \hat{I}_g = \hat{F}_{g\leftarrow t} \diamond I_t \qquad (3)$$

Where $\diamond$ denotes a bilinear interpolation operator which interpolates the pixel value bilinearly with respect to a location (x, y) and its four neighbouring pixels in the image. In addition to start and goal images, the user needs to specify one or several designated pixel positions (for simplicity showing only the case for one designated pixel) $d_0 \in \mathbb{N}^2$ in the start image and the corresponding pixels locations $d_g \in \mathbb{N}^2$ in the goal image (the goal image is optional) to define the task. While the registration network is trained to perform a global registration between the images, we only evaluate it at the points d_0 and d_g chosen by the user. This results in a cost function that ignores distractors. For a current image I_t, $\hat{F}_{0\leftarrow t}$ puts it in correspondence with I_0, and $\hat{F}_{g\leftarrow t}$ puts it in correspondence with I_g. The registration network is then used to find the pixel locations corresponding to d_0 and d_g
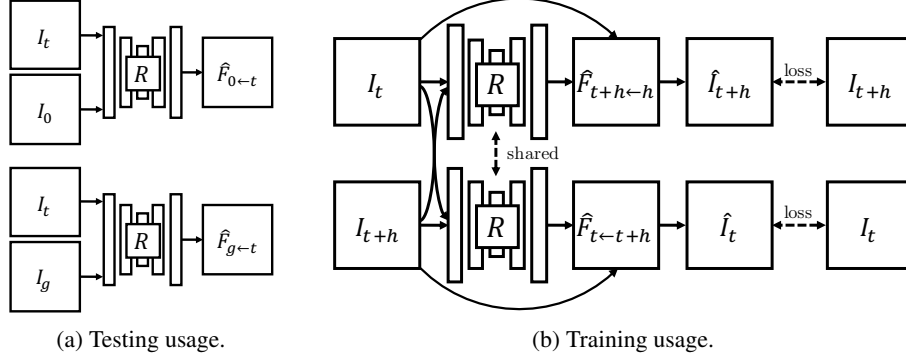
(a) Testing usage. (b) Training usage.

Figure 3: (a) At test time the registration network registers the current image I_t to the start image I_0 (top) and goal image I_g (bottom), inferring the flow-fields $\hat{F}_{0 \leftarrow t}$ and $\hat{F}_{g \leftarrow t}$. (b) The registration network is trained by warping images from randomly selected timesteps along a trajectory to each other.
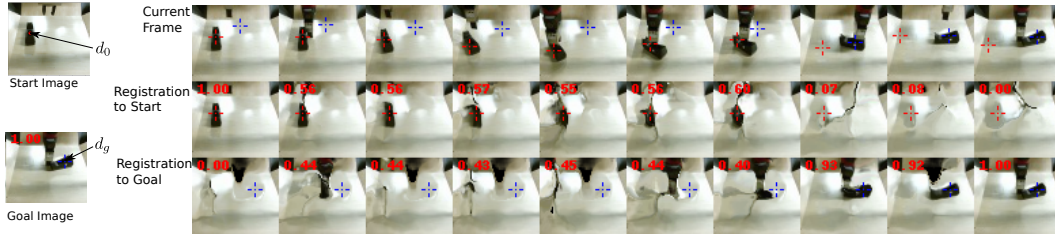

Figure 4: Outputs of registration network. The first row shows images from a trajectory executed by the robot, the second shows each image warped to the initial image via registration, and the third shows the same for the goal image. A successful registration in this visualization would result in images that closely resemble the start (or goal). In these images, the locations where the designated pixel of the start image d_0 and the goal image d_g are found is marked with red and blue crosses, respectively. It can be seen that, for the registration to the start image (red cross) the object is tracked for the first 7 frames, while the registration to the goal image (blue cross) succeeds for the last 3 time steps. The numbers in red indicate the trade off factors λ between the views and are used as weighting factors for the planning cost.

in the current frame:

$$\hat{d}_{0,t} = d_0 + \hat{F}_{0 \leftarrow t}(d_0) \qquad\qquad \hat{d}_{g,t} = d_g + \hat{F}_{g \leftarrow t}(d_g) \qquad\qquad (4)$$

Figure 4 visualizes the tracking results during a pushing task. On the left we show the start image and goal image provided by the user at the beginning of the trajectory. The first row shows the video, with the estimated location of d_0 marked in red and the estimated location of d_g marked in blue. In this example, registration succeeds with respect to the first image succeeds for the first 7 steps but fails after that. However registration with respect to the goal image succeeds for the last 3 steps. Thus, there is enough information at each time step to determine the cost, which we discuss in detail in the next section.

5   Planning Costs

As shown in the previous section, registration can fail when distances between objects in the images are large. Here we propose a mechanism that estimates which image is registered correctly, allowing us to utilize only the successful registration for evaluating the planning cost. When registering to both start and goal-image or when using more than one view, every pair of designated pixel and goal pixel defines a cost function (the expected distance to to the goal). As the robot approaches the goal, the difference to the initial image increases, making it increasingly harder to register. However, the difference to the goal image (hopefully) decreases, making it easier to register. We exploit this to compute accurate costs for all stages of the task by using a weighting factor that gives a high weight to the designated pixels $\hat{d}_{i,t}$ that are successfully tracked and a low (ideally zero) weight to the designated pixels where the registration is poor. We propose to use the photometric distance (we found that the norm in RGB space works well) between the true frame and the warped frame as an estimate for *local* registration success. A low photometric error indicates that the registration network predicted a flow vector leading to a pixel with a similar color, thus indicating warping

success. However this does not necessarily mean that the flow vector points to the correct location. For example, there could be several objects with the same color and the network could simply point to the wrong object. Letting $I_i(d_i)$ denote the pixel value in image I_i for position d_i, and $\hat{I}_i(d_i)$ denote the corresponding pixel in the image warped by the registration function (i.e., $\hat{I}_i = \hat{F}_{i \leftarrow t} \diamond I_t$, where I_t is the current image), we can define the general weight factors λ_i as

$$\lambda_i = \frac{||I_i(d_i) - \hat{I}_i(d_i)||_2^{-1}}{\sum_j^N ||I_j(d_j) - \hat{I}_j(d_j)||_2^{-1}}. \tag{5}$$

In the case of the single view model and a single designated pixel, the index i iterates over the start and goal image (and $N = 2$), in the case of multi-view models or multiple designated pixels the index also loops over the views and indices of the designated pixels. The MPC cost is then computed as the average of the costs c_i weighted by λ_i, where each c_i is the expected distance (see Equation 1) between the predicted pixel positions starting from the registered point $\hat{d}_{i,t}$ and the goal point $d_{g,i}$, where the registered point is found by $\hat{d}_{i,t} = d_i + \hat{F}_{i \leftarrow t}(d_i)$ (analogous to Equation 4). Finally the cost used for planning is $c = \sum_i \lambda_i c_i$.

5.1 Training procedure

The registration network R is trained to find correspondences between two images by warping one image to the other. The network is trained on the same data as the video-prediction model, but it does not share parameters with it.[1] This approach is similar to the optic flow method proposed by Meister et al. [34]. However, unlike this prior work, our method computes registrations for frames that might be many time steps apart, and the goal is not to extract optic flow, but rather to determine correspondences between potential highly distinct images. For training, two images are sampled at random times steps t and $t + h$ along the trajectory and the images are warped to each other in both directions.

$$\hat{I}_t = \hat{F}_{t \leftarrow t+h} \diamond I_{t+h} \qquad\qquad \hat{I}_{t+h} = \hat{F}_{t+h \leftarrow t} \diamond I_t \tag{6}$$

The network, which outputs $\hat{F}_{t \leftarrow t+h}$ and $\hat{F}_{t+h \leftarrow t}$ (see Figure 3), is trained to minimize the photometric distance between $\hat{I}_t$ and I_t and $\hat{I}_{t+h}$ and I_{t+h}, in addition to a smoothness regularizer that penalizes abrupt changes in the outputted flow-field. The details of this loss function follow prior work [34]. We found that gradually increasing the temporal distance h between the images during training yielded better final accuracy, as it created a curriculum for the model. The network R is implemented as a fully convolutional network which takes in as input the two images stacked together along the channel dimension. We use three convolutional layers each followed by a bilinear downsampling operation followed by three layers of convolution each followed by a bilinear upsampling operation (all convolutions use stride 1). By using bilinear sampling instead of we avoid artifacts that would be present when using strided convolutions and deconvolutions.

6 Scaling up Visual Model-Predictive Control

Extension to multiple cameras. Prior has considered visual MPC with a single camera [4, 5], where objects are manipulated on a plane. To define goals in 3D, we extend visual MPC to include multiple camera views (with two cameras in our prototype). Since tasks are defined in terms of pixel motion in 2D image space, the combination of multiple 2D tasks with cameras oriented appropriately defines a 3D task. In our experiments, we show that we can define 3D manipulation tasks (such as lifting an object from the table) that would be ambiguous using only a single camera view. The registration method described in the previous section is used separately per view to allow for dynamic retrying to solve temporally extended tasks. The planning costs from each view are combined using weighted averaging where the weights are provided by the registration network (see equation 5).

Combined prehensile and non-prehensile manipulation. In prior work on video-prediction based robotic manipulation [5, 4] the capabilities that emerged out of self-supervised learned were generally restricted to pushing and dragging objects. To enable more complex and temporally

---

[1]in principle sharing parameters with the video-prediction model might be beneficial, however this is left for future work
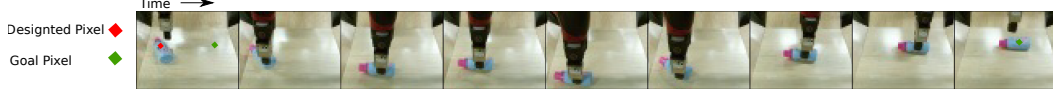
Figure 6: Applying our method to a pushing task. In the first 3 time instants the object behaves unexpectedly, moving down. The tracking then allows the robot to retry, allowing it to eventually bring the object to the goal.
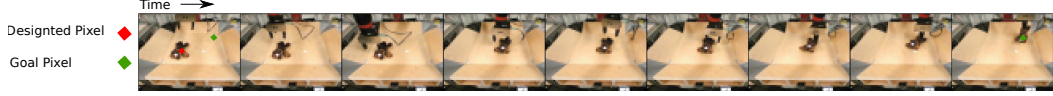

Figure 7: Retrying behaviour of our method combining prehensile and non-prehensile manipulation. In the first 4 time instants shown the agent pushes the object. It then loses the object, and decides to grasp it pulling it all the way to the goal. Retrying is enabled by applying the learned registration to both camera views (here we only show the front view).

extended tasks, we also explore how visual MPC can enable behaviors that include picking and lifting objects for rearrangement. One of the main challenges with this is that random exploration is unlikely to pick up objects a sufficiently large fraction of the time to allow the model to learn grasping skills. To alleviate this challenge, we provide a simple "reflex" to the robot during data collection, where the gripper automatically closes when the height of the wrist above the table is lower than a small threshold. This reflex is inspired by the palmar reflex observed in infants [37]. With this primitive, about 20% of training trajectories included some sort of grasp on an object. It is worth noting that, other than this reflex, no grasping-specific engineering was applied to the policy, and the video prediction model still needed to learn to predict the raw image frames of these grasps to take advantage of picking skills. In our experiments, we evaluate our method using data obtained both with and without the grasping reflex, evaluating both purely non-prehensile and combined prehensile and non-prehensile manipulation.

7 Experiments

Our experimental evaluation, conducted using two Rethink Sawyer robotic manipulators, evaluate the ability of our method to learn both prehensile and non-prehensile object relocation tasks entirely through autonomously collected data and self-supervision. In particular, we aim to answer the following questions: (1) How does our MPC approach with self-supervised goal image registration compare to alternative cost functions, such as off-the-shelf tracking, forward prediction via flow-based models, and pixel difference costs? (2) When the robot can continuously retry a task with goal image registration, how much is the success rate for object relocation tasks improved? (3) Can we learn predictive models that enable both non-prehensile and prehensile object manipulation? We also present additional experimental comparisons in a simulated environment. Videos and visualizations can be found on this webpage: https://sites.google.com/view/robustness-via-retrying.
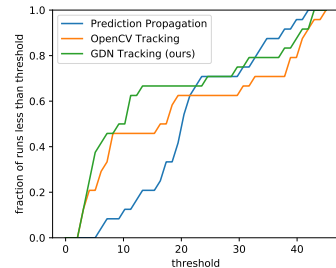

Figure 5: Real-world benchmark for pushing with 20 different tasks evaluated on unseen objects. Fraction of runs where final distance (in pixel units of 48x64 image) is lower than threshold. Our method shows a clear gain over OpenCV tracking and predictor propagation.

7.1 Real-World Experiments

To train both our prediction and registration models, we collected 20,000 trajectories of pushing motions and 15,000 trajectories with gripper control, where the robot was allowed to randomly move and pick up objects. The data collection process is fully autonomous, requiring human intervention only to replace and change out the objects in front of the robot. The action space consisted of Cartesian movements along the x, y, and z axes, and changes in azimuthal orientation of the gripper. For evaluation, we selected novel objects that were never seen during training. The evaluation tasks required the robot to move objects in its environment from a starting state to a goal configuration, and performance was evaluated by measuring the distance between the final object position and goal position. In all experiments, the maximum episode length was 50 time steps.

Pushing with retrying. In the first experiment, we aim to evaluate the performance of different visual MPC cost functions, including our proposed self-supervised registration cost. For this exper-

iment, we disable the gripper control, which requires the robot to push objects to the target. We evaluate 20 different pushing tasks that require pushing an object to a target position. For comparisons, we include a baseline where the object is tracked using the "multiple instance learning tracker" MIL [32] in OpenCV. Note that our method does not have any prior knowledge of objects – it is only provided with the position of one designed pixel in the initial and goal images, and must use the learned model to infer that this pixel belongs to an object that can be moved by the robot. We also compare to the visual MPC method proposed by Ebert et al. [5], which does not track the object explicitly, but relies on the flow-based video prediction model to keep track of the designated pixel, which we call "predictor propagation." We also tested a method where the visual MPC cost is calculated by directly evaluating the error between the predicted image and goal image, but we found that this approach was unable to make meaningful progress in moving the object to the target. Instead it resulted in a policy that always moves the arm to the position indicted in the target image, as the arm accounts for the majority of the movable pixel mass in the image. This can be interpreted as undesirable local minimum which is not present in tracking based cost we propose.

Figure 5 illustrates that our approach not only outperforms prior work [5], but also outperforms the hand-designed object tracker [32]. This is due to the fact that using our learned registration the agent is more frequently able to successfully recovering from situations where the object behaved differently than predicted the model predicted (see Figure 6).

Combined prehensile and non-prehensile manipulation. In the setting where the gripper is enabled it is part of the task to decide whether to solve a task by prehensile or non-prehensile manipulation. Similarly to the pushing setting we perform a benchmark where we define a set of 20 object relocation tasks and measure the final distance between the object and the target at the end of the episode. Interestingly we observe that in the majority of the cases the agent decides to grasp the object, as can be seen in the supplementary video. Figure 8 shows that the performance of our method is comparable with the performance of OpenCV tracking.
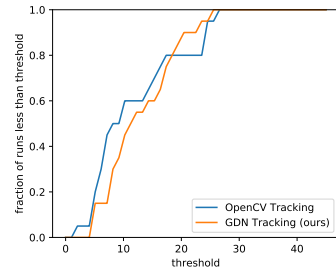

Figure 8: On the real-world grasping benchmark our method is on par with OpenCV tracking.

8 Discussion

In both [38, 39] pushing of unknown objects is learned from interaction data between the robot and objects. However the models employed in these works rely on hand-engineered input features (e.g. in the form of the approach angle to the object [39] or a set of points on the silhouette of an object [38]) which can make it hard to scale to complex real-world scenarios where for example objects can be partially occluded or cluttered. It as been shown in [40] and others that grasps on unseen objects can be found by matching the sensed 3D geometry to a database of precomputed object-grasp pairs. One downside to this approach is that physics are generally ignored in this scheme. For example information about mass and friction properties which can be estimated from visual clues (but not from geometry alone) cannot be leveraged. In [41] large databases of 3D meshes are used to label synthetic depth images with an analytic physics based grasp quality metric. This approach has the downside of relying on a suitable metric which is not available for other manipulation tasks (such as pushing) in general. Furthermore most prior approaches are not capable of selecting whether a grasping or pushing (or dragging) strategy is better suited to solve the given task and also do not exhibit a "retrying" behavior for recovering from failure. Video-prediction based manipulation is more general than many existing methods for robotic manipulation such as grasping specific [42, 40, 43] or pushing specific works [38, 39] because it does not require any human intervention at training time (i.e. human labels), large databases of 3D objects meshes and does not make any assumptions about object boundaries or the dimensionality of the state space.

Our method uses a video prediction model to predict future frames from autonomously gathered exploration data, and uses this same data to train a registration model that can be used to evaluate the distance to the goal for the predicted frames. We demonstrate that using a cost function derived from this distance substantially improves performance on temporally extended tasks. We further show that, by including a simple grasping "reflex" inspired by the palmar reflex in infants, we can effectively learn both non-prehensile and prehensile object relocation skills, allowing the robot to plan to pick up

and move objects when necessary. Our experiments show a very large improvement in success rates compared to a prior visual MPC method [5].

References

[1] 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.

[2] 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.

[3] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *arXiv preprint arXiv:1803.09956*, 2018.

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

[5] F. Ebert, C. Finn, A. X. Lee, and S. Levine. Self-supervised visual planning with temporal skip connections. *CoRR*, abs/1710.05268, 2017. URL http://arxiv.org/abs/1710.05268.

[6] 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.

[7] A. Byravan, F. Leeb, F. Meier, and D. Fox. SE3-Pose-Nets: Structured deep dynamics models for visuomotor planning and control. *arXiv preprint arXiv:1710.00489*, 2017.

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

[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] M. Jagersand and R. Nelson. Visual space task specification, planning and control. In *International Symposium on Computer Vision*, 1995.

[11] K. Deguchi and I. Takahashi. Image-based simultaneous control of robot and target object motions by direct-image-interpretation method. In *International Conference on Intelligent Robots and Systems (IROS)*, 1999.

[12] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, 2015.

[13] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel. Deep spatial autoencoders for visuomotor learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016.

[14] S. Hutchinson, G. D. Hager, and P. I. Corke. A tutorial on visual servo control. *IEEE transactions on robotics and automation*, 12(5):651–670, 1996.

[15] D. Kragic and H. I. Christensen. Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, 15, 2002.

[16] G. N. DeSouza and A. C. Kak. Vision for mobile robot navigation: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 24(2):237–267, 2002.

[17] J. T. Feddema and O. R. Mitchell. Vision-guided servoing with feature-based trajectory generation (for robots). *IEEE Transactions on Robotics and Automation*, 5(5):691–700, 1989.

[18] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3):313–326, 1992.

[19] W. J. Wilson, C. C. Williams Hulls, and G. S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5):684–696, 1996.

[20] G. Caron, E. Marchand, and E. M. Mouaddib. Photometric visual servoing for omnidirectional cameras. *Autonomous Robots*, 35(2-3):177–193, 2013.

[21] A. Saxena, H. Pandya, G. Kumar, A. Gaud, and K. M. Krishna. Exploring convolutional networks for end-to-end visual servoing. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[22] Q. Bateux, E. Marchand, J. Leitner, and F. Chaumette. Visual servoing from deep neural networks. In *International Conference on Robotics and Automation (ICRA)*, 2018.

[23] A. X. Lee, S. Levine, and P. Abbeel. Learning visual servoing with deep features and fitted Q-iteration. *International Conference on Learning Representations (ICLR)*, 2017.

[24] E. F. Camacho and C. B. Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[25] D. H. Shim, H. J. Kim, and S. Sastry. Decentralized nonlinear model predictive control of multiple flying robots. In *Decision and control, 2003. Proceedings. 42nd IEEE conference on*, volume 4, pages 3621–3626. IEEE, 2003.

[26] G. Allibert, E. Courtial, and F. Chaumette. Predictive control for constrained image-based visual servoing. *IEEE Transactions on Robotics*, 26(5):933–939, 2010.

[27] T. M. Howard, C. J. Green, and A. Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. In *Field and Service Robotics*, pages 69–78. Springer, 2010.

[28] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1714–1721. IEEE, 2017.

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

[30] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[31] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*. Springer, 2004.

[32] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.

[33] X. Mei and H. Ling. Robust visual tracking using 1 minimization. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1436–1443. IEEE, 2009.

[34] S. Meister, J. Hur, and S. Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. *arXiv preprint arXiv:1711.07837*, 2017.

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

[36] 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.

[37] D. Sherer. Fetal grasping at 16 weeks' gestation. *Journal of ultrasound in medicine*, 12(6), 1993.

[38] T. Hermans, F. Li, J. M. Rehg, and A. F. Bobick. Learning contact locations for pushing and orienting unknown objects. Georgia Institute of Technology, 2013.

[39] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini. *A vision-based learning method for pushing manipulation*, volume 54. University of Pennsylvania, 1993.

[40] C. Goldfeder, M. Ciocarlie, J. Peretzman, H. Dang, and P. K. Allen. Data-driven grasping with partial sensor data. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1278–1283. Citeseer, 2009.

[41] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[42] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[43] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *arXiv preprint arXiv:1710.01330*, 2017.

[44] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.
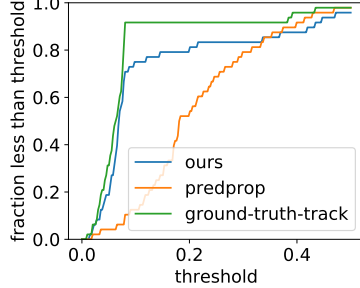
Appendix

.1 Simulated Experiments

Figure 9: Simulated evaluation. Fraction of trajectories with final object distance lower than threshold (higher is better).

In order to provide a more controlled comparison, we also set up a realistic simulation environment using MuJoCo [44], which includes a robotic manipulator controlled via Cartesian position control, similar to our real world setup, pushing randomly-generated L-shaped objects with random colors (see details in supplementary materials). We trained the same video prediction model in this environment, and set up 50 evaluation tasks where blocks must be pushed to target locations with maximum episode lengths of 120 steps. We compare our proposed registration-based method, "predictor propagation," and ground-truth registration obtained from the simulator, which provides an oracle upper bound on registration performance. Figure 9 shows the results of this simulated evaluation, where the x-axis shows different distance thresholds, and the y-axis shows the fraction of evaluation scenarios where each method pushed the object within that threshold. We can see that, for thresholds around 0.1, our method drastically outperforms predictor propagation (i.e., prior work [5]), and has a relatively modest gap in performance against ground-truth tracking. This indicates that our registration method is highly effective in guiding visual MPC, despite being entirely self-supervised.
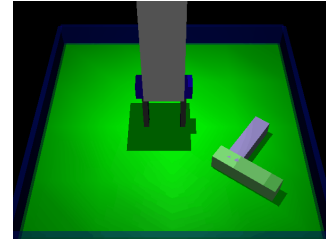

Figure 10: Block pushing simulator

In our simulated experiments, we use end-effector control with the arm illustrated in Figure 10. In this environment, the video prediction model was trained with using 60,000 training trajectories.

A   Improvements to online optimization procedure

In the visual MPC setting the action sequences found by the optimizer can be very different between execution times steps (not to be confused with prediction time steps). For example at one time step the optimizer might find a pushing action leading towards the goal and in the next time step it determines a grasping action to the optimal to reach the goal. Naive replanning at every time step can then result in alternating between a pushing attempt and a grasping attempt indefinitely causing the agent to get stuck and not making any progress towards to goal.

We show that we can resolve this problem by modifying the sampling distribution of the first iteration of CEM so that the optimizer commits to the plan found in the previous time step. In prior work [5] the sampling distribution at first iteration of CEM is chosen to be a Gaussian with diagonal covariance matrix and zero mean. We instead use the best action sequence found in the optimization of the previous time step as the mean. Since this action sequence is optimized for the previous time step we only use the values $a_{1:T}$ and omit the first action, where T is the prediction horizon. To sample actions close to the action sequence from the previous time step we reduce the entries of the diagonal covariance matrix for the first $T-1$ time steps. It is crucial that the last entry of the covariance matrix at the end of the horizon is not reduced otherwise no exploration could happen for the last time step causing poor performance at later time steps.