Accurate Vision-based Manipulation through Contact Reasoning

Alina Kloss[1], Maria Bauza[2], Jiajun Wu[2,3], Joshua B. Tenenbaum[2], Alberto Rodriguez[2] and Jeannette Bohg[1,3]

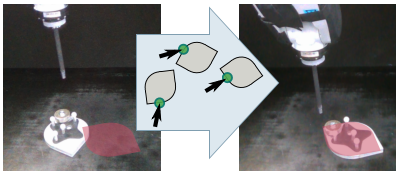
***Abstract*— Planning contact interactions is one of the core challenges of many robotic tasks. Optimizing contact locations while taking dynamics into account is computationally costly and, in environments that are only partially observable, executing contact-based tasks often suffers from low accuracy. We present an approach that addresses these two challenges for the problem of vision-based manipulation. First, we propose to disentangle contact from motion optimization. Thereby, we improve planning efficiency by focusing computation on promising contact locations. Second, we use a hybrid approach for perception and state estimation that combines neural networks with a physically meaningful state representation. In simulation and real-world experiments on the task of planar pushing, we show that our method is more efficient and achieves a higher manipulation accuracy than previous vision-based approaches.**

I. Introduction

In many robotics applications that involve manipulation or legged locomotion, planning contact interactions is one of the core challenges. The main problems are the computational cost of optimization and the uncertainty induced by imperfect perception. Current approaches roughly fall into two categories that align with these problems. The first focuses on reducing the computational cost of motion optimization especially for long sequences and complex dynamics. Such approaches typically make the strong assumption of a fully observable state and prior knowledge of the robot and environment, which are rarely fulfilled in practice. The second category focuses on including perception and being robust to the resulting uncertainty. Approaches in this category are typically learning-based and provide a larger level of generalization to variations of the environment, such as unknown objects. However, this often comes at the cost of accuracy. We propose an approach that addresses both main challenges in planning contacts, imperfect visual perception and the computational complexity of the task. We show that by combining learning-based perception with an explicit state representation, we can achieve accuracy and generalization, while disentangling contact and motion optimization allows for efficient planning.

[1] Max Planck Institute for Intelligent Systems, akloss@tue.mpg.de
[2] Massachusetts Institute of Technology, {bauza, jbt, albertor}@mit.edu
[3] Stanford University, jiajunwu@cs.stanford.edu, bohg@stanford.edu

This research was supported in part by the Max-Planck-Society, the Toyota Research Institute (TRI), and ONR MURI N00014-16-1-2007. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations or any other Toyota entity. The authors thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting Alina Kloss.



(a) Planar Pushing

(b) Test objects

Fig. 1: (a) To push an object to a desired pose (red), a robot has to reason over *where* (green contact points) and *how* (black arrows) to push. (b) The test objects triangle, butter and hexagon.

As a case study, we use quasi-static planar pushing with a point contact. This non-prehensile manipulation primitive can be represented by a simple, low-dimensional state, but has surprisingly complex dynamics, making it difficult to control. Prior work can be split according to the aforementioned two main challenges. Approaches based on analytical models and a physically meaningful state representation often achieve high accuracy, but assume full observability and known object shape [1, 2, 3, 4, 5]. Learning-based approaches address the perception problem and make fewer assumptions about the environment, but are less accurate [6, 7, 8, 9, 10, 11, 12]. Moreover, many of these works do not explicitly reason over where to push, but sample random actions [1, 2, 6, 7, 8, 9]. We argue that explicitly optimizing contact points makes planning more efficient by focusing evaluations on promising regions.

We address the problem of pushing an object to a goal pose given RGBD images, illustrated in Figure 1a. The robot has to decide *where* and *how* to push the object. We use a learned model to capture the shape of the object from the visual input and predict a physically meaningful representation of the object state. This allows us to use filtering to estimate latent variables like the centre of mass of the object online and increase the accuracy of prediction.

Each point on the object outline is densely annotated with approximate predictions of the object motion it affords. This allows sampling promising candidates of *where* to push the object given a desired target object pose. At these contacts, we then optimize *how* to push. For predicting the possible object motion, we compare an approach based on a physical model to a model-free, learned one. The learned model makes fewer assumptions and shows advantages in cases that are not well-captured by the physical model. However, the physics-based model is generally more accurate and even generalizes to scenarios that violate some of its assumptions.

In summary, we propose a system for planar pushing that:

- allows for efficient planning by explicitly reasoning about contact-location,

- improves over model-based approaches by including perception and online estimation of latent object properties,
- achieves higher accuracy than previous vision-based works by combining learned and analytical elements.

We quantitatively evaluate our method in simulation through ablation studies and comparison to state of the art. We also demonstrate that it transfers to a real robotic platform.

II. Related Work

There is a wide range of research on robotic pushing, from modeling the dynamics, e.g. [13, 14, 15, 16], to state estimation for pushed objects [17, 18]. Here, we focus on works that include planning, for a broader review see [19].

A. Efficient Contact Planning under Full Observability

Hogan et al. [3] present a real-time controller for tracking a desired trajectory under full observability. While the push is locally optimized by a neural network that predicts sticking or sliding, the global contact location is not. Zito et al. [1] present an approach to push an object into a desired pose in multiple steps by combining a global RRT planner with a local, sampling based planner. Dafle et al. [4] reorient a known object in-hand by pushing it against elements in the workspace. Similar to our work, they use motion-cones to efficiently describe the set of possible object movements. Ajay et al. [20] use a hybrid approach that augments the predictions from a physical model with learned residuals to push two disks that are already in contact. The method evaluates a predefined set of contacts.

Optimizing contacts is also considered in legged locomotion. Deits and Tedrake [21] compute a sequence of footsteps given a set of obstacle-free regions. For efficiency, the dynamics of the robot are not taken into account. To address this issue, Lin et al. [22] take a similar approach to ours: they train an approximate dynamics model over a discrete set of actions that can be used for efficient contact planning while taking robot dynamics into account.

All these approaches assume full observability and known models of dynamics and geometry.

B. Push Planning under Partial Observability

Agrawal et al. [10] train a network to predict the pushing action required to transform one RGB image into another. In contrast, Li et al. [6], Finn and Levine [7], Ebert et al. [8, 9] do not directly predict actions but learn a dynamics model for predicting the effect of sampled pushes. The input is either a segmentation mask or a full RGB image. Push-Net [6] samples 1000 actions by pairing pixels inside and outside of the object, while [7, 8, 9] sample pusher motions that are refined iteratively. Neither work reasons over contact locations, whereas our approach directly samples promising contact points. Push-Net [6] also estimates the centre of mass of objects during interaction using an LSTM. We rely on an *Extended Kalman Filter* (EKF) that estimates the physically meaningful state representation during interactions. Similar to our work, Hermans et al. [11] learn a scoring function from histogram features for finding suitable contact points. Stüber et al. [12] learn a contact model and a contact-conditioned predictive model for pushing with a mobile robot.

While making much fewer assumptions, these vision and learning-based methods are less accurate than model-based methods. Our method significantly improves on this.

III. Methods

Figure 2 shows an overview of our system. At each time step, it receives an RGBD image of the current scene, the last robot action and the target object pose as input. In the perception module, we use a *Convolutional Neural Network* (CNN) to segment the object and estimate its position and orientation. Since we do not assume prior knowledge of object shape, we extract a representation based on the segmentation map. Together with the last action, the object pose is input to an EKF that estimates the full object state including latent properties like centre of mass (COM).

The next module approximates the object motions that can be produced by applying a discrete set of pushes at each point on the object silhouette. We refer to the output as *push affordances* of the contact points. While this may be considered an abuse of terminology [23], we use the term for a clear distinction to other parts in our model. The affordances are continuously updated as they depend on object properties that are estimated by the EKF, while the object shape has to be computed only once. Finally, the state estimate and affordances are the input to the planning module which selects suitable contact points and optimizes the pushing actions beyond the discrete set that is considered in the affordance model.

A. Planar Pushing

We consider the task of quasi-static planar pushing of a single object using a point contact, where quasi-static means that the force is enough to move but not to further accelerate the object. We parametrize a pushing action by the contact point $\mathbf{r}$ and the pushing motion $\mathbf{u}$. Pushes are executed at a constant velocity of 20 mm/s.

The dynamics of pushing depend on object shape, friction and pressure distribution of the object on the surface. The relation between push force and resulting object motion is often modeled using the limit-surface [24, 25]. We use an analytical model by Lynch et al. [26]. It assumes continuous object-surface contact and an uniform pressure distribution for an ellipsoidal approximation of the limit surface parameterized by l. The model predicts object translation and rotation around the COM given l, the push $\mathbf{u}$, the normal $\mathbf{n}$ at the contact point and the coefficient of friction between pusher and object μ. We use $\mathbf{x} = \left(\mathbf{p}, \theta, \mathbf{c}, l, \mu\right)$ as object state, where θ is the orientation of the object and $\mathbf{c}$ is the position of the COM relative the object frame origin $\mathbf{p}$.

B. Perception and State Estimation

We train a CNN to segment the object in each image and compute its world-frame position from segmentation mask and depth values. A bounding box around the segmentation mask is reprojected into a top-down view centered on the

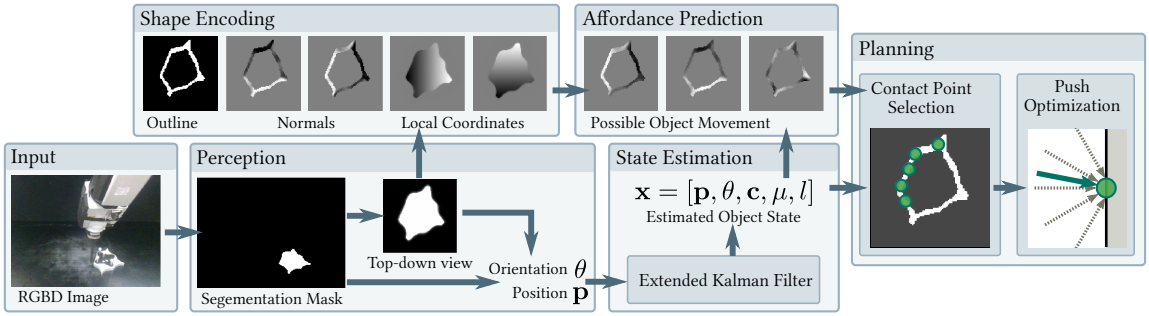


Fig. 2: Overview: the perception module segments the object and computes its pose. An EKF estimates the full object state including latent properties like the COM $\mathbf{c}$. The object shape is encoded by a silhouette, coordinates and normals in a top-down view. It is input to the affordance prediction module, that approximates the possible object motions at each contact point on the silhouette. The planning module selects contact point candidates using the predicted affordances and optimizes the pushing motion there.

object. The orientation of the object is computed relative to the first step by comparing stepwise rotations of the current top-down projection to the initial one. We also evaluated using a neural network for this task but found it to be less reliable. The output object pose is used as observation for an EKF that estimates the full object state $\mathbf{x}$. The filter uses the analytical model [26] as process model and an identity matrix selecting the object pose from $\mathbf{x}$ as observation model.

C. Shape Encoding

Our shape encoding needs to be independent of the object position and contain the necessary information for predicting the effect of pushes, i.e. the possible contact points and the surface normals. We use the object-centric top-down projection of the segmentation mask and depth values to compute the x and y coordinates of each object pixel in this frame. Together with the mask, the coordinates are the input to a CNN that predicts the object outline and the unit 2D surface normals to each point on the outline. Figure 2 shows an example of the resulting $100 \times 100 \times 5$ image (outline, coordinates and normals) under Shape Encoding.

D. Affordance Prediction

For each point on the object outline, the affordance module makes an approximate prediction of the object motions that can be achieved by pushing there. For this, it densely evaluates a predictive model for a fixed set of representative pushing motions. This prediction then informs contact point selection for pushing the object towards the target.

For our experiments, we use a relatively large set of ten representative pushes: we take five directions relative to the respective surface normal ($0°$, $\pm 30°$ and $\pm 60°$) with two lengths each (1 cm and 5 cm). In general, the expressiveness of the affordance model is a tuning parameter of our method that trades off accuracy against computational speed.[1]

We evaluate two predictive models for obtaining the affordances, the analytical model and a learned model.

1) Affordances from the Analytical Model: Given the representative pushes, the shape encoding and parameters $\mathbf{c}$, l and μ from the state estimation module, we can apply the analytical model [26] at each contact point. We use a one-step prediction, which can be done efficiently on GPU but is less accurate than rolling out the model over smaller substeps: During the push, values like the contact point and normal there can change, e.g. when the the pusher slides along the object or even loses contact completely. Such changes of the model's input values cannot be taken into account without substeps.

2) Affordances from a Learned Model: Alternatively, we train a CNN to predict object movement given the pushes, $\mathbf{c}$ and the shape encoding. Different from the analytical model, it does not require the parameters l and μ, but can take the shape around the contact point into account to predict effects of pusher sliding like loss of contact. For this, the model uses a 3-layer CNN with max-pooling to process the object outline. The resulting local shape features, the pushes and the shape encoding serve as input for predicting the object motion using a second 3-layer CNN without pooling.

E. Planning

We use a greedy planner to find the contact point and straight pushing motion that brings the object closest to the desired goal pose at each step. We found this approach to be sufficient in our scenario where no obstacles are present. For planning around obstacles, our model could be combined with a global planner for object poses, e.g. [1, 4].

Instead of jointly optimizing contact point and pushing motion, we divide the problem into two subtasks. We first propose a set of contact points and then separately optimize the pushing motions at each candidate point before selecting the most promising combination.

1) Contact Point Proposal: Our method uses the affordances to score each point on the object outline by how close pushing there could bring the object to the target pose:

$$s(\mathbf{r}_i) = \min_{\mathbf{u} \in U_i} \| \mathbf{v}_d - \mathbf{v}_p(\mathbf{u}, r_i) \|_2 + \lambda |\dot{\theta}_d - \dot{\theta}_p(\mathbf{u}, r_i)| \quad (1)$$

Here, $\mathbf{v}_d$ and $\dot{\theta}_d$ are the desired object translation and rotation, U_i are the representative pushing motions at contact point $\mathbf{r}_i$, and $(\mathbf{v}_p(\mathbf{u}, r_i),\ \dot{\theta}_p(\mathbf{u}, r_i))$ their predicted object motion. We weight the rotation error (in degree) stronger ($\lambda = 2$) for a good trade-off between translation and rotation. A softmax function turns the scores, $s(\mathbf{r}_i)$, into a probability distribution that is used to sample k candidate points. We

[1] Ablation studies (not included for space) showed that including fewer pushes has an overall small effect. Removing the 5 cm pushes was worst, increasing the average number of steps taken by more than 15 %.

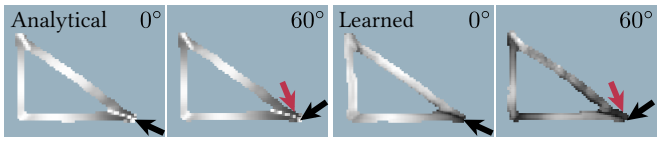


Fig. 3: Predicted translation magnitude from the analytical and learned affordance model (brighter is higher) for pushes along the normal and at a 60° angle. In contrast to the analytical model, the learned model predicts low magnitude for pushes that are unlikely to properly hit the object (black arrows) or pushes that will slide off the object (red arrows).

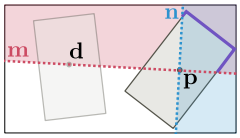


Fig. 4: Heuristic for contact point selection (*geo*): line $\mathbf{m}$ connects the current $\mathbf{p}$ and desired object position $\mathbf{d}$, $\mathbf{n}$ is its normal. Points on the blue side of $\mathbf{n}$ afford pushing towards $\mathbf{d}$, points to the right of $\mathbf{m}$ (red area) are proposed for counter-clockwise rotation. For rotations below 2°, candidates need to lie within 2 cm of $\mathbf{m}$. The intersection of both areas (purple) defines the set of possible contact points for sampling.

found that sampling the contact points instead of choosing the k best points improved the robustness of our method.

2) Push Motion Optimization: The discrete set of actions evaluated for the affordance model will in general not contain the optimal pushing motion at each point. We thus optimize push direction and length at each candidate contact point by interpolating between five base pushes U_b with different directions as follows:

1) For each $\mathbf{u_b} \in U_b$, roll out the analytical model over the max. push length of 5 cm in substeps of 0.5 cm.
2) At each step, score the predicted object movement so far using Equation 1.
3) Truncate each $\mathbf{u_b}$ at its best-scoring step (min. push length 1 cm). This gives $\bar{\mathbf{u}}_\mathbf{b}$ with optimal scores $\bar{s}_b$.
4) Find the optimal push direction $\mathbf{u_d}$ by interpolating between the $\bar{\mathbf{u}}_\mathbf{b}$ with the best $\bar{s}_b$ and the two $\bar{\mathbf{u}}_\mathbf{b}$ with neighbouring directions.
5) Optimize the length of $\mathbf{u_d}$ as in steps (1) - (3) to find the optimal push $\mathbf{u}^*$ with score s^*

As explained before, rolling out the analytical model over shorter substeps is more accurate than predicting the outcome of the full 5 cm push in one step. The planner finally returns the contact point and action with the highest s^*.

In our specific case, the affordances already contain predictions for the same five push directions that we also use for U_b. This allows us to use the affordance predictions in step (1) of the push optimization. Instead of steps (2,3) and (5), the push length is then optimized by linearly rescaling the push and prediction to match the desired motion. We compare this to our regular method in Experiment V-D.

IV. Training

For training the perception, shape encoding and learned affordance model, we rely mostly on simulated data generated in pybullet [27]. Each datapoint contains an RGBD image of an object on a surface, its ground truth position, segmentation mask and outline with normals. We annotate 20 random contact points per object with the object movement in response to the ten representative pushing actions defined in Section III-D. Properties like object mass, centre of mass and friction coefficients are sampled randomly. We generate more than 15k examples using 21 objects, of which we hold out three for testing (shown in Figure 1b, which also shows the real-world setup after which we modeled the simulation). While the segmentation and shape encoding network are finetuned on real data from the Omnipush dataset [28], the learned affordance model is only trained on simulated data. We train the models in tensorflow [29] using Adam [30].

V. Simulation Experiments

A. Setup

We evaluate three different tasks: translating the object by 20 cm without changing the orientation (*translation*), rotating the object by 0.5 rad (28.6°) without changing the position (*rotation*) and translating for 10 cm plus rotating by 0.35 rad (20°) (*mixed*). A trial counts as successful if it brings the object within less than 0.75 cm of the desired position and 5° of the desired orientation in at most 30 steps. We evaluate the percentage of successful trials and the average number of steps until the goal pose is reached.

For each task, object and method, we perform 60 trials. At the beginning of each, the object is placed at the center of the workspace. We vary its initial orientation in 20 steps from 0 to 360° and perform three runs with each orientation.

B. Affordance Prediction

We first qualitatively compare the learned and the analytical affordance model to see if there are any major differences between them. Overall, both models predict similar directions of movement, with the analytical model predicting more pronounced rotation. A potential advantage of using a learned model becomes apparent when we compare the magnitude of the predicted translational movement, which is shown in Figure 3. The analytical model predicts strong translation for pushes to the sharp corners of the triangle, whereas the learned model predicts comparatively low magnitudes there. The same effect is visible for angled pushes that cause the pusher to slide towards corners. As discussed before, the analytical affordance model cannot predict a loss of contact due to pusher sliding. However, this is more likely when pushing at sharp corners or with high angles. The learned model takes the object shape around the contact point into account and is therefore better at predicting such cases.

C. Contact Point Selection

In this experiment, we test our hypothesis that explicitly reasoning about the contact locations makes planning more efficient as compared to sampling actions that collide with the object in random locations. For this, we vary the number of sampled contact points and compare our approach (that uses the affordances to propose promising contact points) to two baselines that select the contact points more randomly.

The simplest baseline samples uniformly from all points on the object outline (*rdn*). A more informed approach (*geo*) uses a geometric heuristic explained in Figure 4. Based on

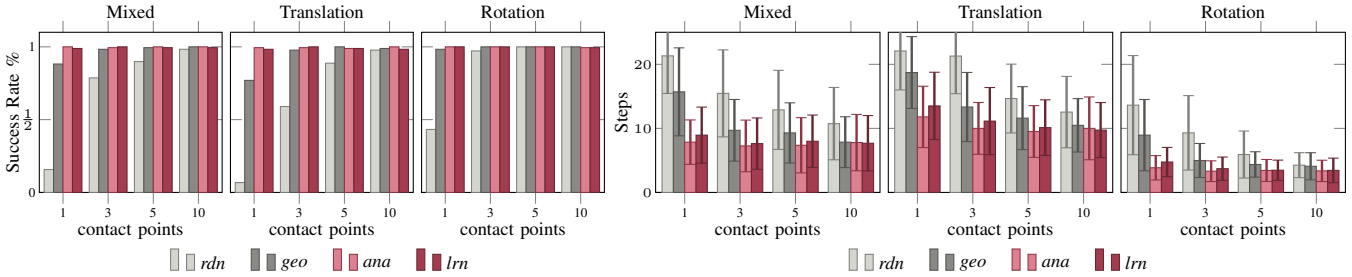


Fig. 5: Pushing performance over number of sampled contact points. We compare different sampling methods of contact locations: randomly (*rdn*) or according to the geometric baseline (*geo*), analytical (*ana*) or learned (*lrn*) affordances. We analyse performance for three different tasks: Pure object translation, pure object rotation and a mixed motion. Results are averaged over three test objects (See Fig. 1b). Our proposed affordance model (either *ana* or *lrn*) generally requires only one contact point sample to achieve a high success rate with the lowest number of steps to get to a target object pose.

the desired motion, it defines a quadrant of the object from which the contact points are sampled. In contrast to *rdn*, *geo* better avoids sampling contact points at which the object can only be pushed away from the goal. It however ignores the exact shape of the object and can thus still propose unsuitable contact locations especially for non-convex objects.

To minimize the influence of other components of our system on the results, we do not use filtering for state estimation in this experiment but assume access to perfect state information at every step.

Results: We first compare the success rates in Figure 5 (left). By sampling from the affordance model (learned *lrn* or analytical *ana*), our method can already achieve a success rate close to 100% with only one contact point. The geometric heuristic also performs well and often reaches 100% with as few as three contact points. We only see a big impact of the number of contact points when sampling randomly. On the tasks that involve translation, *rdn* only reaches the success rate of the other methods with ten contact points. The number of sampled points is generally more important for translating than for rotating.

Figure 5 (right) shows the number of steps each method took until the goal pose was reached. Even in successful runs, *rdn* needs significantly more steps than the other methods. *Geo* again performs better, although still worse than our proposed method using the affordance prediction. Both *lrn* and *ana* work very well with only one contact point and their performance mostly saturates at three sampled candidates. There is no significant difference between using the analytical or the learned model for obtaining the affordances.

To summarize, using an affordance model to sample contact points makes planning more efficient by reducing the number of contact points that have to be evaluated per step and the number of steps taken until the goal is reached.

D. Pushing Motion Optimization

In this experiment, we test if the predicted affordances are accurate enough to also be used for optimizing the pushing actions (see Section III-E.2). This is especially interesting for evaluating the quality of the learned model. We call the variants that use the predictions from the affordance model directly *ana direct* and *lrn direct* respectively.

Results: As shown in Figure 6, using the analytical affordance predictions does not significantly increase the number

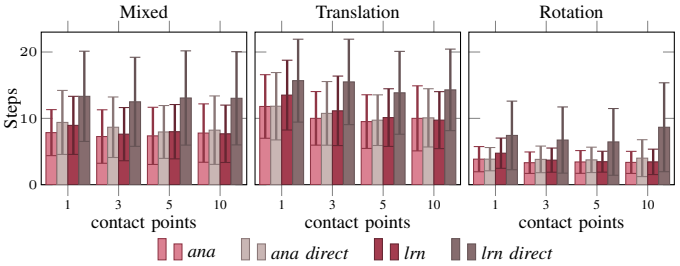


Fig. 6: Steps taken vs. sampled contact points when rolling out the analytical model for optimizing the push motions (*ana, lrn*) or directly using the affordance (*ana-direct, lrn-direct*). While *ana, lrn* and *ana-direct* perform similar, *lrn-direct* is less accurate and thus needs more steps to succeed.

of steps as compared to *ana* which optimizes the actions by rolling out the analytical model over smaller substeps. When using the learned affordances for push optimization, the number of steps increases by up to four and the success rate drops by up to 10% compared to *lrn*. This implies that while being sufficient for selecting contact points, the learned model is not as accurate as the analytical model for predicting the outcome of a push. The negative effect of using the learned model is also not compensated by evaluating more contact points, which emphasizes the value of an accurate predictive model for optimizing the pushes.

E. Full System

Now we evaluate the accuracy of our full system including the state estimation module, with three contact points sampled per step. In all experiments, $\mathbf{c}$ is initialized to zero and l and μ to reasonable estimates. We first test on objects whose centre of mass coincides with the geometric center to evaluate the perception module and how well planning works with imperfect pose information. In the second experiment, we verify the benefit of estimating latent properties of the object on the example of the COM. For this, we sample the COM uniformly inside the objects. We also compare to Push-Net [6] under this condition. Push-Net uses top-down segmentation maps of the current and desired pose as input to evaluate randomly sampled actions. A local planner generates sub-goals by interpolating between the current and the goal pose with a fixed step size, we use 5 cm and 10°.

Results: In the previous experiments, we used ground truth object pose information. Here, we compare those results to doing pose estimation by filtering. We find that using the filter has no impact on the success rate or the number of

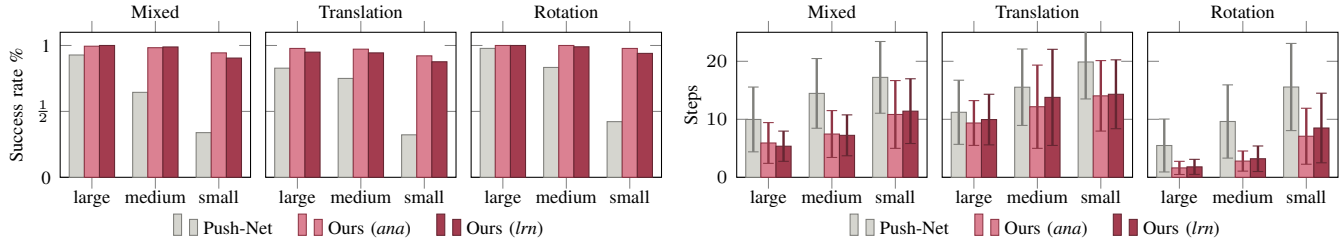


Fig. 7: Performance of our method and Push-Net on objects with random COM (averaged over objects). We evaluate three goal region sizes, *small* (0.75 cm 5°), *medium* (2.5 cm 7.5°) and *large* (5 cm 10°). Our method has a higher success rate on smaller goal regions and needs fewer steps to reach the goal.

steps taken. However, it increases the (true) average end pose error from 5.0±1.8 mm, 1.8±1.3° to 8.7±4.2 mm and 3.0±2.2°. This is expected as we use the *estimated* object pose to determine if the goal is reached. Therefore, the real pose error can be higher than the (7.5 mm, 5°) margin of the goal region.

On objects with a randomly sampled COM, we first verify that estimating the COM position is beneficial, by comparing to a variant that assumes a fixed COM. For the triangle and butter shape, the average estimation error for $\mathbf{c}$ is 17.7±8.5 mm, on the hexagon it is around 1 cm higher. The average distance of $\mathbf{c}$ to $\mathbf{p}$ is 37.4±12.3 mm. Despite not being extremely accurate, estimating $\mathbf{c}$ increases the success rate by up to 5% depending on the task and object. The number of steps is also lower, but not significantly, most likely because estimating the COM takes a few steps in which the object often does not move towards the goal.

We also compare our approach to Push-Net, which uses an LSTM to estimate the COM. We evaluate three sizes of the goal region, from the (0.75 cm, 5°) we used in all previous experiments to the (5 cm, 10°) used in the original Push-Net paper, plus a medium size of (2.5 cm, 7.5°). Results are shown in Figure 7. With the largest goal region, Push-Net performs competitive to our approach and it still reaches a good success rate for the medium sized region. On the smallest size however, our approach outperforms Push-Net by a large margin, despite evaluating much fewer actions. Our method also requires less steps to reach each level of accuracy. Qualitatively, Push-Net does well for translating the object, but has trouble controlling its orientation precisely.

VI. Real-Robot Experiments

We also evaluate our approach on a real system (ABB IRB 120 industrial robotic arm, Intel RealSense D415 camera, see Fig. 1a). This is especially interesting with respect to our predictive models: We know that the analytical model makes assumptions that are frequently violated in the real world, while the learned model was trained purely in simulation and might not transfer well to the real world.

1) Setup: To evaluate our affordance models, we first compare *lrn*, *ana* and *lrn direct* given ground truth state information on the butter object from the MIT Push Dataset [31] that we also used in the simulation experiments (see Fig. 1b). Then we test the full system with the analytical affordance model on butter, triangle and a new object from the Omnipush Dataset [28] (shown in Fig. 1a) This objects has weights to change its pressure distribution and centre of mass and thus violates the uniform pressure distribution assumption of the analytical model. For both experiments, we use a new task (12 cm translation, 46° rotation, maximum 20 steps). Every experiment is repeated 15 times.

2) Results: When comparing *lrn*, *ana* and *lrn direct* on the real butter object, the results are very similar to the simulation experiments. Using the analytic push optimization step, both *lrn*, *ana* succeed in all trials and need 5.5±2.6 and 5.2±2.1 steps respectively. For *lrn direct*, the average number of steps increases to 8.2±4.3.

With filtering, we achieve an average success rate of 97% on triangle and butter with an end pose error of 9.2±4.2 mm, 5.8±2.1° in 6.7±3.5 steps. For the omnipush object, the orientation estimation sometimes fails, dropping the success rate to 89% and increasing the number of steps to 8.0±3.6. We still reach an average end pose error of 8.8±4.5 mm, 7.2±5.5°, confirming that our approach generalizes to conditions that violate the assumptions of the analytical model.

VII. Conclusion

We presented an approach for vision-based manipulation that uses an affordance model for selecting contact points during planning. Our experiments on planar pushing show that by explicitly reasoning over contact locations, we evaluate less actions and plan more optimal pushing actions than when sampling random contact locations. Our method also reaches a higher accuracy than previous vision-based work by relying on a physically meaningful state representation.

By comparing a learned and an analytical predictive model, we show that for selecting contact locations, approximate predictions are sufficient. However, to optimize the pushing motion at each contact point, the higher accuracy of the analytical model proved to be important.

We thus find that using a hybrid approach - combining learned and analytical components - is beneficial for robotics. Learning is not only well suited for perception but also for "intuitive physics" models that can quickly narrow down large search spaces to few promising candidates that are then optimized using more accurate but costly analytical models.

Limitations are the simple scenes we consider and that our method assumes a mostly unoccluded object outline. Dealing with strong occlusion is an interesting problem for future work. Accurately estimating object orientation also proved challenging in some cases. Finally, we would like to test on more diverse, non-planar objects. Preliminary results in simulation suggest that our approach is robust to this, but real-world experiments are necessary to confirm these results.

References

[1] C. Zito *et al.*, "Two-level rrt planning for robotic push manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[2] W. C. Agboh, D. Ruprecht, and M. R. Dogar, "Combining coarse and fine physics for manipulation using parallel-in-time integration," *arXiv preprint arXiv:1903.08470*, 2019.

[3] F. R. Hogan, E. R. Grau, and A. Rodriguez, "Reactive planar manipulation with convex hybrid mpc," in *IEEE International Conference on Robotics and Automation*, 2018.

[4] N. C. Dafle, R. Holladay, and A. Rodriguez, "In-hand manipulation via motion cones," in *Robotics: Science and Systems*, 2018.

[5] M. Bauza, F. R. Hogan, and A. Rodriguez, "A data-efficient approach to precise and controlled pushing," in *Conference on Robot Learning*, 2018.

[6] J. Li, W. S. Lee, and D. Hsu, "Push-net: Deep planar pushing for objects with unknown physical properties," in *Robotics: Science and Systems*, 2018.

[7] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *IEEE International Conference on Robotics and Automation*, 2017.

[8] F. Ebert *et al.*, "Self-supervised visual planning with temporal skip connections," in *Conference on Robot Learning*, 2017.

[9] F. Ebert *et al.*, "Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning," in *Conference on Robot Learning*, 2018.

[10] P. Agrawal *et al.*, "Learning to poke by poking: Experiential learning of intuitive physics," in *Advances in Neural Information Processing Systems*, 2016.

[11] T. Hermans *et al.*, "Learning contact locations for pushing and orienting unknown objects," in *IEEE-RAS International Conference on Humanoid Robots*, 2013.

[12] J. Stüber, M. Kopicki, and C. Zito, "Feature-based transfer learning for robotic push manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 5643–5650.

[13] J. Zhou *et al.*, "A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation," *The International Journal of Robotics Research*, vol. 37, no. 2-3, pp. 249–265, 2018.

[14] K. M. Lynch, "Locally controllable manipulation by stable pushing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 318–327, 1999.

[15] M. Bauza and A. Rodriguez, "A probabilistic data-driven model for planar pushing," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3008–3015.

[16] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects," *arXiv preprint arXiv:1710.04102*, 2017.

[17] K.-T. Yu and A. Rodriguez, "Realtime state estimation with tactile and visual sensing. application to planar manipulation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7778–7785.

[18] A. S. Lambert *et al.*, "Joint inference of kinematic and force trajectories with visuo-tactile sensing," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3165–3171.

[19] J. Stüber, C. Zito, and R. Stolkin, "Let's push things forward: A survey on robot pushing," *arXiv preprint arXiv:1905.05138*, 2019.

[20] A. Ajay *et al.*, "Combining physical simulators and object-based networks for control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[21] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *IEEE-RAS International Conference on Humanoid Robots*, 2014.

[22] Y.-C. Lin *et al.*, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *IEEE International Conference on Robotics and Automation*, 2019.

[23] F. Osiurak, Y. Rossetti, and A. Badets, "What is an affordance? 40 years later," *Neuroscience & Biobehavioral Reviews*, vol. 77, pp. 403 – 417, 2017.

[24] R. D. Howe and M. R. Cutkosky, "Practical force-motion models for sliding manipulation," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 557–572, 1996.

[25] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction part 1. limit surface and moment function," *Wear*, vol. 143, no. 2, pp. 307 – 330, 1991.

[26] K. M. Lynch, H. Maekawa, and K. Tanie, "Manipulation and active sensing by pushing using tactile feedback," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1992.

[27] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016.

[28] M. Bauza *et al.*, "Omnipush: accurate, diverse, real-world dataset of pushing dynamics with rgb-d video," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019.

[29] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[30] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] K.-T. Yu *et al.*, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.