PatchGraph: In-hand tactile tracking with learned surface normals

Paloma Sodhi[1,2], Michael Kaess[1], Mustafa Mukadam[2], and Stuart Anderson[2]

[1]Carnegie Mellon University, [2]Meta AI Research

***Abstract*— We address the problem of tracking 3D object poses from touch during in-hand manipulations. Specifically, we look at tracking small objects using vision-based tactile sensors that provide high-dimensional tactile image measurements at the point of contact. While prior work has relied on a-priori information about the object being localized, we remove this requirement. Our key insight is that an object is composed of several local surface patches, each informative enough to achieve reliable object tracking. Moreover, we can recover the geometry of this local patch online by extracting local surface normal information embedded in each tactile image. We propose a novel two-stage approach. First, we learn a mapping from tactile images to surface normals using an image translation network. Second, we use these surface normals within a factor graph to both reconstruct a local patch map and use it to infer 3D object poses. We demonstrate reliable object tracking for over 100 contact sequences across unique shapes with four objects in simulation and two objects in the real-world.**

I. Introduction

We focus on the problem of tracking 3D object poses during in-hand manipulations using tactile image measurements from vision-based tactile sensors [1, 2]. Specifically, we look at tracking small objects without prior geometric models. For instance, a dexterous robot operating in a real-world household environment will need to manipulate novel household objects for which CAD models may not be available. We address the question: Can an object be tracked precisely enough for in-hand manipulation using only local measurements of its geometry?

Prior work has looked at the object tracking problem primarily in the context of planar pushing [3–6]. However, the problem of 3D in-hand manipulation poses additional challenges. Firstly, the motion is less constrained such that different object motions can explain the same measurements. Second, physics priors, such as quasi-static planar pushing models, are less informative in the 3D case. Hence, prior work on in-hand object tracking using tactile feedback has relied on a-priori information about the object being localized, such as 3D global models or a database generated by a simulator [7–10].

Our key insight is that the tactile object tracking problem can be efficiently decomposed in two ways. First, we can decompose an object into many smaller local surface patches, which can be treated independently. Second, most of the information needed to infer the local surface patch geometry is already embedded in corresponding tactile images.

Code and supplementary material can be found on `https://psodhi.github.io/tactile-in-hand`

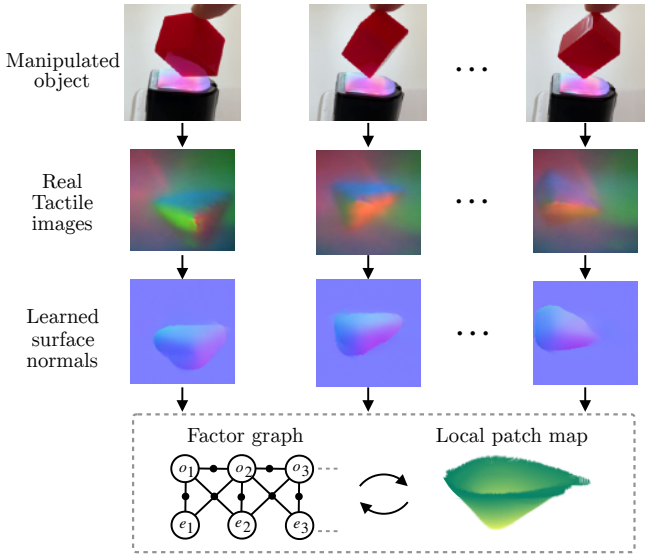

Fig. 1: Tracking latent 3D object poses from a tactile image sequence during in-hand manipulation. We solve this as an inference over a factor graph that does not rely on having prior object models.

We find that reliable tracking is achievable with only a *local patch*—a fused map created from a sequence of key frame images within a continuous contact episode. For instance, even though two objects can have very different global geometries, they can contain very similar local patches that suffice for tracking.

To both create the local patch and track motion relative to it, we must fuse multiple tactile image measurements online while inferring the latent object poses. We formulate this as an inference problem over a factor graph that offers a flexible and efficient way to fuse such information while incorporating other priors derived from physical and geometric constraints [11, 12].

What makes for a good representation for a local patch? An idealized tactile image captures the surface normals of the gel's reflective layer, based on the color and intensity of illumination at each pixel. Hence, it is natural to learn a mapping from image to gel surface normals. While objects may have varying global shapes, the learned surface normal mapping can generalize across these different shapes, because the relationship between pixel intensities and surface normals depends only on the sensor configuration and the local contact geometry. Hence, we infer the surface normals at each pixel in the tactile image, then integrate those normals to create a 3D model of the visible section of the local patch.

We propose a novel two-stage approach for tracking objects in-hand without any prior object model information

(Fig. 1). First, we learn a mapping from tactile images to surface normals using an image translation network. Second, we use these surface normals within a factor graph to both reconstruct a local patch map and use it to infer 3D object poses. Our key contributions are:

1) A factor graph formulation for 3D in-hand tactile tracking that does not rely on prior object models.
2) A factor that works across different global object shapes by relying on local patches generated from learned surface normals.
3) Empirical evaluation on both simulation and real-world trials.

II. Related Work

Factor graphs for localization Localization and mapping problems are increasingly formulated as optimization objectives that leverage the inherent sparsity of the problem to give tractable and more accurate solutions over filtering approaches [13]. Factor graphs are a popular way for solving such optimization objectives [3, 11, 12, 14, 15]. They offer a flexible way to fuse multiple measurements while being computationally efficient to optimize. Factors in the graph encode local potentials on variables such as observation models between measurements and states as well as other priors such as physics and geometry. We formulate our problem using a factor graph based framework in this paper.

Vision-based touch sensing The advent of vision-based tactile sensors [1, 2, 16, 17] has enabled high-dimensional tactile image measurements that capture the local deformation at the point of contact. Recent work has also looked at creating accurate simulation models for such sensors [18–20]. These sensors are being explored for various tactile manipulation applications. One class of approaches use tactile images directly as local feedback to solve for control actions on tasks such as object insertion [21], box packing [22], and in-hand manipulations [1, 23]. However, such representations tend to overfit to the particular task or require significant amount of data to generalize across tasks. Our work focuses on extracting a state representation like the global object pose that is easy to use and generalizes across different downstream control and planning tasks.

Estimation from touch Prior work on estimating states from touch during manipulation has included filtering methods [24–27], learning-only methods [28, 29], methods utilizing prior model information [8–10], and graph-based optimization for planar pushing [3–6]. In particular, graph-based optimization offers benefits such as being more accurate than filtering, an ability to incorporate analytic as well as learned models, and can be solved in real-time making use of efficient, incremental solvers [30–32] in literature.

Of these different approaches, the work in [7–10] is most closely related in terms of the final objective of tracking 3D object poses during in-hand manipulations.These, however, require prior object model information either as offline models [7, 9, 10] or a database from a simulator [8]. In contrast, we do not require a prior model of the object being tracked. Instead, we build a local patch map on the fly for the current contact episode and use that within a factor graph framework.

III. Problem Formulation

We begin by formalizing the estimation problem as factor graph optimization. A factor graph is a bipartite graph with two types of nodes: variables $x \in \mathcal{X}$ and factors $\phi(\cdot) : \mathcal{X} \to \mathbb{R}$. Variable nodes are the latent states to be estimated, and factor nodes encode potentials on these variables from observations, physics, or geometry (Fig. 2).

We solve for the maximum a posteriori (MAP) objective $\hat{x}$ by maximizing product of all factor graph potentials, i.e.,

$$\hat{x} = \underset{x}{\operatorname{argmax}} \prod_i \phi_i(x_i) \tag{1}$$

To solve this inference objective efficiently, we assume $\phi_t(x)$ to be Gaussian factors corrupted by zero-mean normally distributed noise. Under Gaussian noise model assumptions, MAP inference is equivalent to a nonlinear least-squares objective [12], i.e.,

$$\begin{aligned} \phi_i(x_i) &\propto \exp\left\{-\frac{1}{2}||f_i(x_i; z_i)||^2_{\Sigma_i}\right\} \\ \Rightarrow \hat{x} &= \underset{x}{\operatorname{argmin}} \frac{1}{2}\sum_i ||f_i(x_i; z_i)||^2_{\Sigma_i} \end{aligned} \tag{2}$$

For the in-hand object tracking problem, we define states in the graph to be the 6-DOF object and end-effector poses at every time step $t = 1 \dots T$, i.e. $x_t = [o_t \ e_t]^T$, where $o_t, e_t \in SE(3)$. Factors in the graph include image-to-image factors $f_{im2im}(\cdot)$, image-to-patch factors $f_{im2pc}(\cdot)$, velocity smoothness priors $f_{vel}(\cdot)$, end-effector pose priors $f_{eff}(\cdot)$ and vision priors for re-localization at the beginning of a contact episode $f_{vis}(\cdot)$. At every time step, new variables and factors are added to the graph. Writing out Eq. 2 for our problem,

$$\begin{aligned} \hat{x}_{1:T} = \underset{x_{1:T}}{\operatorname{argmin}} \sum_{t=1}^{T} \{ \ &||f_{im2im}(o_{t-1}, o_t, e_{t-1}, e_t)||^2_{\Sigma_{im2im}} + \\ &||f_{im2pc}(o_t, e_t)||^2_{\Sigma_{im2pc}} + ||f_{vel}(o_{t-2}, o_{t-1}, o_t)||^2_{\Sigma_{vel}} + \\ &||f_{eff}(e_t)||^2_{\Sigma_{eff}} + ||f_{vis}(o_t)||^2_{\Sigma_{vis}} \ \} \end{aligned} \tag{3}$$

Individual cost terms in Eq. 3 are described in detail in Section IV-C. Eq. 3 is the optimization objective that we must solve for every time step. Instead of resolving from scratch every time step, we make use of efficient, incremental solvers [31] to solve this in real-time.

IV. Approach

We present a two-stage approach: First, we learn a mapping from tactile images to surface normals (Section IV-A) which can be integrated to create a 3D reconstruction (Section IV-B). Second, we use these surface normals to create a 3D local patch map online within a factor graph for inferring the latent 3D object poses (Section IV-C).

A. Learning surface normals

Here we discuss how we learn to predict surface normals using tactile color images from the Digit sensor [1]. Color

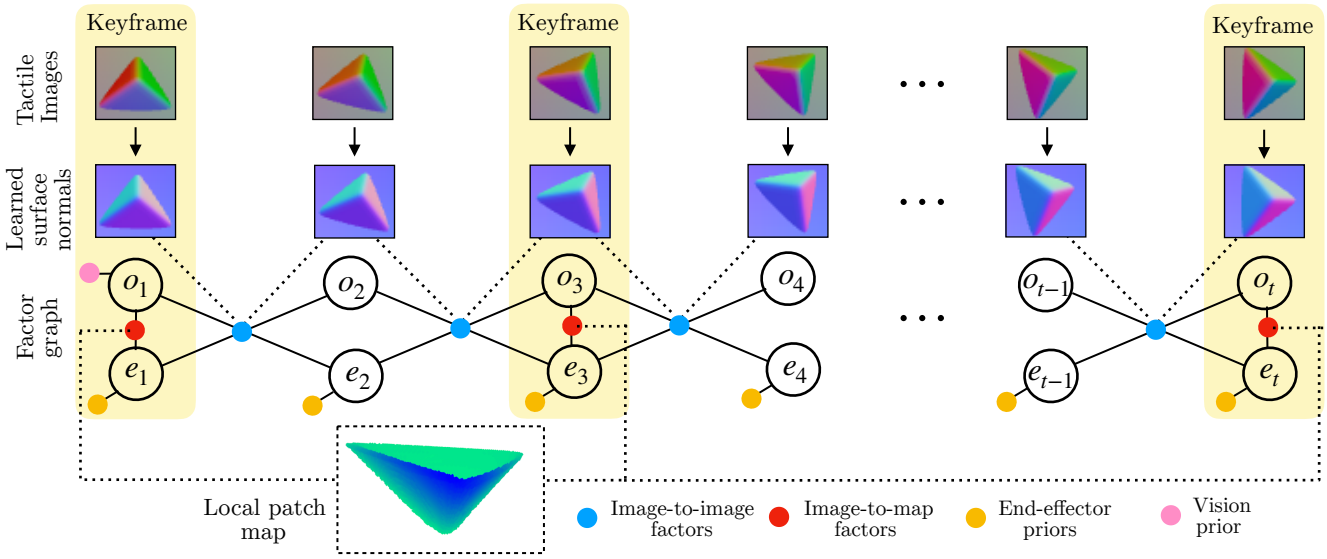

Fig. 2: Overview of our factor graph formulation, where variables are object poses and factors are constraints on poses. We first predict learned surface normals from tactile images used to reconstruct a 3D point cloud. We then create a set of factors. Image-to-image factors (blue) encode relative pose between two point clouds. Image-to-patch factors (red) encode relative pose between point cloud and a fused local patch. We also include global pose priors as unary factors on end-effector (yellow) and first object pose (pink).

images are generated by illuminating the gel surface with three light sources (Fig. 3(a)), such that each light has a unique color and direction. When pressing an object into the gel, the gel surface conforms to the object. Under an idealized model of the sensor, the color and intensity of light reaching the camera due to diffused reflection from a point on the surface are directly related to the surface normal of the gel at that point. Hence, the RGB intensity values of image pixel are expected to contain significant information about the corresponding local surface normals of the object.

To infer surface normal images from tactile images we train an image translation network, pix2pix [33]. The pix2pix model is based on a generator-discriminator network architecture that enables it to learn mappings from a low amount of training data. It also enables us to learn a generalized mapping, i.e. we test on objects unseen during training. To train the model, we use a dataset of ground truth pairs of color and surface normal images $z = \{\mathcal{I}_c, \mathcal{I}_n\}$. We extract datasets in two ways. For simulation dataset, we generate surface normals by adding a normal shader to the Tacto simulator [18] based on Pyrender [34]. For the real dataset, we follow a similar procedure as [2, 7] of using a ball bearing of known radius whose ground truth normals can be synthesized. We manually annotate the circular patches in RGB images, and synthesize ground truth normals in the foreground annotations. We also augment our dataset with additional simulated images to improve generalization.

B. Reconstruction from normals

Given an image with local surface normal information, we generate a 3D representation of the corresponding surface geometry. This surface normal image $N(x, y)$ is related to the gradient of a corresponding depth image $z(x, y)$ as,

$$\frac{\partial z}{\partial x} = \frac{n_x}{n_z}, \qquad \frac{\partial z}{\partial y} = \frac{n_y}{n_z} \tag{4}$$

Given depth gradients $\{\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}\}$, we can recover the depth map $z(x, y)$ by integration using a fast Poisson solver with discrete sine transform (DST) [35] as used in prior work [2, 7]. For the boundary conditions, we use the mean distance to the undisturbed gel surface. A consequence of this choice is that contact regions crossing the edge of the image will not be correctly handled by the solver. We filter out such images in our trials.

Finally, once we have computed the depth map, we inverse project it to obtain a 3D point cloud. We use the OpenGL clip projection model [36] to map from pixel to world coordinates, i.e. $\begin{bmatrix} x_w & y_w & z_w \end{bmatrix} = VP \begin{bmatrix} x_{pix} & y_{pix} & z_{depth} \end{bmatrix}$, where P is the projection matrix based on near, far plane values of the projection frustum, and V is the camera view matrix.

C. Factor graph optimization

Once we have the surface normal images, we integrate those along with other priors as factors within a factor graph. The factor graph optimizer then solves for the joint objective in Eq. 3. We look at each of the cost terms in Eq. 3 in detail.

Image-to-image factors: In some cases, it suffices to look at consecutive tactile images and infer the relative transformation between them. Color images from the tactile sensor at the current and previous time step are converted into point clouds $\{\mathcal{P}_{t-1}, \mathcal{P}_t\}$ in their respective end-effector or sensor frame. The two point clouds are registered against each other using a point-to-plane iterative closest point (ICP) algorithm. The resultant relative transformation is added as a binary factor between consecutive poses in the graph. This is expressed as the $f_{im2im}(.)$ term in Eq. 3, i.e.,

$$||f_{im2im}(o_{t\text{-}1}, e_{t\text{-}1}, o_t, e_t)||^2_{\Sigma_{im2im}} := ||T^{graph}_{t\text{-}1,t} \ominus T^{reg}_{t\text{-}1,t}||^2_{\Sigma_{im2im}} \tag{5}$$

where $T^{graph}_{t\text{-}1,t} = (e_{t\text{-}1} o^{-1}_{t\text{-}1}) \ominus (e_t o^{-1}_t)$ are current relative estimates from the graph and $T^{reg}_{t\text{-}1,t} = \text{argmin}_T \sum_i ||p^{(i)}_t -$

$Tp_{t\text{-}1}^{(i)}||_2^2$ is the measured transformation from ICP registration. Both $T_{t\text{-}1,t}^{graph}$, $T_{t\text{-}1,t}^{reg}$ use gel center as their origin. $(p_{t-1}^{(i)}, p_t^{(i)}) \in (\mathcal{P}_{t-1}, \mathcal{P}_t)$ are pairs of point correspondences in the two point clouds. $\ominus$ denotes difference between two SE(3) manifold elements.

Image-to-patch factors: Image-to-image factors fail whenever the tactile image changes by a non-trivial amount leading to large registration errors. This happens whenever the object undergoes a larger transformation or moves in and out of the gel. To stabilize tracking in these situations we introduce a local patch model by fusing together multiple point clouds and register new tactile point cloud data against this patch. The local patch is maintained by fusing together images at specific key frames within the current contact episode, rather than fusing all available frames. We choose these key frames at fixed intervals given uniform motions but one can also select these based on a field-of-view overlap threshold. The current cloud $\mathcal{P}_t$ is registered against the local patch map cloud $\mathcal{P}_{map}$, and the relative transformation is added as factors to the graph,

$$||f_{im2pc}(o_t, e_t)||^2_{\Sigma_{im2pc}} := ||T_{map,t}^{graph} \ominus T_{map,t}^{reg}||^2_{\Sigma_{im2pc}} \quad (6)$$

where $T_{map,t}^{graph} = o_t e_t^{-1}$ are current estimates from the graph and $T_{map,t}^{reg} = \text{argmin}_T \sum_i ||p_t^{(i)} - Tp_{map}^{(i)}||_2^2$ is the measured transformation from ICP registration. $(p_t^{(i)}, p_{map}^{(i)}) \in (\mathcal{P}_t, \mathcal{P}_{map})$ are pairs of point correspondences in the current point cloud and local patch map.

Constant velocity priors: We add a prior that assumes objects move at a constant velocity, which has the effect of smoothing tracked trajectories. This is a ternary factor between triplets of object poses,

$$||f_{vel}(o_{t\text{-}2}, o_{t\text{-}1}, o_t)||^2_{\Sigma_{vel}} := ||o_{t\text{-}2}^{-1} o_{t\text{-}1} \ominus o_{t\text{-}1}^{-1} o_t||^2_{\Sigma_{vel}} \quad (7)$$

End-effector priors: We model uncertainty about end-effector locations as unary priors on end-effector variables,

$$||f_{eff}(e_t)||^2_{\Sigma_{eff}} := ||e_t \ominus \tilde{e}_t^{mc}||^2_{\Sigma_{eff}} \quad (8)$$

where, $\tilde{e}_t^{mc} = e_t^{mc} \oplus \mathcal{N}(0, \Sigma_{eff})$ are poses from the motion capture system with added Gaussian noise. We model the end-effectors as variables in the graph to keep the formulation general, especially when we have robot end-effector, the pose measurements would be coming from robot kinematics.

Vision prior: We add a global vision pose prior for only the object pose at the start of an episode,

$$||f_{vis}(o_t)||^2_{\Sigma_{vis}} := ||o_t \ominus \tilde{o}_t^{vis}||^2_{\Sigma_{vis}} \quad (9)$$

where, $\tilde{o}_t^{vis}$ are poses with added Gaussian noise $\mathcal{N}(0, \Sigma_{vis})$. For re-localization during multi-contact episodes, we can add such a factor at the start pose of every new contact episode.

V. Results and Evaluation

We evaluate our approach qualitatively and quantitatively on a number of episodes where an object, unknown a priori, must be tracked from a sequence of tactile measurements. We compare against a set of baselines on two fronts: a) on surface normal predictions from images and b) on the final tracking error of object poses. We use PyTorch [37] for training surface normal models, and GTSAM C++ library [38] for factor graph optimization. We specifically use the iSAM2 [31] solver for efficient, real-time optimization.

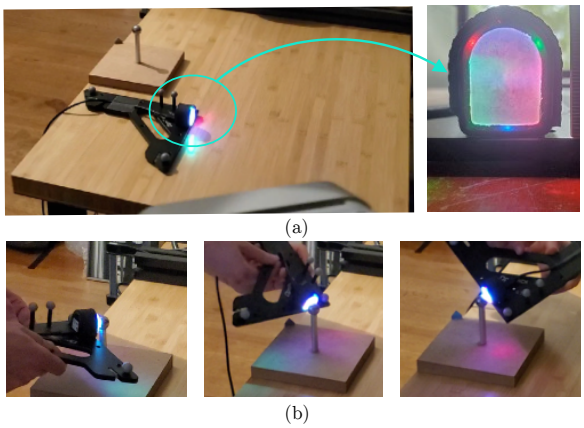

Fig. 3: Real-world experimental setup showing: **(a)** a Sphere object mounted on a workbench with a close-up of the Digit sensor, and **(b)** a contact episode with the sensor moved around the object.

A. Experimental setup

Simulator: We collected simulation data using the Tacto [18] simulator where one can load the Digit sensor, an object and render high-resolution tactile image readings in real-time. The simulator uses PyBullet [39] as the underlying physics engine and Pyrender [34] as the back-end rendering engine for generating images. We generated trials by using a position controller to move the object on the sensor surface. We collect data for a diverse set of objects: Sphere, Cube, Toy human and Toy brick.

Real-world: For real-world episode, we used a Digit [1] sensor to get tactile measurements. We mounted the object on a workbench and mounted the Digit on a movable plate, both of which were tracked by an OptiTrack motion capture

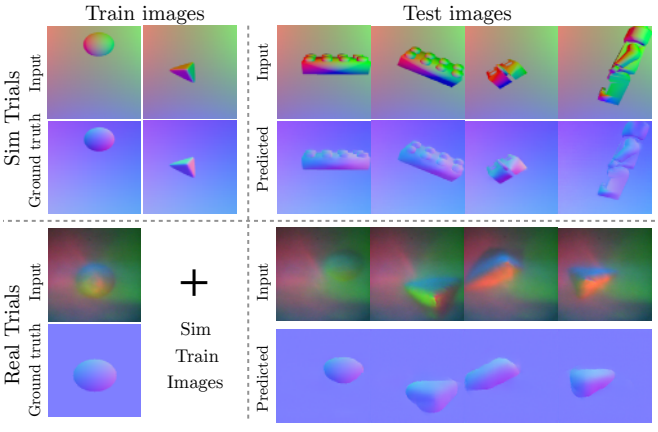

Fig. 4: Training and evaluation of the pix2pix (unet) model that maps input color images to predicted surface normal images.

TABLE I: Learned surface normal performance (validation loss)

Model type	Object Shapes				
	Sim sphere	Cube	Real sphere	Toy brick	Toy human
pix2pix (unet)	**0.4e-3**	**0.5e-3**	**1.0e-3**	**1.1e-3**	**0.9e-3**
pix2pix (resnet)	0.6e-3	1.2e-3	1.6e-3	**1.1e-3**	1.0e-3
MLP 3-layer	1.4e-3	**0.5e-3**	5.8e-3	5.1e-3	6.7e-3

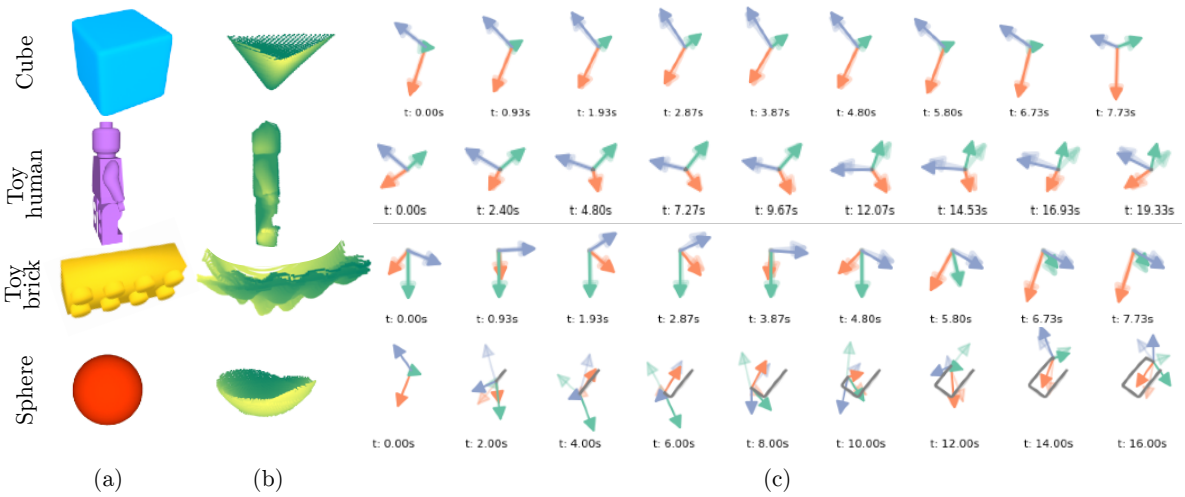

Fig. 5: Qualitative pose tracking performance on simulated trials. **(a)** Object being tracked. **(b)** Local patch map reconstructed online. **(c)** Estimated 3D pose coordinates (light) against ground truth (dark). Rotations in orange (x), purple (y), green (z). Translations in grey.

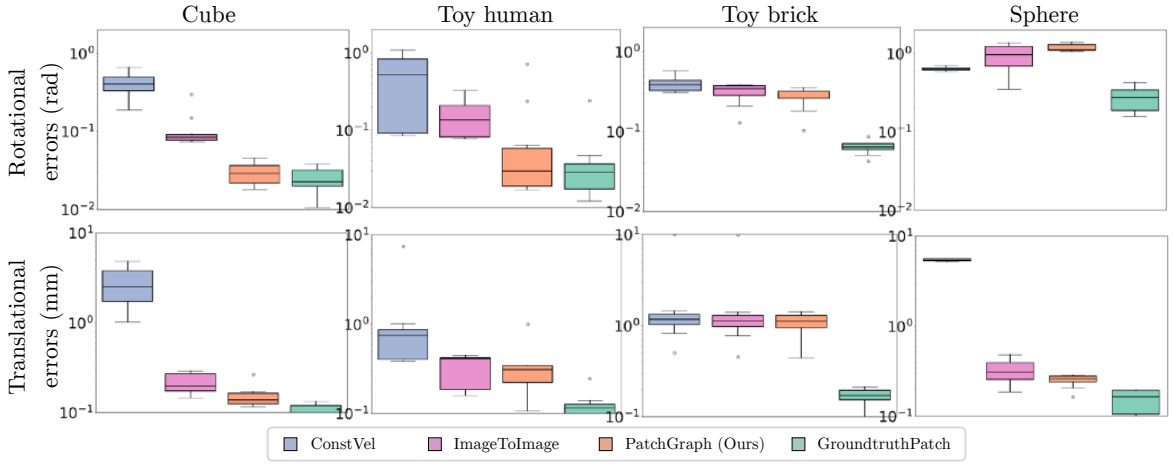

Fig. 6: Final tracking error box plots on simulated trials: **(top)** rotation and **(bottom)** translation errors in semi-log scale.

system to get ground truth poses. We collect real-world data for two objects: a Sphere (ball-bearing) of 1/2" diameter and a Pyramid 1/2" tall with 1.75" side length.

B. Surface normal reconstructions

We first analyze the accuracy of learned surface normals from color images collected by the Digit tactile sensor. For simulation episodes, we only train on two simple objects: 50 images of Sphere and 50 images of Cube. We tested the model on two different held-out shapes: Toy brick and Toy human. For real episodes, we only had ground truth normals for the real sphere. Hence, we expanded the training dataset to include both the training data from simulation and the real Sphere data. We tested the model on held-out real Pyramid object. For baselines, we picked two different pix2pix architectures, unet and resnet. We also trained a baseline 3-layer MLP 5-32-32-3 with tanh activation on an L2 loss, mapping a single color pixel (r,g,b,x,y) to a surface normal (nx,ny,nz), similar to prior work [7].

Qualitative reconstructions: Fig. 4 shows qualitative performance of the pix2pix (unet) model on both real and simulated images. We can see the model generalizes to fairly different shapes such as the Toy human and Toy brick, even though it is trained on simple objects. Moreover, fine-tuning this model with very little real data, i.e., 50 images of the real Sphere, enables it to generalize to unseen geometries in the real-world such as different local patches of a Pyramid.

Quantitative model performance: Table I compares mean-squared pixel loss on the validation dataset for different model choices. We see that pix2pix, for both unet and resnet architecture, has a fairly low MSE loss and generalizes to unseen shapes such as the Toy human and Toy brick. On the other hand, the per-pixel MLP baseline incurs a high MSE loss. In general, the per-pixel MLP is likely to be insufficient for non-ideal tactile sensors with effects such as self-shadowing of the gel. Having convolutional layers, such as with the pix2pix architectures, can address such confounding effects since these are typically localized in the image, e.g. shadows are cast by nearby object features since object depth is small relative to the image size.

C. Factor graph optimization

We now look at the final task performance of tracking 3D object poses using tactile image measurements. We compare 4 objectives, each of which uses different factors: *ConstVel* uses only a constant velocity prior, *ImageToImage* uses

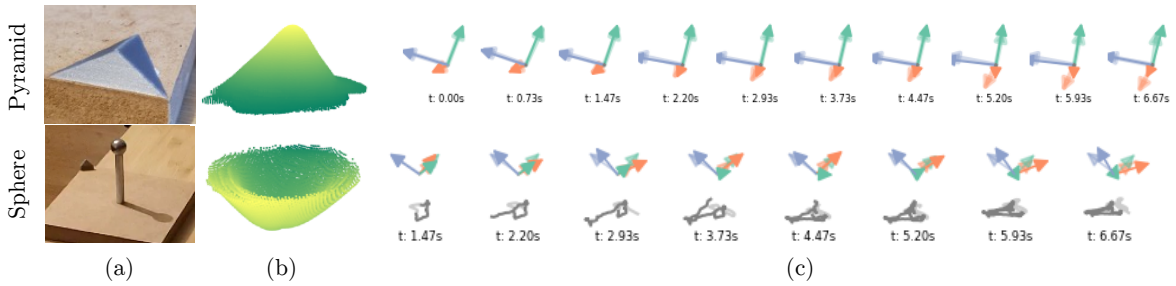

Fig. 7: Qualitative pose tracking performance on real trials. **(a)** Object being tracked. **(b)** Local patch map reconstructed online. **(c)** Estimated 3D pose coordinates (light) against ground truth (dark). Rotations in orange (x), purple (y), green (z). Translations in grey.

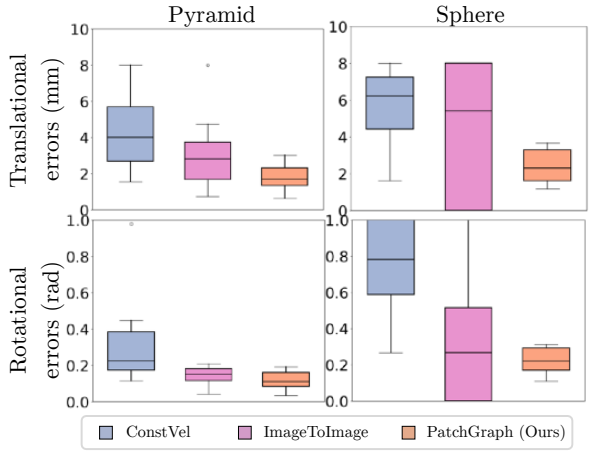

Fig. 8: Final tracking error box plots on real trials: **(top)** rotation and **(bottom)** translation errors.

image-to-image factors, *PatchGraph* (ours) additionally uses Image-to-patch factors (Fig. 2) and *GroundtruthPatch* uses a global object model. *GroundtruthPatch* assumes the object is known a-priori and hence represents the best a method can do. We keep the covariance parameters same across all runs in the graph, i.e. Σ_{im2im}=1e-3 $\times \mathbb{I}_{6\times6}$, Σ_{im2pc}=1e-3 $\times \mathbb{I}_{6\times6}$, Σ_{vel}=1e-2 $\times \mathbb{I}_{6\times6}$, Σ_{eff}=1e-5 $\times \mathbb{I}_{6\times6}$. First pose of the object is assumed to be known, and added as a unary prior to graph.

Simulation tracking performance: Fig. 5 shows the qualitative tracking performance of *PatchGraph* for various objects in simulated trials. For Cube, Toy brick and Toy human, *PatchGraph* is able to reliably track rotations of the object. The local patch, constructed from online estimates, appears to be consistent with the local object geometry, thus explaining the good tracking performance. The object that was most difficult to track was Toy brick, as evidenced by the distortions in the patch, owing to jerkiness of the in-contact motions. We also note that tracking rotations for a Sphere has high errors, which is expected since rotations of a sphere are unobservable from tactile images.

Fig. 6 shows the quantitative rotation and translation tracking performance against baselines on all objects, with 20 distinct contact sequences per object. Overall, *PatchGraph* has the lowest errors matching closely to that of *GroundtruthPatch* that has the global object model. This lends credence to our claim that a local patch suffices for reliable object tracking. *ConstVel* understandably has the highest variance among any baselines. *ImageToImage* fails to outperform *PatchGraph* on any of the datasets. Toy brick appears to be the most challenging among datasets, where *GroundtruthPatch* has a clear performance gap.

Real tracking performance: Fig. 7 shows qualitative tracking performance of *PatchGraph* for various objects in real trials. Fig. 7(b) shows the local patches which appear consistent with the local object geometry, and is a key piece to reliable tracking. Fig. 7(c) shows good rotation tracking for Pyramid and translation tracking for Sphere.

Fig. 8 shows quantitative rotation and translation tracking performance against baselines on all objects, with 10 distinct contact sequences per object. The main observation is that *ImageToImage* performs much worse than *ImageToPatch*, particularly in translation errors for Sphere. This is primarily because point clouds generated from individual images are not as geometrically discriminative as the fused local patch, causing *ImageToImage* factors alone to diverge quickly. *PatchGraph* keeps translation errors under 4mm and rotation errors under 0.2rad, which looks promising for use in dexterous object manipulations.

VI. Conclusion

We presented a factor graph-based approach for tracking 3D object poses from tactile image sequences during in-hand manipulations. We showed reliable tracking on 4 simulated objects and 2 real objects without relying on any a priori object information. We achieved this by exploiting two decompositions of the tracking problem. First, that a complex object can be treated as a composition of many local patches each of which can be mapped and tracked largely independently. Second, surface normal information is highly localized within a tactile image and independent of the global object shape.

A primary limitation that can cause tracking failures is when the local patch map is not sufficiently discriminative geometrically, e.g. flat or featureless patches, or when the patch motions are degenerate in the observed image space, e.g. rotations of a spherical object. As future work, it would be interesting to explore solutions that take into account geometric degeneracies [40, 41] as well as approaches that can detect slip and shear [42] to disambiguate motion degeneracies. Another interesting future direction would be to complement the tracker with a global first pose re-localization that is able to generalize across objects, e.g. using visual images to predict a contact location likelihood.

Acknowledgements

We thank Wenzhen Yuan for insightful feedback on the paper.

REFERENCES

[1] M. Lambeta, P.-W. Chou, S. Tian, B. Yang, B. Maloon, V. R. Most, D. Stroud, R. Santos, A. Byagowi, G. Kammerer, *et al.*, "DIGIT: A novel design for a low-cost compact high-resolution tactile sensor with application to in-hand manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 3838–3845, 2020.

[2] W. Yuan, S. Dong, and E. H. Adelson, "GelSight: High-resolution robot tactile sensors for estimating geometry and force," *Sensors*, vol. 17, no. 12, p. 2762, 2017.

[3] P. Sodhi, M. Kaess, M. Mukadam, and S. Anderson, "Learning tactile models for factor graph-based estimation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021.

[4] S. Suresh, M. Bauza, K.-T. Yu, J. G. Mangelson, A. Rodriguez, and M. Kaess, "Tactile slam: Real-time inference of shape and pose from planar pushing," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2021.

[5] A. S. Lambert, M. Mukadam, B. Sundaralingam, N. Ratliff, B. Boots, and D. Fox, "Joint inference of kinematic and force trajectories with visuo-tactile sensing," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3165–3171, 2019.

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

[7] S. Wang, Y. She, B. Romero, and E. Adelson, "Gelsight wedge: Measuring high-resolution 3d contact geometry with a compact robot finger," *arXiv preprint arXiv:2106.08851*, 2021.

[8] M. Bauza, E. Valls, B. Lim, T. Sechopoulos, and A. Rodriguez, "Tactile object pose estimation from the first touch with geometric contact rendering," *Conference on Robot Learning (CoRL)*, 2020.

[9] J. Liang, A. Handa, K. Van Wyk, V. Makoviychuk, O. Kroemer, and D. Fox, "In-hand object pose tracking via contact feedback and gpu-accelerated robotic simulation," *arXiv preprint arXiv:2002.12160*, 2020.

[10] M. Bauza, O. Canal, and A. Rodriguez, "Tactile mapping and localization from high-resolution tactile imprints," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 3811–3817, 2019.

[11] F. Dellaert, "Factor graphs: Exploiting structure in robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, 2020.

[12] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[14] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "DeepFactors: Real-time probabilistic dense monocular SLAM," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 721–728, 2020.

[15] R. Hartley, M. G. Jadidi, L. Gan, J.-K. Huang, J. W. Grizzle, and R. M. Eustice, "Hybrid contact preintegration for visual-inertial-contact state estimation using factor graphs," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3783–3790, 2018.

[16] E. Donlon, S. Dong, M. Liu, J. Li, E. Adelson, and A. Rodriguez, "Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[17] A. Yamaguchi and C. G. Atkeson, "Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables," in *IEEE Intl. Conf. on Humanoid Robots (Humanoids)*, pp. 1045–1051, 2016.

[18] S. Wang, M. Lambeta, P.-W. Chou, and R. Calandra, "Tacto: A fast, flexible and open-source simulator for high-resolution vision-based tactile sensors," *arXiv preprint arXiv:2012.08456*, 2020.

[19] Z. Si and W. Yuan, "Taxim: An example-based simulation model for gelsight tactile sensors," *arXiv preprint arXiv:2109.04027*, 2021.

[20] A. Agarwal, T. Man, and W. Yuan, "Simulation of vision-based tactile sensors using physics based rendering," *arXiv preprint arXiv:2012.13184*, 2020.

[21] S. Dong, D. K. Jha, D. Romeres, S. Kim, D. Nikovski, and A. Rodriguez, "Tactile-rl for insertion: Generalization to objects of unknown geometry," *arXiv preprint arXiv:2104.01167*, 2021.

[22] S. Dong and A. Rodriguez, "Tactile-based insertion for dense box-packing," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[23] Y. She, S. Wang, S. Dong, N. Sunil, A. Rodriguez, and E. Adelson, "Cable manipulation with a tactile-reactive gripper," in *Robotics: Science and Systems (RSS)*, 2020.

[24] G. Izatt, G. Mirano, E. Adelson, and R. Tedrake, "Tracking objects with point clouds from vision and touch," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 4000–4007, 2017.

[25] B. Saund, S. Chen, and R. Simmons, "Touch based localization of parts for high precision manufacturing," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 378 – 385, 2017.

[26] M. C. Koval, N. S. Pollard, and S. S. Srinivasa, "Pose estimation for planar contact manipulation with manifold particle filters," *Intl. J. of Robotics Research*, vol. 34, no. 7, pp. 922–945, 2015.

[27] Z. Pezzementi, C. Reyda, and G. D. Hager, "Object mapping, recognition, and localization from tactile geometry," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 5942–5948, 2011.

[28] B. Sundaralingam, A. S. Lambert, A. Handa, B. Boots, T. Hermans, S. Birchfield, N. Ratliff, and D. Fox, "Robust learning of tactile force estimation through robot interaction," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019.

[29] R. Li, R. Platt, W. Yuan, A. ten Pas, N. Roscup, M. A. Srinivasan, and E. Adelson, "Localization and manipulation of small parts using gelsight tactile sensing," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pp. 3988–3993, 2014.

[30] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[31] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *Intl. J. of Robotics Research*, vol. 31, pp. 216–235, Feb. 2012.

[32] P. Sodhi, S. Choudhury, J. G. Mangelson, and M. Kaess, "ICS: Incremental constrained smoothing for state estimation," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2020.

[33] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2017.

[34] M. Matl, "Pyrender," 2019. `https://github.com/mmatl/pyrender`.

[35] J. Doerner, "Fast poisson reconstruction in python." `https://gist.github.com/jackdoerner/b9b5e62a4c3893c76e4c`.

[36] `http://www.songho.ca/opengl/gl_projectionmatrix.html`.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 24 (NeurIPS)*, pp. 8024–8035, 2019.

[38] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," tech. rep., Georgia Institute of Technology, 2012.

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

[40] J. Zhang, M. Kaess, and S. Singh, "On degeneracy of optimization-based state estimation problems," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pp. 809–816, 2016.

[41] E. Westman and M. Kaess, "Degeneracy-aware imaging sonar simultaneous localization and mapping," *Journal of Oceanic Engineering*, 2019.

[42] W. Yuan, R. Li, M. A. Srinivasan, and E. H. Adelson, "Measurement of shear and slip with a gelsight tactile sensor," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.