Neural feels with neural fields: Visuo-tactile perception for in-hand manipulation

Sudharshan Suresh,[1,2*] Haozhi Qi,[2,3] Tingfan Wu,[2] Taosha Fan,[2]
Luis Pineda,[2] Mike Lambeta,[2] Jitendra Malik,[2,3] Mrinal Kalakrishnan,[2]
Roberto Calandra,[4,5] Michael Kaess,[1] Joseph Ortiz,[2] Mustafa Mukadam[2]

[1]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2]FAIR, Meta, 1 Hacker Way Menlo Park, CA 94025, USA
[3]Department of Electrical Engineering and Computer Sciences, UC Berkeley, CA 94720, USA
[4] Institute of Artificial Intelligence, Technische Universität Dresden, 01062, Dresden, Germany
[5]The Centre for Tactile Internet with Human-in-the-Loop (CeTI), 01062, Dresden, Germany

*To whom correspondence should be addressed; E-mail: suddhus@gmail.com.

To achieve human-level dexterity, robots must infer spatial awareness from multimodal sensing to reason over contact interactions. During in-hand manipulation of novel objects, such spatial awareness involves estimating the object's pose and shape. The status quo for in-hand perception primarily employs vision, and restricts to tracking a priori known objects. Moreover, visual occlusion of objects in-hand is imminent during manipulation, preventing current systems to push beyond tasks without occlusion. We combine vision and touch sensing on a multi-fingered hand to estimate an object's pose and shape during in-hand manipulation. Our method, NeuralFeels encodes object geometry by learning a neural field online and jointly tracks it by optimizing a pose graph problem. We study multimodal in-hand perception in simulation and the real-world, interacting with different objects via a proprioception-driven policy. Our experiments show final reconstruction F-scores of 81% and average pose

drifts of 4.7 millimeters, further reduced to 2.3 millimeters with known object models. Additionally, we observe that under heavy visual occlusion we can achieve up to 94% improvements in tracking compared to vision-only methods. Our results demonstrate that touch, at the very least, refines and, at the very best, disambiguates visual estimates during in-hand manipulation. We release our evaluation dataset of 70 experiments, FeelSight, as a step towards benchmarking in this domain. Our neural representation driven by multimodal sensing can serve as a perception backbone towards advancing robot dexterity.

Summary

Neural perception with vision and touch yields robust tracking and reconstruction of novel objects for in-hand manipulation.

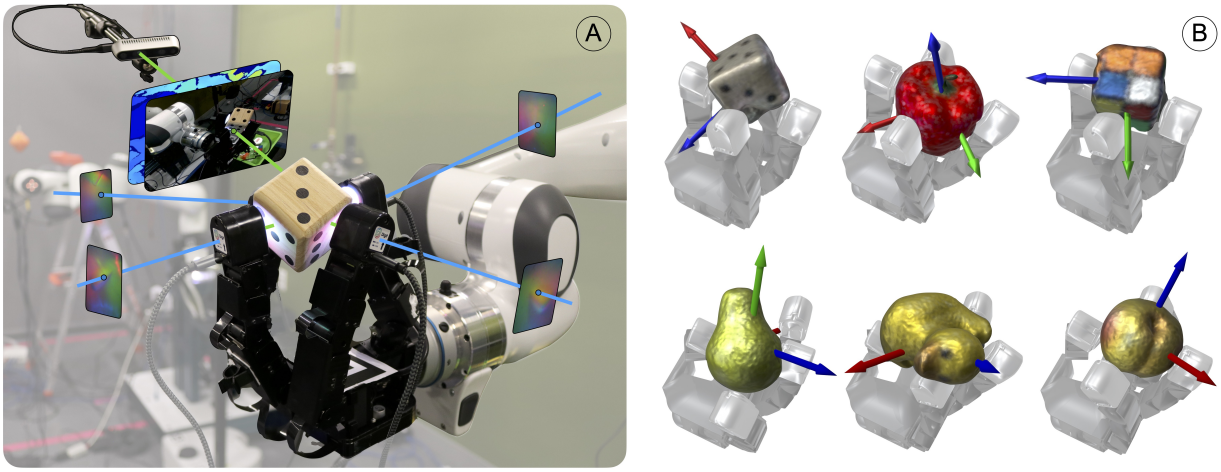


Figure 1: Visuo-tactile perception with NeuralFeels. Our method estimates pose and shape of novel objects during in-hand manipulation **(B)**, by learning neural field models online from a stream of vision, touch, and proprioception **(A)**.

INTRODUCTION

To perceive deeply is to have sensed fully. Humans effortlessly combine their senses for everyday interactions—we can rummage through our pockets in search of our keys, and deftly insert them

to unlock our front door. Currently, robots lack the cognition to replicate even a fraction of the mundane tasks we perform, a trend summarized by Moravec's Paradox (*1*). For dexterity in unstructured environments, a robot must first understand its spatial relationship with respect to the manipulated object. Indeed, as robots move out of instrumented labs and factories to cohabit our spaces, there is a need for generalizable spatial perception (*2*).

Robots need dexterity beyond pick-and-place, although grasping a hammer or screwdriver may be straightforward, tool use requires the ability to rotate and re-grasp in-hand. Specific to in-hand dexterity, knowledge of object pose and geometry is crucial to policy generalization (*3–6*). As opposed to end-to-end supervision (*7–10*), these methods require a persistent three-dimensional (3D) representation of the object. However, the status quo for in-hand perception is currently restricted to the narrow scope of tracking known objects with vision as the dominant modality (*5*). Further, it is common for practitioners to sidestep the perception problem entirely, retrofitting objects and environments with fiducials (*3, 4*). To further progress towards general dexterity, a missing piece is general, robust perception.

With visual sensing, researchers tend to tolerate interaction rather than embrace it. This is at odds with contact-rich problems where self-occlusions is imminent, like rotating (*11*), reorienting (*5, 10*), and sliding (*12, 13*). Additionally, vision often fails in the real-world due to poor illumination, limited range, transparency, and specularity. Touch provides a direct window into these dynamic interactions, and human cognitive studies have reinforced its complementarity with vision (*14*).

Researchers have made advances in tactile sensing for multifinger robots (*15*), most prominent being vision-based fingertip sensors (*16–23*) like the GelSight and DIGIT. Progress in simulation (*24*) enables practitioners to learn tactile observation models that transfer to real-world interactions (*22, 25, 26*). With a fingertip form-factor, their illuminated gel deforms on contact and the physical interaction is captured by an internal camera. When chained with robot kinematics, we obtain dense, situated contact that can be processed similar to natural camera images.

Now given multimodal sensing, how best to represent the spatial information? Coordinate-based learning, formalized as neural fields (*27*), has found great success in visual computing. With neural fields, practitioners can create high-quality 3D assets offline given noisy visual data and pose annotation (*28–30*). They are continuous representations that have several advantages over their discrete counterparts like point clouds, meshes, and voxel maps—differentiability, precise reconstructions, and memory efficiency. Although initially developed for offline training, lightweight signed distance field (SDF) models (*31–34*) have made online perception possible. The ease of imparting generative priors (*35*) and pre-training (*36*) make neural fields more adaptable than classical methods.

Researchers have used neural fields not just for continuous 3D quantities like SDFs and radiance (*28, 29, 36*), but also for pose estimation (*34, 37*), planning (*38*), and latent physics (*39*). Neural fields have shown promise in robot manipulation for learning policies (*40*), object deformation (*41*), scene dynamics (*38, 42*), data generation (*43*), and transparent object manipulation (*44, 45*). However, online perception and optimization of multimodal data remains an open question.

The domain of our work—an intersection of simultaneous localization and mapping (SLAM) and manipulation—has been studied for over two decades. A first exemplar is from Moll and Erdmann (*46*), who reconstruct the shape and motion of an object rolled between robot palms. The combination of vision and touch has been explored for reconstructing the shape of fixed objects (*26, 47–52*), tracking known objects (*53–55*), and global localization on known objects (*56, 57*). In full SLAM, tactile-only methods have been investigated for simple objects via planar pushing (*58, 59*), and specialized rolling fingertips (*60, 61*). Closest to our work is the visuo-tactile SLAM system by Zhao et al. (*62*), combining dense touch from a single finger with RGB images, but it does not address the challenging case of in-hand manipulation.

NeuralFeels is an online solution to localize and reconstruct object shape via in-hand manipulation. It builds on the prior work to demonstrate full SLAM with a multifinger robot for

apriori unknown objects, and robust tracking of known objects. We use a dexterous hand (*63*) sensorized with commercial vision-based touch sensors (*20*) and a fixed RGB-D camera (Fig. 1). With a proprioception-driven policy (*11*) we explore the object's extents through in-hand rotation—using the SLAM solution to guide the policy is not an explicit objective of our work. This falls in line with prior work in SLAM for manipulation (*52, 55, 57, 62*), that focus on perception by isolating their evaluation from the manipulation task.

In this article, we study the role that vision and touch play in interactive perception, the effects of occlusion, and visual sensing noise. We present our robot with a novel object, and it infers and tracks its geometry through vision, touch, and proprioception. To evaluate our work, we collect a benchmark dataset of 70 in-hand rotation trials in both the real-world and simulation, with ground-truth object meshes and tracking. Our results on novel objects show average reconstruction F-scores of 81% with pose drifts of just 4.7 mm, further reduced to 2.3 mm with known computer-aided design (CAD) models. Under heavy occlusion, we demonstrate up to 94% improvements in pose tracking compared to vision-only methods. Our combination of rich sensing and spatial perception requires minimal hardware compared to complex sensing cages, and is easier to interpret than end-to-end perception methods. The output of the neural SLAM pipeline—pose and geometry—can drive further research in general dexterity, broadening the capabilities of home robots.

RESULTS

Our multi-fingered robot hand was presented with a novel object, placed randomly between its fingertips. We rotated the object in-hand, through a proprioception-driven policy (*11*), which gave rise to a stream of visual and tactile signals. We combined the visual, tactile, and proprioceptive sensing into our online neural field, for a persistent, evolving 3D representation of the unknown object. The full pipeline of our NeuralFeels is illustrated in Fig. 2.

We evaluated NeuralFeels over simulated and real-world interactions, totaling up to 70 ex-

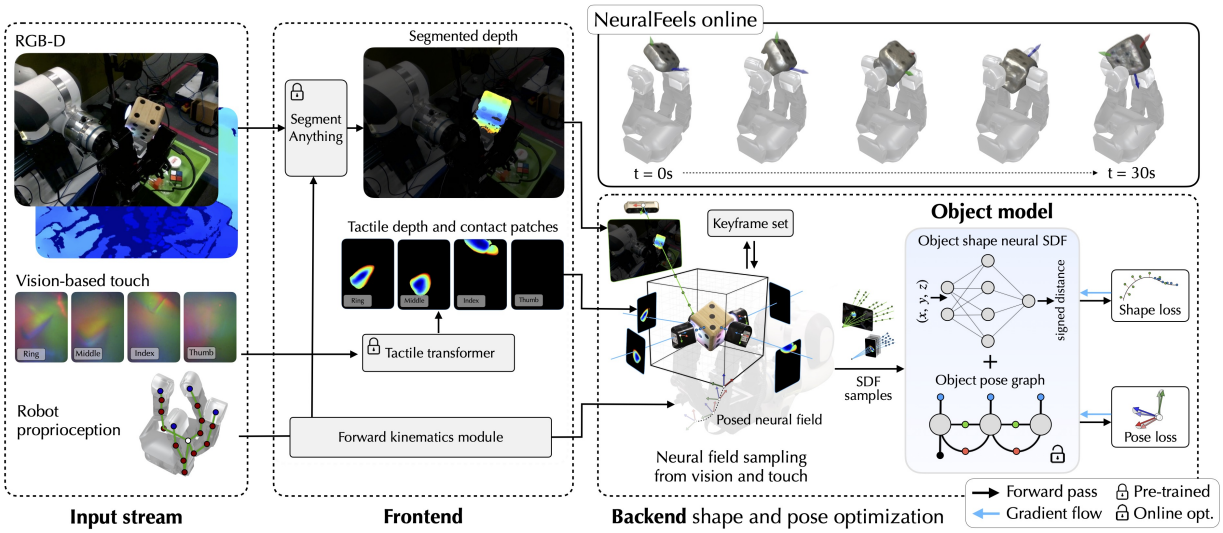


Figure 2: A visuo-tactile perception stack amidst interaction. An online representation of object shape and pose is built from vision, touch, and proprioception during in-hand manipulation. Raw sensor data is first fed into the frontend, which extracts visuo-tactile depth with our pre-trained models. Following this, the backend samples from the depth to train a neural signed distance field (SDF), and the pose graph tracks the neural field.

periments over different object classes. First, we demonstrated SLAM results for novel objects, and highlight some qualitative examples. Next, we demonstrated pose-tracking when we have a priori shape of the manipulated object. Finally, we analyzed the role touch plays in improving perception under occlusion and visual sensing noise. Movie S1 and S2 visualize representative neural SLAM results for the bell pepper and rubber duck objects respectively. Movie S3 provides a longer narrated summary of our results and methodology.

Metrics and baseline

Pose and shape metric

We used the symmetric average Euclidean distance (ADD-S) (*64*), henceforth referred to as the pose metric, to evaluate tracking error over time. The ADD metric is commonly used in manipulation (*64–67*) as a geometrically-interpretable distance metric for pose error. It is computed by sub-sampling the ground-truth object mesh and averaging the Euclidean distance between the point-set in the estimated and ground-truth object pose frames. Rather than pairwise distance, ADD-S considers the closest point distance, which disambiguates symmetric objects (*64*).

For shape, we compared how accurate (precision) and complete (recall) the neural SDF is in comparison to the ground-truth mesh. The F-score, an established metric in the multi-view reconstruction community (*68, 69*), combines these two criteria into an interpretable $[0-1]$ value. To compute this, henceforth referred to as the shape metric, we first sub-sampled the ground-truth and reconstructed meshes in their object-centric frame. Given a distance threshold, in our case $\tau = 5$ mm, precision measured the percentage of reconstructed points within τ distance from the ground-truth points. Conversely, recall measured the percentage of ground-truth points within τ distance from the reconstructed points. The harmonic mean of these two quantities gave us the F-score, which jointly captured surface reconstruction accuracy and shape completion. Broadly, a higher F-score with tighter τ bounds implied better object reconstructions.

Ground-truth shape and pose

We evaluated these metrics against the ground-truth estimates of object shape and pose. For each object, the ground-truth shape is obtained from offline scans (Fig. S1). Ground-truth object pose was straightforward in simulation experiments, directly exposed by IsaacGym (*70*). In the real-world, we estimated a pseudo ground-truth, via multi-camera pose tracking of the experiment. Instrumented solutions, such as 3D motion capture, were infeasible as it both visually and physically interfered with the experiments. We opted to install two additional cameras and ran NeuralFeels in pose tracking mode with the ground-truth object shape. This represents the best tracking estimates given known shape and occlusion-free vision. For further details, refer to the "Ground-truth shape and pose" section in the Supplementary Materials.

Object initialization

In practice, the object-centric reference frame in our SLAM experiments would be picked arbitrarily (such as the centroid of the initial point cloud or the robot fingers). However, the ground-truth reference frame was defined as the centroid of the complete CAD model, oriented along its major axis. This mismatch in the reference frames was expected in a causal system, but will lead to an incorrect calculation of the object-centric shape metric. Additionally, object tracking with a

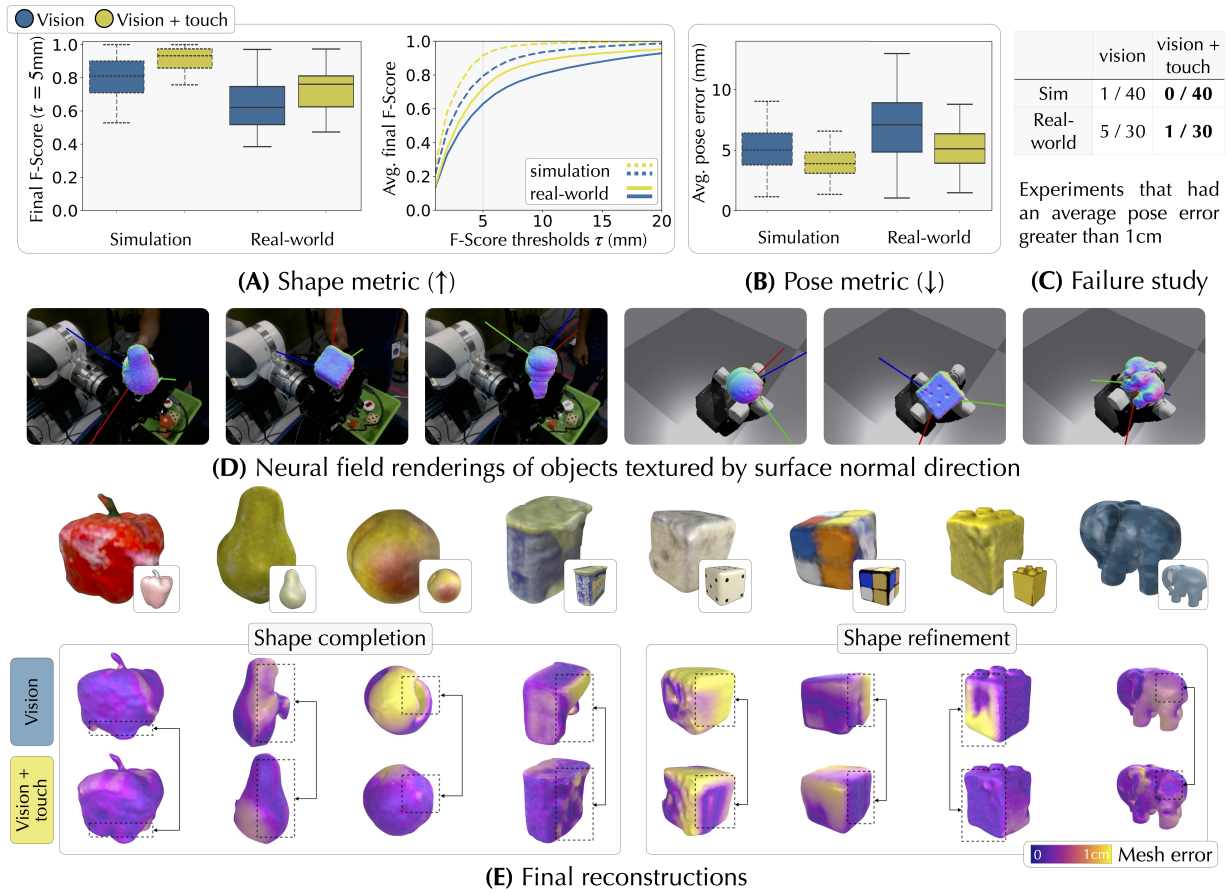
known shape is quite sensitive to initial orientation of the reference frame (*71*). To address these issues, we assumed the initial object pose was known and aligned to the initial ground-truth pose. We instead focused on the subsequent tracking and shape reconstruction, which was challenging even with good initialization. In the future, a coarse initialization can be obtained from a feature-based frontend (*72*). To ensure that our evaluation did not benefit from this object initialization, we only started computing our pose metric five seconds into each trial.

Neural SLAM: object tracking and shape estimation

Motivation and importance

As a first experiment, we evaluated NeuralFeels' ability to track and reconstruct unknown objects from multimodal sensing. This is important for robots deployed in unstructured environment with apriori unknown objects, such as households. We presented the robot with a novel object, and the robot was tasked with building an object model on-the-fly. Our SLAM method made no assumptions about the object geometry, which was built from scratch, or manipulation actions, which were decided at deployment. We processed visuo-tactile data sequentially without access to future information or category-level priors. This formulation aligns with prior dexterous manipulation work (*5, 6, 10, 11*), and is less restrictive than that of Zhao et al. (*62*), where the object was always in contact with a single tactile sensor and the camera was unobstructed.

We evaluated over a combined 70 experiments in simulation and real-world across of 14 different objects. The objects were placed in-hand, after which the policy collected 30 seconds of vision, touch, and proprioception data. As each run was non-deterministic, we averaged our results across 5 different seeds, resulting in a total of 350 trials. The first frame of each sequence only presented limited visual knowledge: a single side of Rubik's cube or large dice; the underside of the rubber duck. Through the course of any 30 second sequence, in-hand rotation exposed previously unseen geometries to vision and touch filled in the rest of the occluded surfaces. In Fig. 3, we show the main set of results, where we compared the multimodal fusion schemes against ground-truth.



	vision	vision + touch
Sim	1 / 40	**0 / 40**
Real-world	5 / 30	**1 / 30**

Experiments that had an average pose error greater than 1cm

Figure 3: Summary of SLAM experiments. **(A, B)** Aggregated statistics for SLAM over a combined 70 experiments (40 in simulation and 30 in the real-world), with each trial run over 5 different seeds. We compare across simulation and real-world to show low pose drift and high reconstruction accuracy. Each boxplot represents the aggregate error over all experiments, where the central line is the median, extents of the box are the upper and lower quartiles, and the whiskers represent $1.0\times$ the interquartile range (IQR). **(C)** The number of trials that our method failed to track (and reconstruct) the object. **(D)** Representative examples of the final object pose and neural field renderings from the experiments. Each object is textured by mapping the surface normal directions to a red-green-blue (RGB) colormap. **(E)** The final 3D objects generated by marching cubes on our neural field. Here, we highlight the role tactile played in both shape completion and shape refinement.

Object reconstructions

Fig. 3A shows the final shape metric at the end of each sequence for a fixed threshold τ. Here we picked $\tau = 5\,\text{mm}$ for this evaluation, around 3% of the maximum diagonal length of the objects. The greater the value of the shape metric, the closer the surface reconstructions were to ground-truth. We observed large gains when incorporating touch, with surface reconstructions on average 15.3% better in simulation ($p < 0.001$) and 14.6% better in the real-world ($p < 0.001$). Our final reconstructions, as seen in Fig. 3E, had a median error of 2.1mm in simulation and

3.9mm in the real-world. Additionally, the second plot compares the final shape metrics against a range of τ thresholds. Here we observed that multimodal fusion led to consistently better shape metrics across all τ values in simulation and the real-world.

Object pose drift

In SLAM, there is a strong correlation between a low shape metric and high pose metric, as often leads to the other. Fig. 3B plots the drift of the object's estimated pose with respect to the ground-truth, lower being more accurate. We observed better tracking with respect to the vision-only baseline, with improvements of 21.3% in simulation ($p < 0.001$) and 26.6% in the real-world ($p < 0.001$). Fig. 3C reports the number of failures in vision-only tracking compared to NeuralFeels. Here, a failed experiment was defined as when the average pose drift exceeded a threshold of 10 mm. This was loosely based on Bauza et al. (*73*) where they considered 10 mm as a coarse initialization for tactile localization. To highlight the importance of our neural field, Fig. S19 shows our method outperformed a baseline that relied only on iterative closest point (ICP) frame-to-frame constraints.

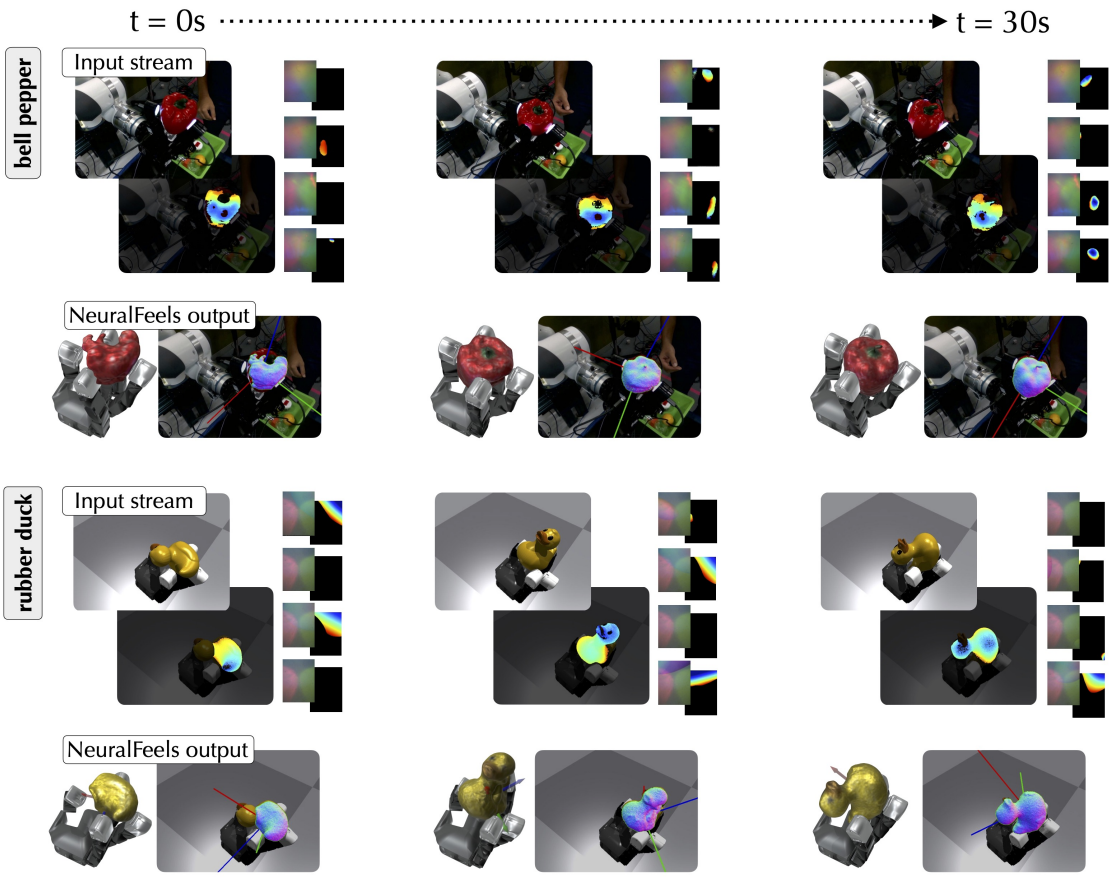
Empirically, we observed a large pose drift in the first few seconds from initialization at ground-truth, due to unknown shape. Over time, we built a better shape model, resulting in more accurate pose tracking (Fig. S17 and S18). However, with pose regularization and the lack of long-term loop closures (*72, 74*), small errors in pose would accumulate over time. This cascading effect is common in SLAM (*75*), where pose errors cause a disagreement between the reconstructed map and the physical world.

Due to this, we identified if any trials had an average shape metric that deteriorates over time. This was done by computing the difference between the average shape metric across the first 50% of the sequence and the last 50% of the sequences. We concluded that $\frac{25}{150}$ (16%) of the real-world trials had a shape estimates that deteriorated over time, as the other $\frac{125}{150}$ (83%) improved. In simulation, our method performed better: $\frac{9}{200}$ (4.5%) of trials had shape estimates that deteriorated over time as $\frac{191}{200}$ (95%) improved.

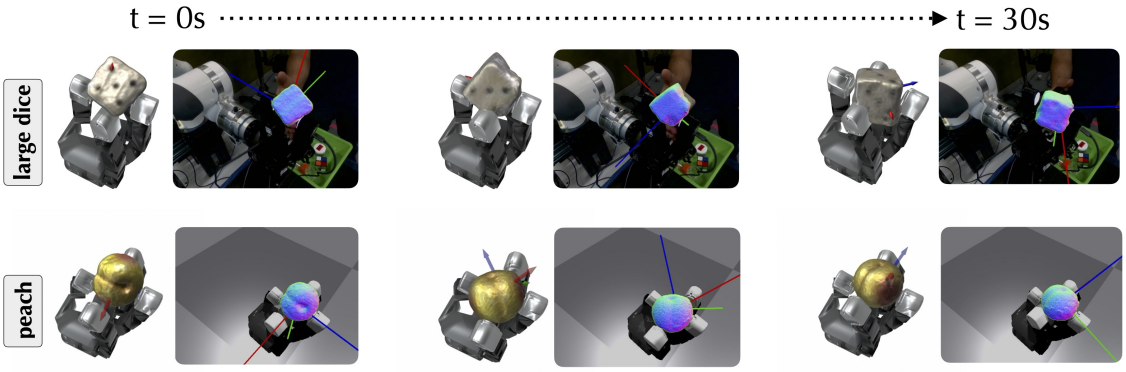
Qualitative results

Fig. 3D visualizes the rendered normals of the posed neural field at the end of each experiment, with the 3D coordinate axes superimposed. The final 3D reconstructions, generated via marching cubes (*76*), are shown in Fig. 3E alongside the ground-truth meshes. Below that, we highlight the gains with visuo-tactile integration for each of the reconstructed objects. We categorize these into shape completion—coverage of object surfaces that were occluded from vision—and shape refinement—touch measurements could compliment vision to better reconstruct visible surfaces.

Fig. 4 shows incremental pose tracking and reconstruction of objects across different time slices of a few representative experiments. At each timestep, we highlight the input stream, frontend depth and output object model. Movie S1 and S2 provide an animated version of the experiments in Fig. 4A. In the current formulation of our problem touch-only SLAM was not permissible. This is because the tracking (and thereby reconstruction) failed early in the sequence due to lack of prior shape information and the field-of-view of the sensor could not rapidly give us global geometry.



(A) Representative SLAM results from the bell pepper (real-world) and rubber duck (simulation) objects, alongside input RGB-D and tactile images at each timestep



(B) Representative SLAM results at each timestep from the large dice (real-world) and peach (simulation)

Figure 4: Representative SLAM results. (A) We show the input stream of RGB-D and tactile images, paired with the posed reconstruction at timestep t for the bell pepper and rubber duck objects. In each case, we partially reconstructed the object at the initial frame, and built the surfaces out progressively over each 30 second experiment. The 3D visualizations are generated by marching-cubes, in addition to the rendered normals of the neural field projected onto the visual image. The rendering was textured by mapping the surface normal directions to a red-green-blue (RGB) colormap. **(B)** Further representative results with the large dice (real-world) and peach (simulation) objects.

Object tracking given apriori known shape

Motivation and importance

These experiments studied the accuracy of pose tracking with NeuralFeels when provided with apriori known object CAD models. Tracking known geometries is an active area of research in manipulation (*5, 71*), with some work that incorporates touch as well (*13, 53–55, 77*). This is applicable in environments like warehouses and manufacturing lines, where robots have intimate knowledge of the manipulated objects (*77*). It is further useful in household scenarios, where the robot has already generated an object model through interaction.

In implementation, the object's SDF was pre-computed from a given CAD model. During runtime, we froze the weights of the neural field, and only performed visuo-tactile tracking with the frontend estimates. Similar to the SLAM experiments, we ran each of the 70 experiments over 5 seeds, and report the pose metrics with respect to ground-truth.

Results from pose tracking

Fig. 5A shows some qualitative examples of tracking the pose of the Rubik's cube and potted meat can with vision and touch. For the given examples, the pose metric over the sequences are plotted in Fig. 5B. We observed low, bounded pose error even with imprecise visual segmentation (Fig. S24) and sparse touch signals. In Fig. 5C we observed the role touch plays in reducing the average pose error over all experiments to the range of $2.3\,$mm. Given the CAD model, we observed that incorporating touch could refine our pose estimates, with a decrease in average pose error by 22.29% in simulation ($p < 0.001$) and 3.9% in the real-world ($p = 0.21$). We posit the relatively high real-world p-value is because the real DIGIT elastomer was less sensitive, leading to sparser contacts. Sparse contacts played a large role in full SLAM, by coarsely reconstructing unseen surfaces, but they only played a refinement role when full shape was known. In addition, the viewpoint did not have many occlusions—in the following section, we highlight greater improvements when visual sensing was sub-optimal.

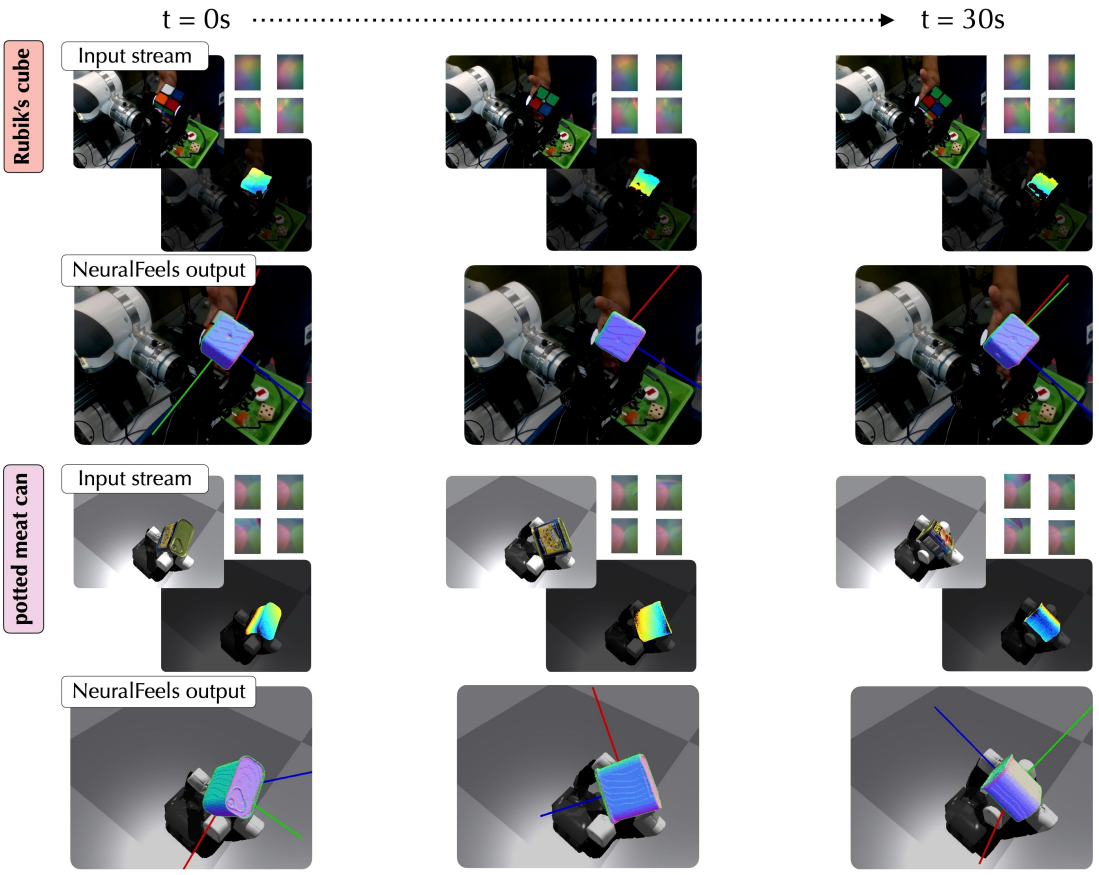
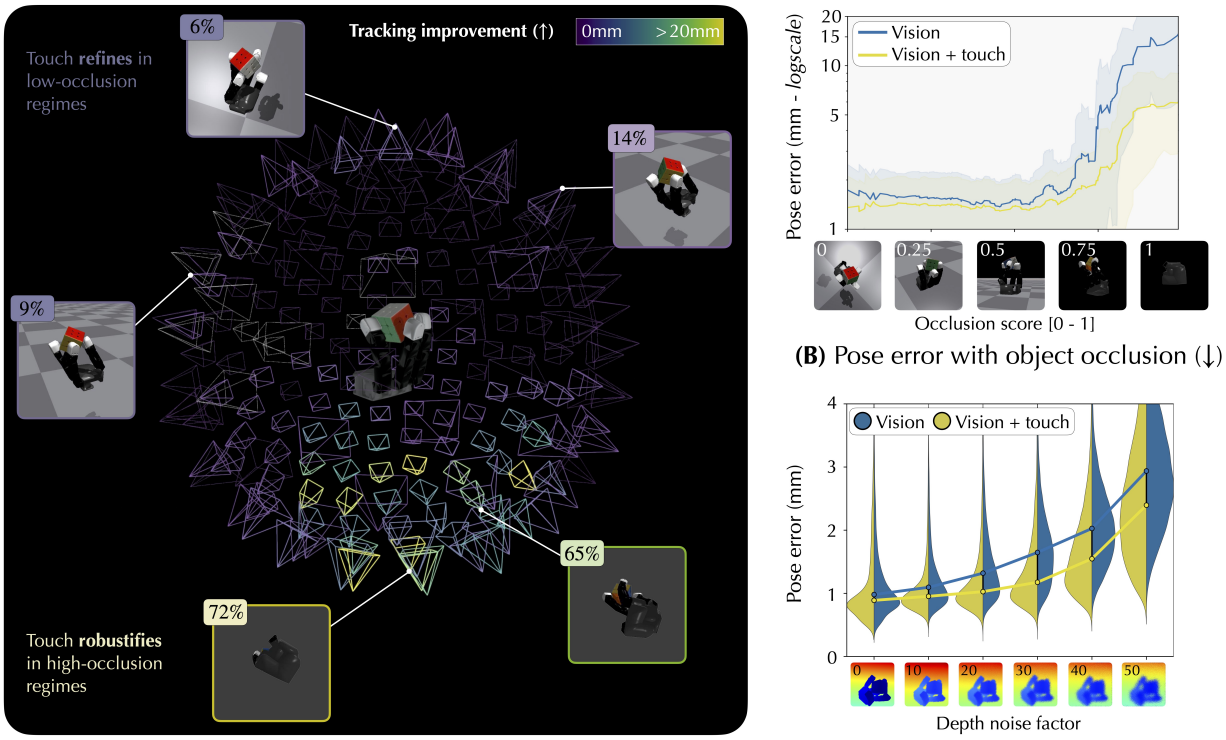


Figure 5: Neural pose tracking of known objects. **(A)** We show the input stream of RGB-D and tactile images, paired with the pose tracking at timestep t for the Rubik's cube and potted meat can objects. With known ground-truth shape, we could robustly track objects with vision and touch. Each experiment was 30 seconds long, and the object renderings were textured by mapping the surface normal directions to a red-green-blue (RGB) colormap. **(B)** We observed reliable tracking performance, with average pose errors of 2 mm through the sequence. **(C)** Aggregated statistics for pose tracking over a combined 70 experiments (40 in simulation and 30 in the real-world), with each trial run over 5 different seeds. Each boxplot represents the aggregate pose error in logscale, where the central line is the median, extents of the box are the upper and lower quartiles, and the whiskers represent $0.25\times$ the interquartile range (IQR). With a known object model and good visibility, touch played the role of pose refinement. Additionally, we note that touch-only tracking is error-prone and infeasible.



(A) Object occlusion: Tracking improvement by fusing touch (↑)

(B) Pose error with object occlusion (↓)

(C) Pose error with depth noise (↓)

Figure 6: Ablations on occlusions and sensing noise. (A) Pose tracking results from 200 simulated cameras, in a sphere of radius 0.5 m, each facing towards the robot. Each camera view is colormapped based on the pose tracking improvements from incorporating touch, when compared against vision-only. At occlusion-heavy points-of-view, visuo-tactile fusion provided an unobstructed local perspective leading to improved tracking performance. **(B)** We computed a $[0-1]$ occlusion score for each of the 200 experiments, and plotted the pose errors against it. We observed that touch played a larger role when vision was heavily occluded, and a refinement role when we there was negligible occlusion. The shaded regions represent one standard deviation from the mean. **(C)** We simulated noise in visual depth measurements, and plot the error distribution against the depth noise factor D as a violin plot. The inset image shows the qualitative depth noise for each D, and the inner markers represent the median pose error. We observed that with increase in noise, adding touch led to lower error distribution.

Perceiving under duress: occlusion and visual depth noise

Motivation and importance

In this section, we explored the broader benefits of fusing touch and vision in challenging scenarios — occlusion and visual noise. The previous results were achieved through largely favorable camera positioning and precise stereo depth tuning. Indeed, this attention to detail was necessary for prior practitioners as well (*5, 10*), but could we also use touch to improve over sub-optimal visual data? We considered two such scenarios in simulation, where we could freely control these parameters, and evaluated on the pose tracking problem from the previous section.

The effects of camera-robot occlusion

In an embodied problem, third-person and egocentric cameras are both susceptible to occlusion from robot motion and environment changes. For example, if we were to retrieve a cup off the top shelf in the kitchen, we would rely primarily on tactile signals to complete the task. For the perception system, this translates to the object of interest disappearing from the field of view, while local touch sensing is still unaffected. To emulate this we considered tracking the pose of a known Rubik's cube. We simulated 200 different cameras in a sphere of radius $0.5\,$m, each facing towards the robot. As shown in Fig. 6A, each camera captured a unique vantage point of the same in-hand sequence, with varying levels of robot-object occlusion. This served as proxy for occlusion faced by an egocentric or fixed camera when either the hand or environment occluded the object.

To simplify the experiment, we assumed the upper-bound performance of the vison-only frontend by providing ground-truth object segmentation masks. We characterized the visibility in terms of an occlusion score by calculating the average segmentation mask area for each viewpoint, and normalizing them to $[0-1]$. For example, scores closer to 0 corresponded to viewpoints beneath the hand (most occluded), and those closer to 1 corresponded to cameras placed atop (least occluded). We ran pose tracking experiments for each of the 200 cameras in two modes: vision-only and visuo-tactile and compared between them.

In Fig. 6A we colormapped each camera view based on the pose tracking improvements from incorporating touch. On average the improvement across all cameras was 21.2%, and it peaked at 94.1% at heavily occluded views. Across the $[0-1]$ range of occlusion scores, we had $p < 0.001$. We inset frames from a few representative viewpoints and their corresponding relative improvement with visuo-tactile fusion. In Fig. 6B the pose error for each modality is further plotted versus the $[0-1]$ occlusion score. This corroborated the idea that touch refined perception in low-occlusion regimes and robustified it in high-occlusion regimes.

The effects of noisy visual depth

Depth from commodity RGB-D sensors are degraded as a function of camera-robot distance, environment lighting, and object specularity. Even in ideal scenarios, the RealSense depth algorithm has 35 hyperparameters (*78*) that considerably affect the frontend input to NeuralFeels. To simulate this, we corrupted the depth maps progressively with a realistic RGB-D noise, and observed the tracking performance for a known geometry.

As implemented by Handa et al. (*79*), we simulated common sources of depth-map errors as a sequence of pixel shuffling, quantization, and high frequency noise. The depth noise factor D determined the magnitude of these operations, with the depth-maps visualized in Fig. 6C. All prior simulation experiments had been collected with $D = 5$, but here we varied the magnitude from $0{-}50$ in intervals of 10. At each noise level, we ran pose tracking across the 5 Rubik's cube experiments with 5 unique seeds, resulting in a total of 150 experiments. In Fig. 6C we plotted error against the noise factor D, showing an expected upward trend in error with noise. However, we saw better tracking when fusing touch, especially in high-noise regimes.

DISCUSSION

The experiments show NeuralFeels achieves robust object-centric SLAM for multimodal, multifinger manipulation. As shown in the Fig. 3A, we achieve average reconstruction F-scores of 81% across simulation and real-world experiments on novel objects. Simultaneously, we stably track these objects amidst interaction with minimal drift, an average of $4.7\,$mm. Although the vision-only baseline may suffice for some scenarios, the results validate the utility of rich, multimodal sensing for interactive tasks. This corroborates years of research in interactive perception from touch and vision (*26, 77, 80*), now applied on a dexterous manipulation platform.

Interactive perception is far from ideal, an embodiment can more often than not get in the way of sensing. As seen in Fig. 4, in-hand manipulation suffers from challenges such as frequent occlusions, limited field-of-view, noisy segmentation, and rapid object motion. Proprio-

ception helps focus the perception problem: we can accurately singulate the object of interest through embodied prompting (refer to the "Frontend" section of Materials and methods). When combined with touch, we robustify our visual estimates by giving us a window into local interactions. These are evident in simulated / real SLAM and pose tracking experiments, where multimodal fusion leads to improvements of 15.3% / 14.6% in reconstruction and 21.3% / 26.6% in pose tracking.

Qualitatively, we see touch performs two key functions: disambiguating noisy frontend estimates, and providing context in the presence of occlusion. The former alleviates the effect of noisy visual segmentation and depth with co-located local information for mapping and localization. The latter provides important context hidden from visual sensing, like the occluded face of the large dice or back of the rubber duck. The final reconstructions in Fig. 3E support these findings, with improved shape completion and refinement.

With known shape ("Object tracking given apriori known shape" section of Results), touch plays a refinement role (Fig. 5) when there aren't many visual occlusions. The largest gains from incorporating touch, unsurprisingly, are in heavy-occlusion regimes (Fig. 6A and B), where we observe up to 94.1% improvements at certain camera viewpoints. To our knowledge, a detailed study on how object visibility affects perception has not been explored in prior manipulation work. This doesn't just demonstrate the complementary nature of the modalities, but further, the ideal configurations for occlusion-free manipulation. Finally, our results in tactile-only tracking (Fig. 5C) support the findings of Smith et al. (*49*) that learning exclusively from touch leads to poor performance as it lacks any global context.

As opposed to an end-to-end perception, our modular stack marries pre-training with online learning. This allows us to combine foundation models trained on large-scale image and tactile data (frontend), with SLAM as online learning (backend). Furthermore, our backend is a combination of state-of-the-art neural models (*29*) with classical least-square optimization (*81*) that have found success in SLAM (*82*).

This modular design has benefits for future generalization of our system: other models of tactile sensors (*16, 19, 22*) can be easily integrated as long as they can be accurately simulated; alternate scene representations (*83, 84*) can supplant our neural field model; additional state knowledge can be integrated as factor graph costs like tactile odometry (*62*) and force-constraints (*59*); any combination of tactile and visual sensors can be fused given appropriate calibration and kinematics.

NeuralFeels is relevant to researchers who require spatial perception with a single camera and affordable tactile sensing. It can be extended to not just in-hand rotation, but also object-centric tasks like reorientation (*10*), pick-and-place (*77*), insertion (*61*), nonprehensile sliding (*85*), and planar pushing (*59*). Although not explored in this work, the benefit of an online SDF is the ability to seamlessly plan for dexterous interactions. Recent works demonstrate the benefit of apriori-known object point-clouds (*6*) and SDFs (*86*) for goal-conditioned planning, and running our perception stack in-the-loop is the next natural step.

System limitations

Our findings indicate that the benefits of multimodal fusion are less pronounced in real-world deployment when compared to simulation. This is a common problem in sim-to-realmanipulation—prior work has encountered similar disparities in object pose estimation (*3, 5*). Additionally, we identify that: the DIGIT elastomer is less sensitive in real-world deployment, leading to sparser contacts (Fig. 4); our reinforcement learning (RL) policy is less reliable in the real-world, often requiring human intervention and causing large jumps in object motion (Fig. S21). To tackle these shortcomings we can focus on real-world fine tuning of our simulator (*87*) and explicitly modeling sensor deformation and stress (*88*). Through multimodal RL (*6, 10*), we can deliver more robust policies than those driven "blindly" by proprioception.

We are currently restricted to a fixed-camera setup, with an online hand-eye calibration or egocentric vision; this can be relaxed. Depth uncertainty (*89*) is valuable information for our neural model to handle visually-adversarial objects like glass and metal. We use vision-based touch (*20*)

over tactile-arrays (*90*) or binary sensing (*7*), but future work can consider the merits of each. In the section titled "The role of touch" found in the Supplementary materials, we present ablations on the benefits of higher resolution and a comparison against binary sensing. In our SLAM experiments, each pose graph iteration takes 0.79 ± 0.36 secs (20 iterations of Levenberg–Marquardt (*75*)), and the shape optimization takes 0.06 ± 0.09 secs (one iteration of gradient descent). For execution in a real-time loop, we can speed-up Segment Anything (SAM) inference time (*91*), reduce SDF samples and downsample feature grid resolution, substitute the pose graph with an incremental optimizer (*92*). Finally, we can increase tracking robustness through feature-based methods (*93*) and loop-closure detection (*72*).

Future directions

Our method learns a 3D geometry of a novel object from scratch, and thus the pose tracker has a higher chance of failure in the initial seconds, when the SDF is unknown. Additionally, our rotation policy might not completely explore the object in the real-world, resulting in a lower average final F-Score of 81%. Given an initial occluded view, integration of large reconstruction models (*36, 94, 95*) can yield a good initial-guess SDF. In manipulation, Wang et al. (*48*) have seen promising results in using shape priors for visuo-tactile reconstruction of fixed objects.
Geometry is just a starting point for neural models: interaction reveals latent properties like texture (*85*), friction (*39*), and object dynamics (*96*). With neural fields, we can embed these latents as auxiliary optimization variables to benefit tasks that go beyond just spatial quantities. Applications can range from learning to manipulate inertially-challenging objects (like a hammer), to identifying a grasp point from local texture (like a saucepan handle).

In summary, NeuralFeels leverages vision, touch, and robot proprioception to reconstruct and track novel objects with high precision. The system is simpler than complex fiducial tracking, uses affordable touch sensing, and provides more interpretable output than end-to-end perception. Our approach combines ideas from SLAM, neural rendering and tactile simulation, and serves as an important step towards advancing robot dexterity.

MATERIALS AND METHODS

Similar to classical SLAM frameworks, NeuralFeels first has a frontend, which converted the the vision (RGB-D) and touch (RGB) input stream into a format suitable for estimation (segmented depth). Thereafter, the backend fused this data into an optimization structure that inferred the object model: an evolving posed object SDF. An illustration of the entire pipeline is found in Fig. 2, which we refer the reader back to throughout this section. Additionally, a narrated summary of our method can be found in Movie S3.

Task definition

NeuralFeels incrementally built an object model, simultaneously optimizing for the object SDF network's weights θ and its corresponding pose $\mathbf{x}_t$ at timestep t. For object exploration, we used a proprioception-driven policy π_t that executed the optimal action to achieve stable rotation. The input stream (Fig. 2) of all sensors $\mathcal{S}$ consisted of the following: RGB-D vision—image I_t^c and depth D_t^c from calibrated camera $c \in \mathcal{S}$; RGB touch—images I_t^s from four DIGITs (*20*); $s \in \{d_{\text{index}}, d_{\text{middle}}, d_{\text{ring}}, d_{\text{thumb}}\} \in \mathcal{S}$; and proprioception—joint-angles $\mathbf{q}_t$ from robot encoders.

Robot hardware and simulation

The Allegro hand (*63*) was retrofit with four DIGIT vision-based tactile sensors (*20*), at each of the distal ends. The DIGIT produced a 240×320 RGB image of the physical interaction at 30 Hz. The Allegro published 16D joint-angles so as to situate the tactile sensors with respect to the base frame. The hand was rigidly mounted on a Franka Panda arm, with an Intel Realsense D435 RGB-D camera placed at approximately 27 cm from its palm. The camera extrinsics were computed with respect to the base frame of the Allegro through ArUco (*97*) hand-eye calibration. For our vision pseudo-ground-truth we used three such cameras in the workspace (Fig. 7), jointly calibrated via Kalibr (*98*), to achieve approximately 1 px reprojection error. Our simulator replicated the real-world setup: a combination of the IsaacGym physics simulator (*70*) with the

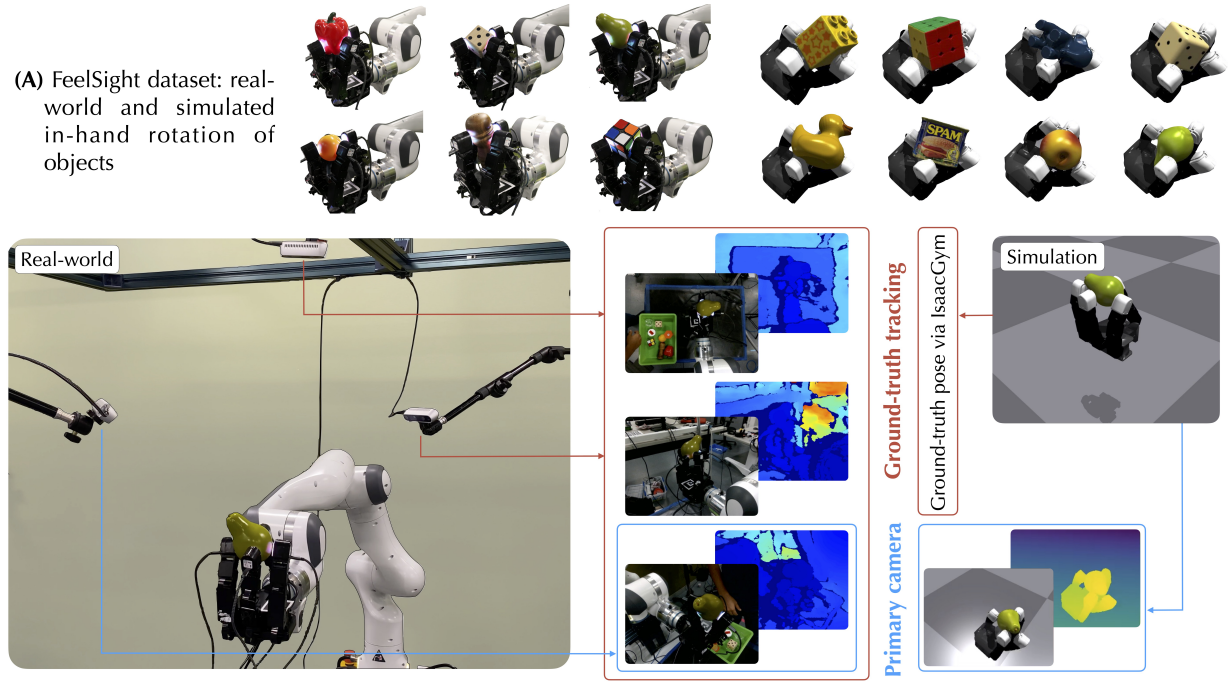


Figure 7: Experiment setup in the real-world and simulation. (A) Still frames of the Allegro robot manipulating objects from the FeelSight dataset with our in-hand rotation policy. These visuo-tactile interactions are captured across the real-world and physics simulation. **(B)** The robot cell was made up of three realsense RGB-D cameras, an Allegro robot hand mounted on a Franka Panda, and four DIGIT tactile sensors. All real-world results used the primary camera and DIGIT sensing, and the additional cameras were fused for our ground-truth pose tracking. In simulation, we simulated an identical primary camera in IsaacGym with simulated touch from the TACTO simulator.

TACTO touch renderer (*24*). In this case, we recorded and store the true ground-truth object pose directly from IsaacGym.

FeelSight: a visuo-tactile perception dataset

Visuo-tactile perception lacks a dataset that has driven progress in adjacent fields like visual tracking (*99*), SLAM (*100*), and reinforcement learning (*101*). Towards this, we collected our FeelSight dataset for visuo-tactile manipulation. We used an in-hand rotation policy to collect vision, touch, and proprioception for 30 seconds per trial.

Reinforcement learning for object rotation

When we encounter a novel object, we tend to twirl it in our hand to get a better look from different views, and regrasp it from different angles. The equivalent for a multi-fingered hand, in-hand

rotation, is an ideal choice for the interactive perception problem. We adopted the method of Qi et al. (*11*) where they trained a proprioception-based policy in simulation, and directly transferred it to the real-world. The policy training and deployment, reward function, and performance are discussed in the "In-hand rotation policy" section of the Supplementary Materials. In all our experiments, a single policy π_t updated at 20 Hz (300 Hz low-level PD control) via the Robot Operating System (ROS) Allegro package.

This achieved multifingered rotation of novel objects and interesting visuo-tactile stimuli. The dataset has five rotation trials each of six objects in the real-world and eight objects in simulation; a total 35 minutes of interaction. As explained in Fig. 7, we recorded a pseudo-ground-truth in the real-world, and exact ground-truth poses in simulation. The policy resulted in a translation / rotation of 25mm/sec / $32.6°$/sec in simulation and 20mm/sec / $9.9°$/sec in the real-world.

The selected objects varied in geometry and size from 6-18 cm in diagonal length. Empirically, objects with irregular aspect ratios were harder to manipulate with the hand morphology; our choice of objects was based on the ability of our RL policy rather than the SLAM solution. Deformable object manipulation was deemed out-of-scope as we relied on IsaacGym (*70*) and TACTO (*24*), which assumed rigid body simulation. Ground-truth real-world meshes were created with the Revopoint 3D scanner (*102*), and the simulated objects used ground-truth meshes from the Yale-CMU-Berkeley (YCB) (*103*) and ContactDB (*104*) datasets.

For objects like the Rubik's cube, we assisted the policy through human intervention in case of slip events (Fig. S21). In the real-world, we found that with this robot hand morphology it was difficult to achieve gaits for stable cube rotation. This was because, unlike prior work that pivots the object atop the fingers using gravity (*11*), we relied on frictional contact to get tactile signals on our lateral-facing DIGITs. Thus we opted for this strategy of de-risking our experiments with a human-in-the-loop. These interventions enabled us to collect a large set of experiments, but they were adversarial to perception as they led to additional occlusions and sudden jumps in

object pose.

Method overview

We represented the object SDF as a neural network with weights θ, whose output was transformed by the current object pose $\mathbf{x_t}$. This continuous function $F^{\theta}_{\mathbf{x}_t}(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{R}$ mapped a 3D coordinate $\mathbf{p}$ to a scalar signed-distance from the object's closest surface. Online updates were decomposed into alternating steps between refining the weights of the neural SDF θ, and optimizing the object pose $\mathbf{x_t}$. Our bespoke object model represents both the pose and object geometry over time. This is further described in the "Object model" section of Materials and Methods.

Given RGB-D vision, RGB touch, and proprioception inputs, our frontend returned segmented depth measurements compatible with our backend optimizer. These modules were pretrained with a large corpus of data. The shape optimizer used the frontend output and optimized for θ at fixed object pose $\bar{\mathbf{x}}_\mathbf{t}$ via gradient descent (*29*). Each shape iteration resulted in improved object SDF $F^{\theta}_{\bar{\mathbf{x}}_t}$. Finally, the pose optimizer built and solved an object pose-graph (*81*) for $\mathbf{x}_t$ given fixed network weights $\bar{\theta}$. Every pose iteration spatially aligned the evolving object SDF with the current set of frontend output. This is further described in the "Frontend" and "Backend: shape and pose optimizer" sections of Materials and Methods.

Key insights

NeuralFeels is a posed neural field

The object model $F^{\theta}_{\mathbf{x_t}}$ is estimated by an alternating optimization of both the neural field weights θ, and the object pose $\mathbf{x}_t$. Prior work estimated the pose of a sensor in a trained neural field by "inverting" this optimization—iNeRF (*37*) is a key example of this idea. Other works looked at jointly-optimizing the weights of the neural field and pose (*32, 33, 105*). In our case, robot kinematics gave us the pose of the touch sensors, and extrinsics gave us the pose of the camera. So, we instead flipped this paradigm to estimate the pose of the neural field with respect to

known-pose sensors.

Touch is vision, albeit local

Another insight is that vision-based touch could be approximated as a perspective camera model in tactile simulators like TACTO (*24*). There were, however, differences that must be accounted for in image formation. First, vision-based tactile sensor imposed their own color and illumination to the scene, which made it hard to get reliable visual cues. Second, a tactile image stream had considerably smaller metric field-of-view and depth-range was usually in centimeters rather than meters. Third, tactile images had depth discontinuities along all non-contact regions, as opposed to natural images which only had them along occlusion boundaries. Our method addressed these challenges as it consistently used depth rather than color for optimization, sampled at different scales (centimeter v.s. meter) based on sensing source, and sampled only surface points for touch, but both free-space and surface points for vision. More details are in the "Backend: shape and pose optimizer" section of Materials and Methods.

Object model

In general, a neural SDF (*29, 31, 106*) represents 3D surfaces as the zero level-set of a learnable function $F(\mathbf{p}) : \mathbb{R}^3 \rightarrow \mathbb{R}$. The scalar field's sign indicates if any query point $\mathbf{p}$ in the volume is inside (negative), outside (positive) or on (≈ 0) the reconstructed surface. $\mathbf{p}$ is first positionally-encoded (*107*) into a higher-dimensional space, which helped the network better approximate high-frequency surfaces. This is followed by a multi-layer perceptron (MLP) that fit the encoding to a scalar field. Typically, this network is optimized with depth samples from a camera of known intrinsics, and annotated poses from structure-from-motion (*108*).

A neural SDF is more compact than the more popular neural radiance fields (*28*), as they do not model color and appearance properties. This was sufficient for manipulation, as we cared more about estimating geometry than generating novel-views. Recently, Instant-NGP (*29*) demonstrated a learnable multiresolution hash table as a positional encoding that greatly accelerates SDF optimization with small MLP backbones. This had been successfully leveraged for

real-time SLAM in indoor scene (*105*). In our work, $F^{\theta}_{\mathbf{x}_t}$ represented the neural SDF of the object at a given pose $\mathbf{x}_t$. $\mathbf{x}_0$ was initialized to be at the object ground-truth, and θ was randomly initialized. Both shape and pose were estimated via alternating optimization, emulating the paradigm of tracking and mapping that had achieved success in robot vision (*82*).

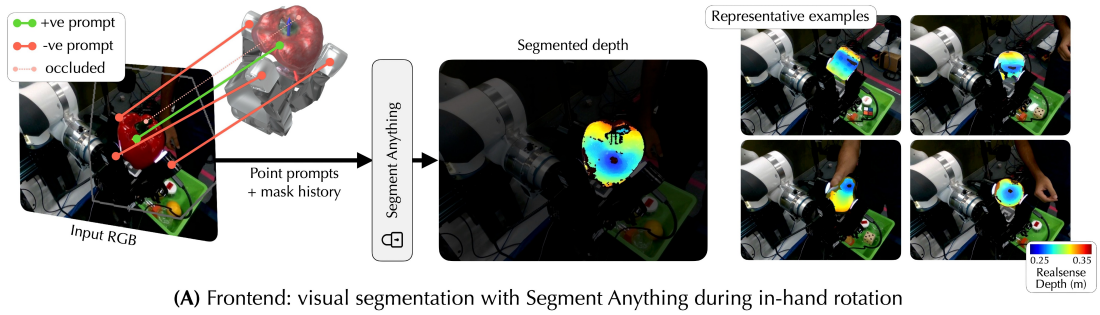
Frontend

The frontend, shown in the center column of Fig. 2, extracted depth measurements from raw vision and touch sensing. Depth was available as-is in an RGB-D camera, but the challenge was to robustly segment out object depth pixels in occluded interactions. Towards this, we introduced a kinematics-aware segmentation strategy using powerful vision foundation models (*109*). For vision-based touch, estimating depth from images was an open research problem (*22, 25, 26, 66, 110*). Towards this, we presented a transformer architecture that accurately predicted DIGIT contact patches from inputs images. Both of the frontend networks were pre-trained from a large corpus of data. The output of our frontend was a segmented depth image $\hat{D}^s_t$ for each sensor $s \in \mathcal{S}$.

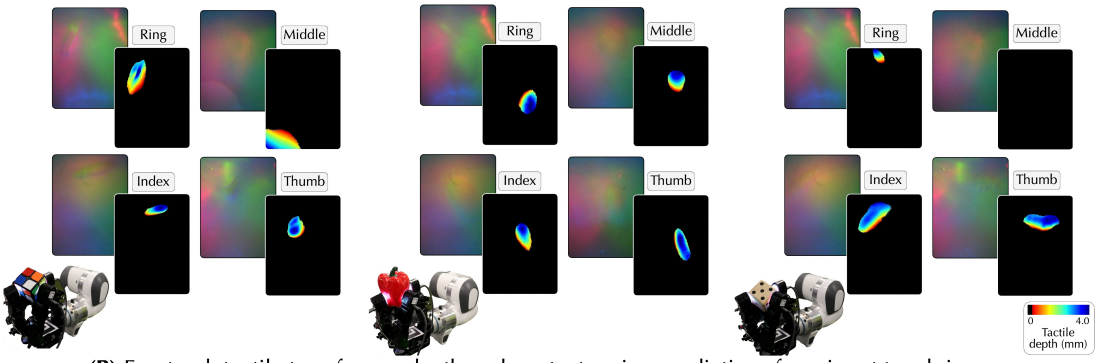
Segmented visual depth

Robust segmentation of the image stream I^c_t had successfully been demonstrated by image foundation models, like the Segment Anything Model (SAM) (*109*). Trained with a vision transformer (ViT) in the data-rich natural image domain, SAM generalized to novel scenes for state-of-the-art, zero-shot instance segmentation. For any input RGB image, SAM outputs an embedding that must be queried by user prompts (such as point, binary mask, bounding-box, or natural language prompts). At timestep t, we fed the model both positive and negative point prompts alongside the mask prediction from timestep $t-1$.

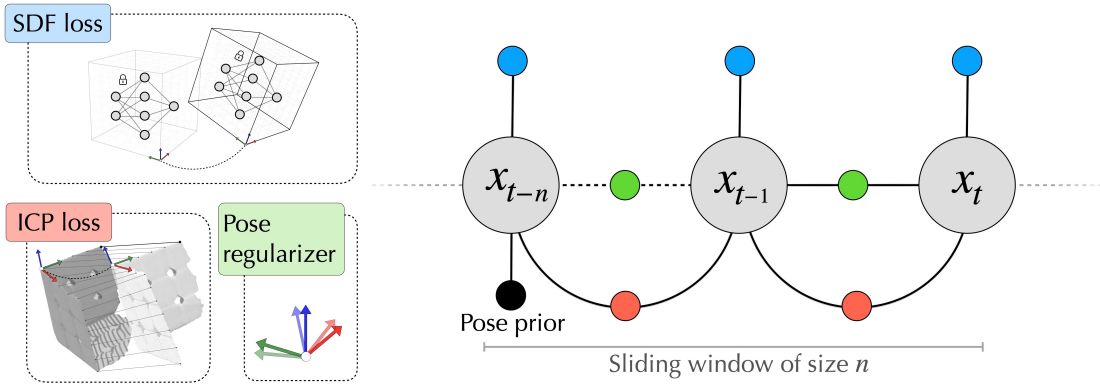
Through robot proprioception (refer Fig. 2), we obtained the four fingertip positions $\mathbf{p}_{\text{f}}$ and the computed the centroid $\mathbf{p}_{\text{c}} = \bar{\mathbf{p}}_{\text{f}}$. Given our camera c with known projection operation Π^c, we could obtain any such 3D point $\mathbf{p}$ as a pixel $(u, v) = \Pi^c(\mathbf{p})$ on the image I^c_t. Assuming the object exists in-hand, the centroid pixel $\Pi^c(\mathbf{p}_{\text{c}})$ served as a useful positive prompt.



(A) Frontend: visual segmentation with Segment Anything during in-hand rotation



(B) Frontend: tactile transformer depth and contact region predictions from input touch images



(c) Backend: pose graph optimization

Figure 8: Frontend and backend. **(A)** Through reasoning about finger occlusion and object pose with respect to the fingers, we can accurately prompt Segment-Anything (*109*) for robust output masks. **(B)** Representative examples of the sim-to-real performance of the tactile transformer. Each RGB image is fed through the network to output a predicted depth, along with a contact mask. **(C)** Our sliding window nonlinear least squares optimizer estimates the object pose x_t from the outputs of the frontend. Each object pose x_t is constrained by the signed distance field (SDF) loss, frame-to-frame iterative closest point (ICP), and pose regularization to ensure tracking remains stable.

In practice, this prompt alone did not suffice—the robot hand frequently appeared in segmentation causing large errors in backend optimization. To address this, we first rendered the current object model $F^{\theta}_{\mathbf{x_t}}$ onto camera c to check if it occluded any fingertip pixels $\Pi^c(\mathbf{p}_\mathrm{f})$. Each unoccluded fingertip pixel $\Pi^c(\mathbf{p}_\mathrm{f})$ was used as a negative prompt.

In Fig. 8A we visualized the segmentation on real-world images, alongside the SAM prompts. In our experiments we used the Vision Transformer Large (ViT-L) model with 308M parameters. This achieved a speed of around 4Hz, but in practice we could use efficient segmentation models (*91*) for speeds up to 40Hz. The "Additional implementation details" section of the Supplementary Materials highlights steps we took for robust visual segmentation.

Tactile transformer

In contrast, vision-based touch images were out-of-distribution from images SAM was trained on, and did not directly provide depth either. The embedded camera perceives an illuminated gelpad, and contact depth is either obtained via photometric stereo (*16*), or supervised learning (*22, 25, 26, 66, 110*). Existing touch-to-depth relies on convolution, however recent work had shown the benefit of a ViT for dense depth prediction (*111*) in natural images. We trained a tactile transformer for predicting contact depth from vision-based touch to generalize across multiple real-world DIGIT sensors.

The architecture was trained entirely in tactile simulation, using weights initialized from a pre-trained image-to-depth model (*111*). The tactile transformer represented the inverse sensor model $\mathbf{\Omega} : I_t^s \mapsto \hat{D}_t^s$ where $s \in \{d_{\text{index}}, d_{\text{middle}}, d_{\text{ring}}, d_{\text{thumb}}\} \in \mathcal{S}$. This architecture was based on the dense vision transformer (*111*) and was lightweight (21.7M parameters) compared to its fully-convolutional counterparts (*13*).

Similar to prior work (*13, 26*), we generated a large corpus of tactile images and paired ground-truth depthmaps in the optical touch simulator TACTO (*24*). We collected 10K random tactile interactions each on the surface of 40 unique YCB objects (*103*). For sim-to-real transfer we augmented the data with randomization in sensor light-emitting diodes (LEDs), indentation depth, and pixel noise. In TACTO, image realism was achieved by compositing with template non-contact images from real-world DIGITs. For more details on the training and data, refer to the "Tactile transformer: data and training" section of the Supplementary Materials.

These augmentations enabled generalized performance across our multi-finger platform, where

each sensor had differing image characteristics. Our tactile transformer was supervised on mean-square depth reconstruction loss against the ground-truth depthmaps from simulation. Based on the predicted depthmaps, the output was thresholded to mask out non-contact regions. We demonstrated an average prediction error of 0.042 mm on simulated test set, and Fig. 8B shows sim-to-real performance on real-world images.

Backend: shape and pose optimizer

The backend took depth and sensor poses from the frontend to build our object model online. This alternated between shape and pose optimization steps using samples from the visuo-tactile depth stream. Similar to other neural SLAM methods (*31*), the modules maintained a bank of keyframes over time, similar to the strategies of Ortiz et al. (*31*) and Sucar et al. (*32*), to generate these samples. More details of the backend and keyframing are found in the "Additional implementation details" section of the Supplementary Materials.

Shape optimizer

For online estimation it was intractable to optimize $F^{\theta}_{\tilde{\mathbf{x}}_t}$ using all input frames as in neural radiance fields (*28*). We opted for an online learning approach (*31, 32*), which built a subset of keyframes $\mathcal{K}$ on-the-fly to optimize over. The backend would both accept new keyframes based on a criteria, and replay old keyframes in the optimization to prevent catastrophic forgetting (*32*). Each iteration of the shape optimizer replayed a batch $k_t \in \mathcal{K}$ of size 10 per sensor to optimize our network. This included the latest two frames, and a weighted random sampling of past keyframes based on average rendering loss. The initial visuo-tactile frame was automatically added as a keyframe:

$$\mathcal{K}_0 = \{\hat{D}^s_0 \mid s \in \mathcal{S}\} \tag{1}$$

and every subsequent keyframe $\mathcal{K}_t$ was accepted using an information gain metric (*32*). For this, the average rendering loss was computed from the frozen network $F^{\theta}_{\tilde{\mathbf{x}}_t}$ using the given keyframe pose and compared against a threshold $d_{\text{thresh}} = 0.01$ m. Finally, if we had not added a keyframe

for an interval $t_{\text{max}} = 0.2$ secs, we forced one to be added.

Sampling and SDF loss

At each iteration, we sampled coordinates in the neural volume from k_t to optimize the neural weights θ. The first step was to sample a batch of pixels $\mathbf{u}_{k_t}$ from k_t—a mix of surface and free-space pixels. The surface pixels directly supervised the SDF zero level-set, and free-space pixels carved out the neural volume. In our implementation, we sampled 50% of camera pixels in free-space, although we only sampled surface pixels for touch. Through each pixel $u \in \mathbf{u}_{k_t}$ given their corresponding sensor pose, we projected a ray into the neural volume. Similar to Ortiz et al. (*31*), we sampled P_u points per ray, a mix of stratified and surface points.

With these samples, we computed an SDF prediction $\hat{\mathbf{d}}_u$ for each $\hat{D}_t \in k_t$, as the batch distance bound (*31*). For each ray, we split the samples into P_u^{f} and P_u^{tr} based on if $\hat{\mathbf{d}}_u$ was within the truncation distance $d_{\text{tr}} = 5$ mm from the surface. Our shape loss resembled the truncated SDF loss of Azinović et al. (*106*):

$$\mathcal{L}_{\text{shape}} = \mathcal{L}_{\text{f}} + w_{\text{tr}}\mathcal{L}_{\text{tr}}, \;\; \text{with} \;\; w_{\text{tr}} = 10 \tag{2}$$

where the free-space and truncated losses were as follows:

$$\mathcal{L}_{\text{f}} = \frac{1}{|\mathbf{u}_{k_t}|} \sum_{u \in \mathbf{u}_{k_t}} \frac{1}{|P_u^{\text{f}}|} |F^{\theta}_{\bar{\mathbf{x}}_t}(P_u^{\text{f}}) - d_{\text{tr}}| \qquad \text{and} \qquad \mathcal{L}_{\text{tr}} = \frac{1}{|\mathbf{u}_{k_t}|} \sum_{u \in \mathbf{u}_{k_t}} \frac{1}{|P_u^{\text{tr}}|} |F^{\theta}_{\bar{\mathbf{x}}_t}(P_u^{\text{tr}}) - \hat{\mathbf{d}}_u| \tag{3}$$

Pose optimizer

Before each shape iteration, we used a pose graph (*75*) to refine the object pose $\mathbf{x}_t$ with respect to the frozen neural field $F^{\bar{\theta}}_{\mathbf{x}_t}$. We achieved this by inverting the problem to instead optimize for the 6 degrees of freedom (DoF) poses in a sliding window of size n. At timestep t, if we had accumulated N keyframes, this represented poses $\mathcal{X}_t = (\mathbf{x}_i)_{N-n \leq i \leq N}$ and measurements $\mathcal{M}_t = \left(\hat{D}_i^s \mid s \in \mathcal{S} \right)_{N-n \leq i \leq N}$. Similar to pose updates in visual SLAM (*32, 33, 37*), the network weights $\bar{\theta}$ were frozen and we estimated the $SE(3)$ poses $\mathcal{X}_t$ instead.

We formulated the problem as a nonlinear least squares optimization with custom measurement factors in Theseus (*81*). Although prior work used gradient descent (*37*), we instead used a

second-order Levenberg–Marquardt (LM) solver, which provided faster convergence (*75*). The pose graph, illustrated in Fig. 8C, solved for the following factors:

$$\hat{\mathcal{X}}_t = \underset{\mathcal{X}_t}{\operatorname{argmin}}\, \mathcal{L}_{\text{pose}}(\mathcal{X}_t \mid \mathcal{M}_t, \bar{\theta}) \qquad \text{where} \qquad \mathcal{L}_{\text{pose}} = w_{\text{sdf}}\mathcal{L}_{\text{sdf}} + w_{\text{reg}}\mathcal{L}_{\text{reg}} + w_{\text{icp}}\mathcal{L}_{\text{icp}} \tag{4}$$

Our **SDF loss** $\mathcal{L}_{\textbf{sdf}}$ factor used the previously-defined shape loss $\mathcal{L}_{\text{shape}}$, modified such that we sample only about surface points of each ray. This worked well for both visual and tactile sensing as we have higher confidence in SDFs about the surface of the object than in free-space. For each depth measurement in $\mathcal{M}_t$, we sampled surface points over M rays, and averaged the SDF loss along each ray. This resulted in an $M \times n$ SDF loss, which we used to update the se(3) lie algebra of $\mathcal{X}_t$. We implemented a custom Jacobian for this cost function, which was up to $4\times$ more efficient than PyTorch automatic differentiation.

The **pose regularizer** $\mathcal{L}_{\textbf{reg}}$ factor applied a weak regularizer between consecutive keyframe poses in $\mathcal{X}_t$ to ensure the relative pose updates stayed well-behaved. This was important for robustness to noisy frontend depth and incorrect segmentations. We further added an **ICP loss** $\mathcal{L}_{\textbf{icp}}$ factor that applied iterative closest point (ICP) between the current visuo-tactile pointcloud $\Pi^{-1}(\mathcal{M}_t)$ and previous pointcloud $\Pi^{-1}(\mathcal{M}_{t-1})$. This gave us frame-to-frame constraints in addition to the frame-to-model $\mathcal{L}_{\text{sdf}}$.

Statistical analysis

All p-values presented in the paper were computed via the paired samples T-Test (*112*) and reported as $(p \leq \cdot)$. Aggregated statistics in Fig. 3A, B and Fig. 5C were computed over a combined 70 trials (40 in simulation and 30 in the real-world), with each trial run over five different seeds. The boxplots Fig. 3A and B have whiskers that span $1.0\times$ the interquartile range (IQR), whereas the log-scale plot in Fig. 5C spans $0.25\times$ IQR. Finally, the line plot in Fig. 6B is shaded to represent one standard deviation from mean.

Supplementary Materials and Methods

Seven supplementary sections

Fig. S1 to S24

Table S1 to S4

Movie S1 to S3

References

1. H. Moravec, Mind children: The future of robot and human intelligence (Harvard University Press, 1988).

2. A. J. Davison, FutureMapping: The computational structure of spatial AI systems, *arXiv preprint arXiv:1803.11288* (2018).

3. OpenAI, *et al.*, Learning dexterous in-hand manipulation, *Intl. J. of Robotics Research (IJRR)* **39**, 3 (2020).

4. OpenAI, *et al.*, Solving Rubik's Cube with a robot hand, *arXiv preprint arXiv:1910.07113* (2019).

5. A. Handa, *et al.*, DeXtreme: Transfer of agile in-hand manipulation from simulation to reality, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (2023), pp. 5977–5984.

6. H. Qi, *et al.*, General In-Hand Object Rotation with Vision and Touch, *Proc. Conf. on Robot Learning (CoRL)* (PMLR, 2023), pp. 1722–1732.

7. Z.-H. Yin, B. Huang, Y. Qin, Q. Chen, X. Wang, Rotating without Seeing: Towards In-hand Dexterity through Touch, *Proc. Robotics: Science and Systems (RSS)* (2023).

8. I. Guzey, B. Evans, S. Chintala, L. Pinto, Dexterity from Touch: Self-Supervised Pre-Training of Tactile Representations with Robotic Play, *Proc. Conf. on Robot Learning (CoRL)* (2023).

9. I. Guzey, Y. Dai, B. Evans, S. Chintala, L. Pinto, See to touch: Learning tactile dexterity through visual incentives, *arXiv preprint arXiv:2309.12300* (2023).

10. T. Chen, *et al.*, Visual dexterity: In-hand reorientation of novel and complex object shapes, *Science Robotics* **8** (2023).

11. H. Qi, A. Kumar, R. Calandra, Y. Ma, J. Malik, In-hand object rotation via rapid motor adaptation, *Proc. Conf. on Robot Learning (CoRL)* (PMLR, 2022), pp. 1722–1732.

12. Y. She, *et al.*, Cable manipulation with a tactile-reactive gripper, *Intl. J. of Robotics Research (IJRR)* **40**, 1385 (2021).

13. S. Suresh, Z. Si, S. Anderson, M. Kaess, M. Mukadam, MidasTouch: Monte-Carlo inference over distributions across sliding touch, *Proc. Conf. on Robot Learning (CoRL)* (2022).

14. H. B. Helbig, M. O. Ernst, Optimal integration of shape information from vision and touch, *Experimental brain research* **179**, 595 (2007).

15. Z. Kappassov, J.-A. Corrales, V. Perdereau, Tactile sensing in dexterous robot hands, *Robotics and Autonomous Systems* **74**, 195 (2015).

16. W. Yuan, S. Dong, E. H. Adelson, GelSight: High-resolution robot tactile sensors for estimating geometry and force, *Sensors* **17**, 2762 (2017).

17. E. Donlon, *et al.*, GelSlim: A high-resolution, compact, robust, and calibrated tactile-sensing finger, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (IEEE, 2018), pp. 1927–1934.

18. B. Ward-Cherrier, *et al.*, The TacTip family: Soft optical tactile sensors with 3D-printed biomimetic morphologies, *Soft robotics* **5**, 216 (2018).

19. A. Alspach, K. Hashimoto, N. Kuppuswamy, R. Tedrake, Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation, *Proc. IEEE Intl. Conf. on Soft Robotics (RoboSoft)* (IEEE, 2019), pp. 597–604.

20. M. Lambeta, *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 (RA-L)* **5**, 3838 (2020).

21. A. Padmanabha, *et al.*, OmniTact: A multi-directional high-resolution touch sensor, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2020), pp. 618–624.

22. S. Wang, Y. She, B. Romero, E. Adelson, GelSight Wedge: Measuring High-Resolution 3D Contact Geometry with a Compact Robot Finger, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2021).

23. W. K. Do, M. Kennedy, Densetact: Optical tactile sensor for dense shape reconstruction, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2022), pp. 6188–6194.

24. S. Wang, M. M. Lambeta, P.-W. Chou, R. Calandra, TACTO: A Fast, Flexible, and Open-source Simulator for High-resolution Vision-based Tactile Sensors, *IEEE Robotics and Automation Letters (RA-L)* (2022).

25. P. Sodhi, M. Kaess, M. Mukadam, S. Anderson, PatchGraph: In-hand tactile tracking with learned surface normals, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (2022).

26. S. Suresh, Z. Si, J. G. Mangelson, W. Yuan, M. Kaess, ShapeMap 3-D: Efficient shape mapping through dense touch and vision, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (Philadelphia, PA, USA, 2022).

27. Y. Xie, *et al.*, Neural fields in visual computing and beyond, *Computer Graphics Forum* (Wiley Online Library, 2022), vol. 41, pp. 641–676.

28. B. Mildenhall, *et al.*, NeRF: Representing scenes as neural radiance fields for view synthesis, *Communications of the ACM* **65**, 99 (2021).

29. T. Müller, A. Evans, C. Schied, A. Keller, Instant neural graphics primitives with a multiresolution hash encoding, *ACM Transactions on Graphics (ToG)* **41**, 1 (2022).

30. Z. Li, *et al.*, Neuralangelo: High-Fidelity Neural Surface Reconstruction, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 8456–8465.

31. J. Ortiz, *et al.*, iSDF: Real-Time Neural Signed Distance Fields for Robot Perception, *Proc. Robotics: Science and Systems (RSS)* (2022).

32. E. Sucar, S. Liu, J. Ortiz, A. J. Davison, iMAP: Implicit mapping and positioning in real-time, *Proc. Intl. Conf. on Computer Vision (ICCV)* (2021), pp. 6229–6238.

33. Z. Zhu, *et al.*, NICE-SLAM: Neural implicit scalable encoding for SLAM, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 12786–12796.

34. B. Wen, *et al.*, BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 606–617.

35. A. Yu, V. Ye, M. Tancik, A. Kanazawa, PixelNeRF: Neural radiance fields from one or few images, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 4578–4587.

36. J. J. Park, P. Florence, J. Straub, R. Newcombe, S. Lovegrove, DeepSDF: Learning continuous signed distance functions for shape representation, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 165–174.

37. L. Yen-Chen, *et al.*, iNeRF: Inverting neural radiance fields for pose estimation, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (IEEE, 2021), pp. 1323–1330.

38. P. Grote, J. Ortiz-Haro, M. Toussaint, O. S. Oguz, Neural Field Representations of Articulated Objects for Robotic Manipulation Planning, *arXiv preprint arXiv:2309.07620* (2023).

39. S. Le Cleac'h, *et al.*, Differentiable physics simulation of dynamics-augmented neural objects, *IEEE Robotics and Automation Letters* **8**, 2780 (2023).

40. D. Driess, I. Schubert, P. Florence, Y. Li, M. Toussaint, Reinforcement learning with neural radiance fields, *Advances in Neural Information Processing Systems* **35**, 16931 (2022).

41. Y. Wi, A. Zeng, P. Florence, N. Fazeli, VIRDO++: Real-World, Visuo-tactile Dynamics and Perception of Deformable Objects, *Proc. Conf. on Robot Learning (CoRL)* (PMLR, 2023), pp. 1806–1816.

42. Y. Li, S. Li, V. Sitzmann, P. Agrawal, A. Torralba, 3D neural scene representations for visuomotor control, *Proc. Conf. on Robot Learning (CoRL)* (PMLR, 2022), pp. 112–123.

43. S. Zhong, A. Albini, O. P. Jones, P. Maiolino, I. Posner, Touching a NeRF: Leveraging Neural Radiance Fields for Tactile Sensory Data Generation, *Proc. Conf. on Robot Learning (CoRL)* (2022).

44. J. Ichnowski, Y. Avigal, J. Kerr, K. Goldberg, Dex-NeRF: Using a neural radiance field to grasp transparent objects, *Proc. Conf. on Robot Learning (CoRL)* (2022).

45. J. Kerr, *et al.*, Evo-NeRF: Evolving NeRF for sequential robot grasping of transparent objects, *Proc. Conf. on Robot Learning (CoRL)* (2022).

46. M. Moll, M. A. Erdmann, Reconstructing the shape and motion of unknown objects with active tactile sensors, *Algorithmic Foundations of Robotics V* (Springer, 2004), pp. 293–309.

47. J. Ilonen, J. Bohg, V. Kyrki, Fusing visual and tactile sensing for 3-D object reconstruction while grasping, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2013), pp. 3547–3554.

48. S. Wang, *et al.*, 3D shape perception from monocular vision, touch, and shape priors, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (IEEE, 2018), pp. 1606–1613.

49. E. J. Smith, *et al.*, 3D shape reconstruction from vision and touch, *Proc. Conf. on Neural Information Processing Systems (NeurIPS)* (2020).

50. W. Xu, *et al.*, Visual-tactile sensing for in-hand object reconstruction, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 8803–8812.

51. Y. Chen, A. E. Tekden, M. P. Deisenroth, Y. Bekiroglu, Sliding touch-based exploration for modeling unknown object shape with multi-fingered hands, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (IEEE, 2023), pp. 8943–8950.

52. M. Comi, *et al.*, TouchSDF: A DeepSDF Approach for 3D Shape Reconstruction using Vision-Based Tactile Sensing, *arXiv preprint arXiv:2311.12602* (2023).

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

54. A. S. Lambert, *et al.*, Joint inference of kinematic and force trajectories with visuo-tactile sensing, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2019), pp. 3165–3171.

55. P. Sodhi, M. Kaess, M. Mukadam, S. Anderson, Learning tactile models for factor graph-based estimation, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2021), pp. 13686–13692.

56. A. Petrovskaya, O. Khatib, Global localization of objects via touch, *IEEE Trans. on Robotics (TRO)* **27**, 569 (2011).

57. G. M. Caddeo, N. A. Piga, F. Bottarel, L. Natale, Collision-aware in-hand 6d object pose estimation using multiple vision-based tactile sensors, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2023), pp. 719–725.

58. K.-T. Yu, J. Leonard, A. Rodriguez, Shape and pose recovery from planar pushing, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (IEEE, 2015), pp. 1208–1215.

59. S. Suresh, *et al.*, Tactile SLAM: Real-time inference of shape and pose from planar pushing, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (2021).

60. C. Strub, F. Wörgötter, H. Ritter, Y. Sandamirskaya, Correcting pose estimates during tactile exploration of object shape: a neuro-robotic study, *4th International Conference on Development and Learning and on Epigenetic Robotics* (IEEE, 2014), pp. 26–33.

61. M. Lepert, C. Pan, S. Yuan, R. Antonova, J. Bohg, In-Hand Manipulation of Unknown Objects with Tactile Sensing for Insertion, *Embracing Contacts-Workshop at ICRA 2023* (2023).

62. J. Zhao, M. Bauza, E. H. Adelson, FingerSLAM: Closed-loop Unknown Object Localization and Reconstruction from Visuo-tactile Feedback, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2023), pp. 8033–8039.

63. Wonik Robotics, Allegro Hand (2023).

64. J. Tremblay, *et al.*, Diff-DOPE: Differentiable Deep Object Pose Estimation, *arXiv preprint arXiv:2310.00463* (2023).

65. Y. Xiang, T. Schmidt, V. Narayanan, D. Fox, PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes, *Proc. Robotics: Science and Systems (RSS)* (2018).

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

67. J. Tremblay, *et al.*, Deep object pose estimation for semantic robotic grasping of household objects, *Proc. Conf. on Robot Learning (CoRL)* (2018).

68. A. Knapitsch, J. Park, Q.-Y. Zhou, V. Koltun, Tanks and temples: Benchmarking large-scale scene reconstruction, *ACM Transactions on Graphics (ToG)* **36**, 1 (2017).

69. M. Tatarchenko, *et al.*, What do single-view 3D reconstruction networks learn?, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 3405–3414.

70. V. Makoviychuk, *et al.*, Isaac Gym: High performance GPU-based physics simulation for robot learning, *arXiv preprint arXiv:2108.10470* (2021).

71. Y. Labbé, *et al.*, Megapose: 6D pose estimation of novel objects via render & compare, *Proc. Conf. on Robot Learning (CoRL)* (2023).

72. J. Sun, Z. Shen, Y. Wang, H. Bao, X. Zhou, LoFTR: Detector-free local feature matching with transformers, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2021), pp. 8922–8931.

73. M. Bauza, A. Bronars, A. Rodriguez, Tac2Pose: Tactile Object Pose Estimation from the First Touch, *arXiv preprint arXiv:2204.11701* (2022).

74. P.-E. Sarlin, D. DeTone, T. Malisiewicz, A. Rabinovich, Superglue: Learning feature matching with graph neural networks, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 4938–4947.

75. F. Dellaert, M. Kaess, Factor Graphs for Robot Perception, *Foundations and Trends in Robotics* **6**, 1 (2017).

76. W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, *Seminal graphics: pioneering efforts that shaped the field* (1998), pp. 347–353.

77. M. Bauza, *et al.*, simPLE: a visuotactile method learned in simulation to precisely pick, localize, regrasp, and place objects, *arXiv preprint arXiv:2307.13133* (2023).

78. L. Keselman, K. Shih, M. Hebert, A. Steinfeld, Optimizing Algorithms From Pairwise User Preferences, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (2023).

79. A. Handa, T. Whelan, J. McDonald, A. J. Davison, A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM, *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)* (IEEE, 2014), pp. 1524–1531.

80. E. J. Smith, *et al.*, Active 3D Shape Reconstruction from Vision and Touch, *Proc. Conf. on Neural Information Processing Systems (NeurIPS)* (2021).

81. L. Pineda, *et al.*, Theseus: A Library for Differentiable Nonlinear Optimization, *Advances in Neural Information Processing Systems* (2022).

82. C. Cadena, *et al.*, Past, present, and future of Simultaneous Localization and Mapping: Toward the robust-perception age, *IEEE Trans. on Robotics (TRO)* **32**, 1309 (2016).

83. J. T. Barron, *et al.*, Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields, *Proc. Intl. Conf. on Computer Vision (ICCV)* (2021), pp. 5855–5864.

84. B. Kerbl, G. Kopanas, T. Leimkühler, G. Drettakis, 3D Gaussian splatting for real-time radiance field rendering, *ACM Transactions on Graphics (ToG)* **42**, 1 (2023).

85. J. Kerr, *et al.*, Self-Supervised Visuo-Tactile Pretraining to Locate and Follow Garment Features, *Proc. Robotics: Science and Systems (RSS)* (2023).

86. D. Driess, J.-S. Ha, M. Toussaint, R. Tedrake, Learning models as functionals of signed-distance fields for manipulation planning, *Proc. Conf. on Robot Learning (CoRL)* (PMLR, 2022), pp. 245–255.

87. C. Higuera, B. Boots, M. Mukadam, Learning to Read Braille: Bridging the Tactile Reality Gap with Diffusion Models, *arXiv preprint arXiv:2304.01182* (2023).

88. Z. Si, *et al.*, DiffTactile: A Physics-based Differentiable Tactile Simulator for Contact-rich Robotic Manipulation, *The Twelfth International Conference on Learning Representations* (2024).

89. E. Dexheimer, A. J. Davison, Learning a Depth Covariance Function, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2023), pp. 13122–13131.

90. J. A. Fishel, G. E. Loeb, Sensing tactile microvibrations with the BioTac—Comparison with human sensitivity, *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)* (IEEE, 2012), pp. 1122–1127.

91. C. Zhang, *et al.*, Faster Segment Anything: Towards Lightweight SAM for Mobile Applications, *arXiv preprint arXiv:2306.14289* (2023).

92. M. Kaess, *et al.*, iSAM2: Incremental smoothing and mapping using the Bayes tree, *Intl. J. of Robotics Research (IJRR)* **31**, 216 (2012).

93. D. DeTone, T. Malisiewicz, A. Rabinovich, Superpoint: Self-supervised interest point detection and description, *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (2018), pp. 224–236.

94. C.-Y. Wu, J. Johnson, J. Malik, C. Feichtenhofer, G. Gkioxari, Multiview Compressive Coding for 3D Reconstruction, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2023).

95. Y. Hong, *et al.*, LRM: Large Reconstruction Model for Single Image to 3D, *arXiv preprint arXiv:2311.04400* (2023).

96. B. Sundaralingam, T. Hermans, In-hand object-dynamics inference using tactile fingertips, *IEEE Transactions on Robotics* **37**, 1115 (2021).

97. S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, M. J. Marín-Jiménez, Automatic generation and detection of highly reliable fiducial markers under occlusion, *Pattern Recognition* **47**, 2280 (2014).

98. P. Furgale, J. Rehder, R. Siegwart, Unified temporal and spatial calibration for multi-sensor systems, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (IEEE, 2013), pp. 1280–1286.

99. T. Hodan, *et al.*, BOP: Benchmark for 6D object pose estimation, *Proc. Eur. Conf. on Computer Vision (ECCV)* (2018), pp. 19–34.

100. A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The KITTI dataset, *Intl. J. of Robotics Research (IJRR)* **32**, 1231 (2013).

101. S. James, Z. Ma, D. R. Arrojo, A. J. Davison, RLbench: The robot learning benchmark and learning environment, *IEEE Robotics and Automation Letters (RA-L)* **5**, 3019 (2020).

102. Revopoint, Revopoint POP 3 3D Scanner (2023).

103. B. Calli, *et al.*, Yale-CMU-Berkeley dataset for robotic manipulation research, *Intl. J. of Robotics Research (IJRR)* **36**, 261 (2017).

104. S. Brahmbhatt, A. Handa, J. Hays, D. Fox, ContactGrasp: Functional multi-finger grasp synthesis from contact, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (IEEE, 2019), pp. 2386–2393.

105. A. Rosinol, J. J. Leonard, L. Carlone, NeRF-SLAM: Real-time dense monocular SLAM with neural radiance fields, *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (2023).

106. D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, J. Thies, Neural RGB-D surface reconstruction, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 6290–6301.

107. M. Tancik, *et al.*, Fourier features let networks learn high frequency functions in low dimensional domains, *Advances in Neural Information Processing Systems* **33**, 7537 (2020).

108. J. L. Schonberger, J.-M. Frahm, Structure-from-Motion revisited, *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 4104–4113.

109. A. Kirillov, *et al.*, Segment anything, *Proc. Intl. Conf. on Computer Vision (ICCV)* (2023), pp. 4015–4026.

110. R. Ambrus, *et al.*, Monocular Depth Estimation for Soft Visuotactile Sensors, *Proc. IEEE Intl. Conf. on Soft Robotics (RoboSoft)* (2021).

111. R. Ranftl, A. Bochkovskiy, V. Koltun, Vision transformers for dense prediction, *Proc. Intl. Conf. on Computer Vision (ICCV)* (2021), pp. 12179–12188.

112. A. Ross, V. L. Willson, A. Ross, V. L. Willson, Paired samples T-test, *Basic and Advanced Statistical Tests: Writing Results Sections and Creating Tables and Figures* pp. 17–19 (2017).

113. A. Dosovitskiy, *et al.*, An image is worth 16x16 words: Transformers for image recognition at scale, *International Conference on Learning Representations* (2021).

114. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *International Conference on Learning Representations (ICLR)* (2015).

115. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347* (2017).

Acknowledgments: We thank Dhruv Batra, Theophile Gervet, Akshara Rai for feedback on the writing, and Wei Dong, Tess Hellebrekers, Carolina Higuera, Patrick Lancaster, Franziska Meier, Alberto Rodriguez, Akash Sharma, Jessica Yin for helpful discussions on the research. **Funding**: S. S. and H. Q. acknowledge funding from Meta, and their work was partially conducted while at FAIR, Meta. S. S. was further partially supported by NSF grant IIS-2008279 while at CMU. R. C. acknowledge support from the German Research Foundation (DFG, Deutsche Forschungsgemeinschaft) as part of Germany's Excellence Strategy – EXC 2050/1 – Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden, and from Bundesministerium für Bildung und Forschung (BMBF) and German Academic Exchange Service (DAAD) in project 57616814 (School of Embedded and Composite AI). **Author contributions**: S. S. developed and implemented the core approach including tactile transformer, visual depth segmentation, neural SDF reconstruction, pose-graph optimization, performed full-stack tuning, worked on Allegro and DIGIT integration, TACTO and IsaacGym integration, camera and robot calibration, data collection, ground truth object scans, and live visualizations, conducted evaluations, made visuals, and wrote the paper. H. Q. designed and implemented in-hand object rotation policies and sim-to-real policy transfer, helped with Allegro and DIGIT integration, TACTO and IsaacGym integration, data collection, did code reviews and bug fixes, and helped edit the paper. T. W. coordinated hardware and software systems integration, performed profiling of software stack, helped with Allegro and DIGIT integration, camera and robot calibration, ground truth object scans, and advised on evaluations. T. F. designed and implemented forward kinematics, helped implement visual depth segmentation, pose-graph cost functions and optimization, software systems integration, and advised on evaluations. L. P. implemented workflow for cluster deployment, streamlined development workflow, helped with modules that use Theseus, did code reviews and bug fixes, and advised on evaluations. M. L. helped with Allegro and DIGIT integration, TACTO and IsaacGym integration, and hardware systems integrations. J. M. advised on the project, gave feedback

on approach, evaluations, and the paper. Mr. K. advised on the project, managed and supported researchers, gave feedback on approach, evaluations, and the paper. R. C. advised on the project, helped with Allegro and DIGIT integration, TACTO and IsaacGym integration, gave feedback on approach, evaluations, and the paper. Mi. K. advised on the project, helped design pose-graph optimization, gave feedback on approach, evaluations, and the paper. J. O. advised on the project, co-developed the core approach, implemented volumetric ray sampling, SDF cost function, and 2D live visualizations, helped implement workflow for cluster deployment, streamlined development workflow, did code reviews and bug fixes, gave feedback on evaluations, designed visuals, and edited the paper. M. M. set the vision and research direction, steered and aligned the team, provided guidance on all aspects of the project including core approach, systems, and evaluations, designed visuals, and edited the paper. **Competing interests**: The authors declare that they have no competing interests. **Data and materials availability**: All the data to validate the paper is available in the main body and supplementary materials. For multimedia, code, and data we refer the readers to the project webpage https://suddhu.github.io/neural-feels. You may also access the code and data through https://doi.org/10.5061/dryad.b2rbnzsqr.

Supplementary Materials for

Neural feels with neural fields: Visuo-tactile perception for in-hand manipulation

Sudharshan Suresh,[1,2*] Haozhi Qi,[2,3] Tingfan Wu,[2] Taosha Fan,[2]
Luis Pineda,[2] Mike Lambeta,[2] Jitendra Malik,[2,3] Mrinal Kalakrishnan,[2]
Roberto Calandra,[4,5] Michael Kaess,[1] Joseph Ortiz,[2] Mustafa Mukadam[2]

[1]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[2]FAIR, Meta, 1 Hacker Way Menlo Park, CA 94025, USA
[3]Department of Electrical Engineering and Computer Sciences, UC Berkeley, CA 94720, USA
[4] Institute of Artificial Intelligence, Technische Universität Dresden, 01062, Dresden, Germany
[5]The Centre for Tactile Internet with Human-in-the-Loop (CeTI), 01062, Dresden, Germany

*To whom correspondence should be addressed; E-mail: suddhus@gmail.com.

This PDF includes

Seven supplementary sections

Fig. S1 to S24

Table S1 to S4

Movie S1 to S3

Ground-truth shape and pose

Ground-truth object scans

To compute our results we require ground-truth object shape to compare against. For this, we use a commercial dual-camera infrared scanner, the Revopoint POP 3 (*102*). The hardware can scan objects from a close range with a minimum precision of 0.05 mm. Each real-world object is placed on a turntable and scanned while rotating about its axis (Fig. S1 (a)). For object's that lack texture, an artificial dot pattern is tracked by adding stickers. After generating the scans, we perform hole-filling for unseen regions of the object, like the bottom. Fig. S1 (b) shows all the scanned meshes—some meshes are directly sourced from the YCB (*103*) and ContactDB (*104*)

object sets.

Pseudo ground-truth pose

In the real-world, we pass three RGB-D cameras as input into our pose tracking pipeline to use as a pseudo ground-truth estimate. This consists of three unique cameras (front left, back right, top down) with complementary but overlapping fields-of-view (Fig. S2 and Fig. S3 (b)). With this broad perspective of the scene, known shape from ground-truth scans, and the tracker running at 0.5Hz, we can obtain accurate tracking of object pose at each timestep. For the tracker to perform well, the initial orientation of the object is chosen through manual 3D annotation.

Tactile transformer: data and training

Model architecture

Our model architecture is based on a monocular depth network, the dense prediction transformer (DPT) (*111*). It comprises of a vision transformer (ViT) backbone that outputs bag-of-words features at different resolutions, finally combined into a dense prediction via a convolutional decoder. Compared to fully-convolutional methods, DPT has a global receptive field and the resulting embedding does not explicitly down-sample the image.

Training and loss metric

Our image-to-depth training dataset comprises of 10K simulated tactile interactions each on the surface of 40 YCB objects. We illustrate examples of the interactions in Fig. S4 (b). We use the ADAM optimizer with momentum and a batch size of 100, trained with mean-square depth reconstruction loss (Fig. S4 (a)). We start with a pre-trained small ViT (*113*), with an embedding dimension of 384 patch size of 16. The dataloader splits the train, test, and validation data into 60%, 20%, and 20% respectively. To supplement our results in the "Frontend" section of Materials and Methods of the main text, we visualize more simulation results in Fig. S5.

Data augmentation

An important aspect of generalization and sim-to-real transfer is the augmentation applied during

data collection and training. These include composing simulated renderings with real-world background images, collected from 25 different DIGIT sensors. These are shown in Fig. S4 (c). Before rendering a sensor pose, we add pose variations by applying noise in rotation/translation and sensing normal direction. Additionally, we randomly vary the distance of penetration into the object surface. We create lighting augmentations by randomizing position, direction and intensity of the three DIGIT LEDs. We add Gaussian noise to RGB data, with a standard deviation of 7px. Finally, we use randomized horizontal flipping, cropping, and rotations of the tactile images.

Segmented visual depth

In our experiments, SAM would often over-segment objects with distinct parts (like different faces of the Rubik's cube). In case of these ambiguities, SAM outputs candidate masks at different spatial scales. We apply a pruning step to select the mask prediction closest to the average mask area we typically observe in simulation experiments. Finally, for the RGB image at $t = 0$ we will not have an accurate object model to check fingertip occlusion against. In this case, we apply a distance-based heuristic to verify if the fingertips are in front or behind the object. Fig. S6 compares the benefit of using our prompting scheme versus a simple positive prompt on the object. With information on robot finger pixels, SAM can better singulate the object without false positives.

Shape optimizer

The neural field is optimized via Adam (*114*) with learning rate of 2e-4 and weight decay of 1e-6. Instant-NGP uses a hash table of size 2^{19} for positional encoding, followed by a 3-layer MLP with 64 dimensional width. We use a uniform random weights θ_{init} and initialize the SDF by running 500 shape iterations using the first keyframe $\mathcal{K}_0$.

For evaluating the neural field we freeze the network and query a 200^3 feature grid. The feature grid's extents are defined as a bounding box of 150 mm side, centered at the object's initial pose $\mathbf{x}_0$. When training, we apply a series of bounding-box checks post hoc, to eliminate any

ray samples P_u found outside this bounding box. Mesh visualizations (Fig. 4) are periodically generated via marching-cubes on the feature grid. We add color to the output mesh by averaging the colored object pointcloud with a Gaussian kernel.

Pose optimizer

We use the vectorized $SE(3)$ pose graph optimizer in Theseus (*81*), with 20 LM iterations of step size 1.0. The keyframe window size $n = 3$ and we run 2 pose iterations for each shape iteration. The weighting factors for each loss are $w_{\text{sdf}} = 0.01$, $w_{\text{reg}} = 0.01$, and $w_{\text{icp}} = 1.0$.

Keyframing

The frontend output from each timestep t is considered for keyframing, as illustrated in the flowchart Fig. S7. Two checks are performed to determine if the current frame will be added to the set. This is based on prior work in continual learning, such as iSDF (*31*) and iMAP (*32*). An Information Gain Check verifies if the average rendering loss of the object SDF with respect to the frontend depth is greater than a defined threshold d_{thresh}. If true, this set of visuo-tactile frames is directly added to the keyframe set $\mathcal{K}$. This ensures that any new depth information, such as a new face of a cube, is directly considered a keyframe. An Image Time Check verifies if the time since the last keyframe was added is greater than the maximum time threshold t_{max}. This will force the visuo-tactile frames to be added to $\mathcal{K}$, which ensures there is a steady temporal stream of keyframes for the optimizer.

If neither check is passed, the visuo-tactile frames are added to backend optimizer at the current timestep t and then immediately discarded. This ensures that we always use the most recent information in the backend, but don't add redundant frames. In our experiments $d_{\text{thresh}} = 0.01\,\text{m}$ and $t_{\text{max}} = 0.2\,\text{secs}$.

Compute and timings

All statistics in the "Results" section of the main text are generated from playing-back the trials at a publishing rate of $1\,\text{Hz}$. Experimentally, however, we can run the pose optimizer at $10\,\text{Hz}$

and full backend at 5 Hz. Fig. S3 (a) has a minimal robot setup of an online SLAM system with rotation policy in-the-loop. The policy runs at 2Hz, so the pose optimizer does not deal with large object motions. Experiments are run on an Nvidia GeForce RTX 4090, while the aggregate results are evaluated on a cluster with Nvidia Tesla V100s.

In-hand rotation policy

Policy training

We first train a policy in simulation with access to an embedding of physical properties such as object position, size, mass, friction, and center-of-mass (denoted as $\mathbf{z}_t$). From the joint-angles $\mathbf{q}_t$ and this embedding $\mathbf{z}_t$, the policy outputs a proportional-derivative (PD) controller target $\mathbf{a}_t \in \mathbb{R}^{16}$. The policy is trained in parallel simulated environments (*70*) using proximal policy optimization (*115*). The reward function is a weighted combination of a rotational reward, joint-angle regularizer, torque penalty, and object velocity penalty. The resulting policy can adaptively rotate objects in-hand according to different physical properties.

During deployment, however, the policy does not have access to these physical properties. The estimator is instead trained to infer $\mathbf{z}_t$ from a history of proprioceptive states, which is in turn fed into the policy π_t. A crucial change compared to Qi et al. (*11*) is that we train the policy to rotate objects with DIGIT sensors on the distal ends (Fig. 1). This results in different gaits, as it relies on finger-object friction instead of gravity, and learns to maintain contact with the DIGIT elastomer.

The objective (reward) used to train the policy contains the rotation reward and several penalties to encourage the smoothness and energy efficiency. Specifically, if we define $\boldsymbol{\omega}$ as the angular velocity of the object, $\hat{\mathbf{k}}$ as the desired rotation axis, $\mathbf{q}$ as the robot's joint positions, $\boldsymbol{\tau}$ as the commanded torque for the robot, and $\boldsymbol{v}$ for the object's linear velocity, we have the rewards:

$$r \doteq r_{\text{rot}} + \lambda_{\text{pose}}\, r_{\text{pose}} + \lambda_{\text{linvel}}\, r_{\text{linvel}} + \lambda_{\text{work}}\, r_{\text{work}} + \lambda_{\text{torque}}\, r_{\text{torque}}$$

where $r_{\text{rot}} \doteq \max(\min(\boldsymbol{\omega} \cdot \hat{\mathbf{k}}, r_{\max}), r_{\min})$ is the rotation reward, $r_{pose} \doteq -\|\mathbf{q} - \mathbf{q}_{init}\|_2^2$ is the

hand pose deviation penalty, $r_{torque} \doteq -\|\boldsymbol{\tau}\|_2^2$ is the torque penalty, $r_{\mathrm{work}} \doteq -\boldsymbol{\tau}^T \dot{\boldsymbol{q}}$ is the energy consumption penalty, and $r_{\mathrm{linvel}} \doteq -\|\boldsymbol{v}\|_2^2$ is the object linear velocity penalty.

Our policy generalizes to objects seen outside training provided they have similar aspect ratios. It is trained in large-scale simulation with primitive shapes: spheres and cylinders of different sizes, with aspect ratio 1 about the z-axis. Concretely, the length:diameter aspect ratios for the cylinders are: $(0.8:1, 0.85:1, 0.9:1, 1:1, 1:1.05, 1:1.1, 1:1.15, 1:1.2)$ These work best in the real-world given the morphology of the hand and sensor positioning. More irregular aspect ratios are harder to manipulate, but we do show it to work with some objects in simulation like the rubber duck and elephant.

Policy success rate

We use time-to-fall (seconds before objects fall down) to measure the success of our policy. This is widely used in the in-hand manipulation community (*10, 11*). We define the trial to be successful if the time-to-fall for an object is greater than 20 seconds (episode length). We measure this metric in simulation across 8 objects in the FeelSight dataset. For each object, we run 5000 trials and each trial is with randomized physical parameters (mass, center of mass, coefficient of friction, and controller gains) and initial grasp configurations. We see an average success rate is 73.87 ± 17.48% across all objects and detailed statistics are in Table S1.

Although it is hard to quantify the success rate in the real-world deployment, we qualitatively note that the objects such as the bell pepper, peach, pear, and pepper grinder, can be rotated with a high success rate. As previously stated, the policy finds it harder to rotate cubes—the Rubik's cube and large dice—due to the hand morphology. This is an interesting failure in sim-to-real transfer, as these objects have the highest simulation success rates (92.78% and 84.32%).

Resulting tactile signal in the real-world

In the "Discussion" section of the main text, we highlight the sim-to-real gap in the contact patches, which are much sparser on the real robot. The policy rate (20Hz) and low-level PD

control gain can affect the forces applied, and thus the indentation of the Digit. With a more aggressive policy (and larger PD gains), it is possible that the contact may be more pronounced. But from our experiments, this negatively affects the stability of rotation. We empirically tune these so as to get best performance in terms of rotation stability (time-to-fall). Another option would be to consider a more elastic tactile sensor, that deforms easier, such as the GelSight (*16*). However, they are susceptible to more wear and tear through the experiments, and may not give consistent results.

The role of touch

Sensor coverage in SLAM

To illustrate the complementary nature of touch and vision, we color the reconstructed mesh regions based on their dominant sensing modality in Fig. S8. After running the experiments in the "Neural SLAM: object tracking and shape estimation" section of the Results in the main text , we first run marching-cubes on the final neural SDF. In the resultant mesh, we assign each vertices color based on if vision or touch is the nearest pointcloud measurement to it. In the case where there is no vision or touch pointcloud within a 5 mm radius, it is assigned as a hallucinated vertex. This is a demonstrable advantage of neural SDFs, where the network can extrapolate well based on information in the neighborhood of the query point. From the meshes in Fig. S8 we see that while vision gets broad coverage of each object, there is considerable tactile signal from the interaction utilized for shape estimation.

In Fig. S9 and S10, we plot the shape completion (recall) and accuracy (precision) percentages for touch and vision. These are averaged over 5 trials of a Rubik's cube experiment in simulation and the real-world. We obtain more complete shape reconstructions much quicker when incorporating touch (Fig. S9). This is important as a more complete model will leads to better pose tracking, as shown in Fig. 3. We observe a similar trend in shape accuracy, or precision. While the % starts out similarly, the optimization of tactile depth lets us get a more accurate reconstructions in both simulation and the real-world.

Touch aligns local geometries with predicted depth

As described in the "Backend: shape and pose optimizer" section of Materials and Methods in the main text, the pose optimizer inverts the neural field to back-propagate a loss in pose space (*32, 33, 37*). This has been illustrated in work such as iNeRF (*37*), where the rendered neural field attempts to match the image measurements via updates to the se(3) Lie algebra of the camera pose. As our framework leverages the idea that vision-based touch is just another perspective camera, we show how the rendered neural field matches with tactile depth features in Fig. S11.

Each RGB image is first passed through the tactile transformer (Section "Frontend" of Materials and Methods in the main text) to output a predicted tactile depthmap. Our pose optimizer aligns the neural rendering of the surface with the measured depthmap, based on 3D samples from the measured depthmap. Thus we can see that both in simulation (Fig. S11 (a)) and the real-world (Fig. S11 (b)), the edge and patch features predicted match well with the rendered object.

Vision-based touch versus binary sensing

We would like to understand how vision-based touch contributes in our solution, when compared to standard binary touch sensing. We approximate binary contact by empirically thresholding the tactile images from the existing dataset. We compute the pixelwise difference between an input tactile image and its non-contact background. This is then converted to grayscale, and we calculate the number of pixels that exceed a set threshold of 10 pixels. If the number of such pixels exceeds 1000 (¿ 1% of the image size), we consider it to be a contact event. We then downsample the image to 2×2 (our SDF sampling method prevents us from using point contacts that are 1×1) and set all depth values in the downsampled image to the maximum detected depth. In Fig. S12 we compare the contact output of our tactile transformer and the binary approximation.

We consider 5 trials from a subset of SLAM experiments in simulation and real-world (Ru-

bik's cube and bell pepper). The average pose and shape error are shown in Table S2 and S3—we see that with the DIGIT sensor we can obtain a lower pose error ($p < 0.001$) and higher shape metric ($p < 0.001$). We posit that, since shape estimation is crucial to the SLAM problem, a larger field-of-view is important for the tactile sensor. In addition, this can validate the benefit of our tactile transformer network in detecting contact patches and masks.

Benefits of high-resolution touch

Although our method does not explicitly rely on the high-resolution of the vision-based touch sensor, we conduct an ablation to measure the benefits of more pixels. Specifically, we perform nearest-neighbor interpolation of the input image/depth images to downsample them by factors of $2\times$, $4\times$, $8\times$. This results in images with resolutions 240×320, 120×160, 60×80, and 30×40 respectively (Fig. S13 (a)). This simulates lower resolution vision-based touch, and is also a coarse approximation of tactile-array based sensing (*90*).

With these different resolutions, we run 5 visuo-tactile SLAM trials of the real-world bell pepper (from Table S2 and S3). Our final comparison point is against the previous binary contact approximation. These results are shown in Fig. S13 (b). We see a gradual deterioration of shape/pose metrics with lower resolution, ending with a sharp drop for the binary contact. These results show that our pixel-sampling based loss works well for a range of image resolutions, but prefers higher resolutions. This is of greater importance when performing fine-manipulation such as tactile insertion (*61*) which is out-of-scope in this work.

Additional results

Shape and pose metrics over time

In Fig. S14, we plot these metrics for each of the experiments in Fig. 4, instead against $0-30$ sec timesteps. For shape, we observe gradual convergence to an asymptote close to 1.0, indicating evolution of both shape completion and refinement over time. Also visualized here is the precision and recall metrics over time, whose harmonic mean represents the F-score. For pose, we

observe stable drift over time, indicating the estimated object pose lies close to the ground-truth estimate.

Effect of camera viewpoint in the real-world

In the results section titled "Perceiving under duress: occlusion and visual depth noise" of the main text, we establish the relationship between occlusion/sensing noise and pose error. Here, we run additional experiments, on a limited set of viewpoints in the real-world. Fig. S3 (b) shows the RGB-D data from three cameras: front left, back right, top down, at distances of 27 cm, 28 cm, and 49 cm respectively from the robot. We run our vision-only pose tracker with known shape using each of three cameras over all 5 Rubik's cube rotation experiments and plot the average metrics in Fig. S3 (c). We observe that the front left and back right viewpoints result in lowest average pose error due to their closer proximity. The top down camera gives less reliable depth measurements and segmentation output, leading to almost 2x greater pose error.

Class-specific metrics

In Fig. S15, we present additional metrics for the "Neural SLAM: object tracking and shape estimation" section of the Results in the main text, dividing based on object class. This helps us make some assessments on how object geometry and scale can affect our results. We first see that objects with symmetries about their rotation axis are challenging for our depth-based estimator. This leads to higher pose errors for the peach and pear, for example. Additionally, partial visibility of the large objects, such as the pepper grinder, affect the completeness of the reconstructions. Touch in this case is not advantageous since the finger gait does not span the length of the object to provide coverage. Finally, smaller-sized objects, such as the peach, may demonstrate better shape metrics as their scale is closer to the F-score threshold of 5 mm.

Better shape leads to better tracking

To highlight the importance of a good shape estimate in object tracking, we perform an ablation over the ground-truth mesh quality. For the Rubik's cube tracking experiment (from the "Neural

SLAM: object tracking and shape estimation" section of the Results in the main text), we voxelize the ground-truth mesh at different resolutions and feed the resulting SDF to our tracker. These resolutions vary from our standard 0.5mm all the way to 10mm, and their renderings are shown with the graph. Our SDF-based tracker yields acceptable results at even coarse voxelization, but the expected trend is corroborated. This serves to motivate the need for good object priors when the object shape is unknown.

Object pose drift

Fig. S17 and S18 supplement the results on object pose drift in results section titled "Neural SLAM: object tracking and shape estimation" of the main text. Empirically, we observe a large pose drift in the first few seconds from initialization at ground-truth, due to unknown shape. Over time, we build a better shape model, resulting in more accurate pose tracking. However, in the absence of long-term loop closures (*72, 74*), small errors in pose will accumulate over time. We visualize two experiments and their corresponding errors in Fig. S18—we see that that the poor visibility of the object is reflected in final pose metric.

Importance of the neural SDF loss

We compare a version of our SLAM system with ICP and the pose regularizer, lacking our frame-to-model neural SDF loss. Without this module, the pose regularizer ensures the object's motion is smooth while the ICP factor provides updates at each timestep based on the visuo-tactile keyframes. We run this on 5 trials of the real-world Rubik's cube SLAM experiment, averaging the shape and pose statistics in Table S4 ($\mathbf{p} < 0.001$)

We see that ICP method fails in our comparison, accruing large pose errors and deteriorating final shape metric. A side-by-side comparison of the visuo-tactile SLAM results are shown in Fig. S19. This is expected due to the lack of any frame-to-model constraints that allow us to match against the shape model we are building. Additionally, ICP performs poorly when there is little overlap between the keyframes (such as two different sides of the cube), is highly

susceptible to local minima, and requires careful tuning.

Additional visualizations

All experiments from the FeelSight dataset

In Fig. S20 we illustrate all of the 70 visuo-tactile experiments that comprise our dataset, further shown as individual sequences in Fig. S21. While both simulation and real data collection use the proprioception-driven policy (*11*), the policy generalizes better in simulation across the class of objects. Some objects in the real require a human-in-the-loop to assist with in-hand rotation; like supporting cube-shaped objects from the bottom occasionally to prevent it from falling out of hand.

Additional neural tracking visualizations

Fig. S22 shows rendering results along with the pose axes from the "Neural SLAM: object tracking and shape estimation" section of the Results in the main text. We see good alignment of the renderings when overlaid on the RGB camera frame.

Further visual segmentation results

Fig. S23A and B shows additional qualitative results of visual segmentation for real-world and simulated rotations sequences. Failure modes of the segmentation are captured in Fig. S24. These include partial segmentation of complex objects (potted meat can, rubber duck), partial segmentation objects that cannot be distinguished from the robot hand (pepper grinder), and oversegmentation of the faces of the Rubik's cube.

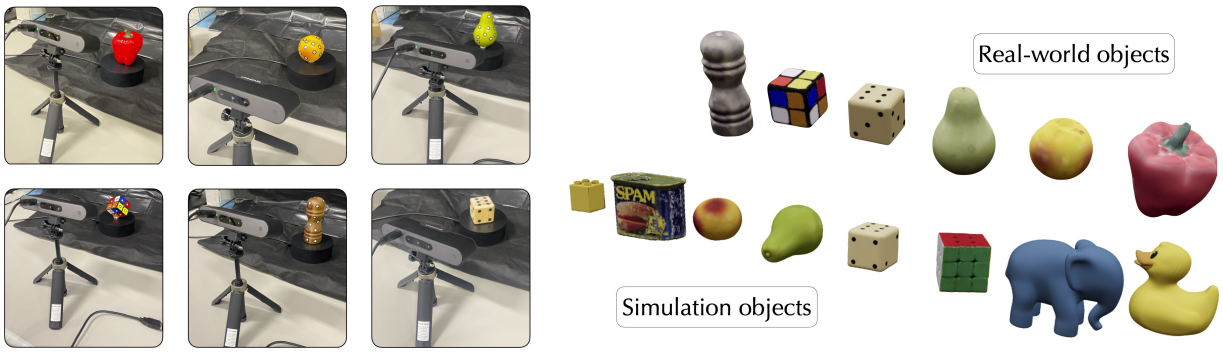
Supplementary movies

Movie S1 shows online neural SLAM over a real-world experiment with the bell pepper object. It shows the input stream of RGB-D and tactile images, paired with the posed reconstruction. We partially reconstructed the object at the initial frame, and built the surfaces out progressively over each 30 second experiment. The 3D visualizations are generated by marching-cubes, in addition

to the rendered normals of the neural field projected onto the visual image. The rendering was textured by mapping the surface normal directions to a red-green-blue (RGB) colormap.

Movie S2 shows online neural SLAM over a simulated experiment with the rubber duck object. It shows the input stream of RGB-D and tactile images, paired with the posed reconstruction. We partially reconstructed the object at the initial frame, and built the surfaces out progressively over each 30 second experiment. The 3D visualizations are generated by marching-cubes, in addition to the rendered normals of the neural field projected onto the visual image. The rendering was textured by mapping the surface normal directions to a red-green-blue (RGB) colormap.

Movie S3 is an 11-minute long explanation of our work with narration. It summarizes the key results, discusses motivation and prior work, and goes through the methodology.



(a) Ground-truth object scanning **(b)** Resulting ground-truth meshes

Figure S1: Object ground-truth with dual-camera infrared scanner. **(A)** Objects are placed on a turntable and scanned, followed by post-processing to ensure complete, accurate meshes. **(B)** Meshes visualized for the real and simulated FeelSight objects.

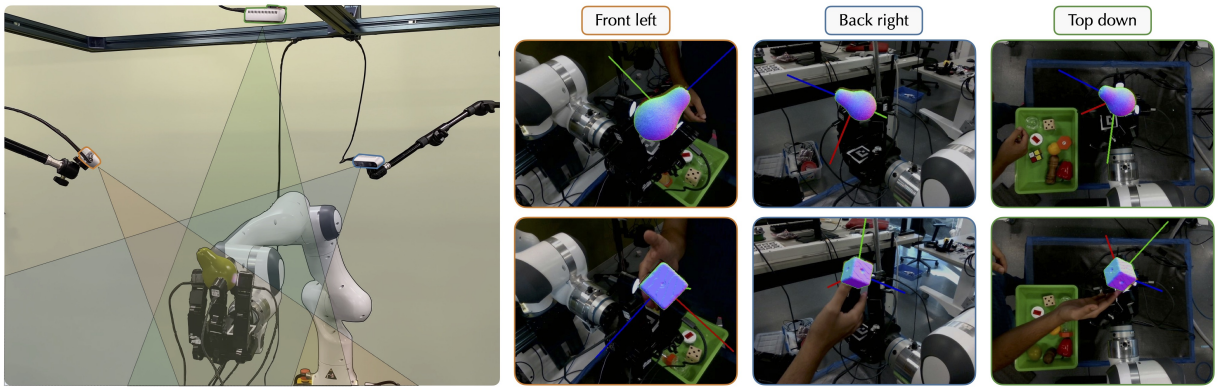
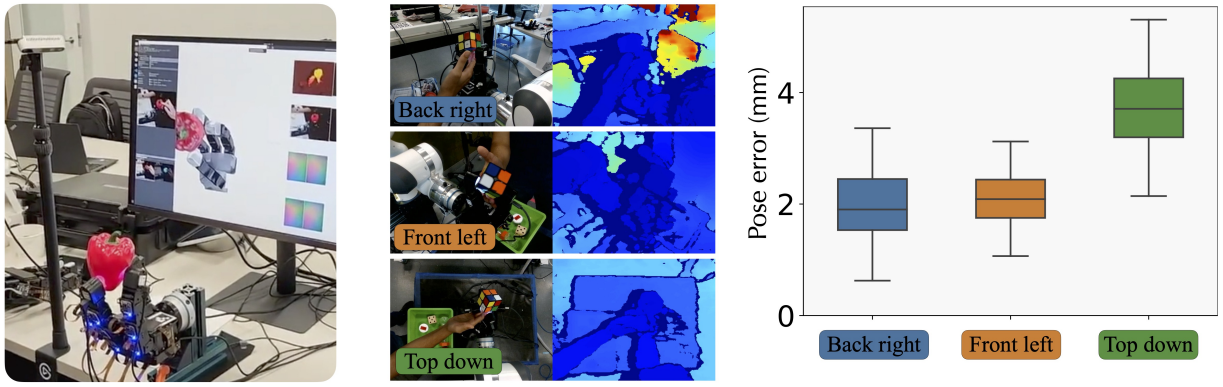
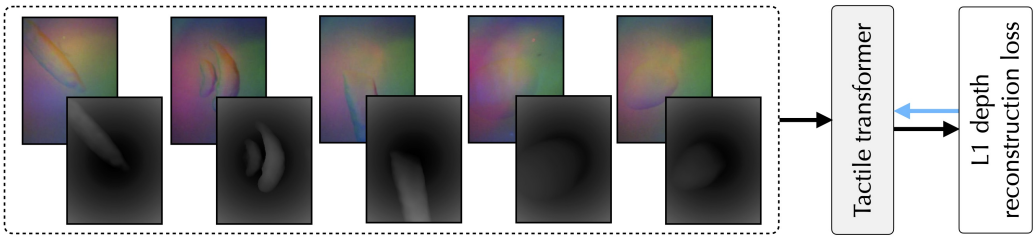


Figure S2: Robot cell for pseudo-ground-truth tracking. Each of the three camera's captures a different field-of-view of the interaction. For a pseudo-ground-truth, we pass the RGB-D stream from all three cameras into our pipeline, with known shape obtained from scanning. The output pose tracking represents the ground-truth we compare to in the real-world results.

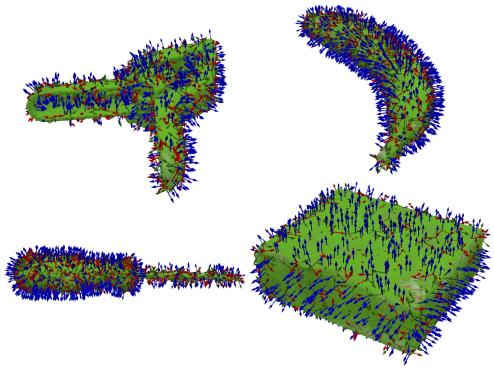


(a) Example of a minimal robot cell **(b)** RGB-D views from cameras **(c)** Pose errors based on camera viewpoint

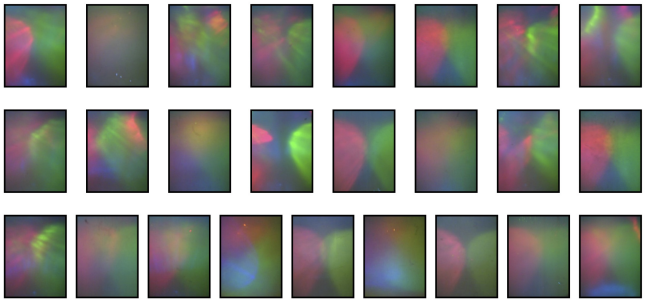
Figure S3: (A) As proof-of-concept, we assembled a minimal robot cell for demonstrating our method with one RGB-D camera and the policy deployed at 2Hz. **(B)** The three different RGB-D viewpoints in our full robot cell used to collect FeelSight evaluation dataset. **(C)** Average pose error for known shape experiments based on camera viewpoint. In each boxplot, the central line is the median, extents of the box are the upper and lower quartiles, and the whiskers represent $1.0\times$ the interquartile range (IQR). We see that the front (27cm) and back (28cm) cameras perform comparably, and there are larger errors in the top-down (49cm) camera as it is further away.



(a) Tactile transformer training with simulation rendered image-depth pairs



(b) Examples of posed tactile samples on YCB objects in TACTO



(c) 25 real DIGIT background images used for data augmentation

Figure S4: Our tactile transformer is trained in simulation with real-world augmentation. **(A)** The tactile transformer is supervised from paired RGB-depth images rendered in TACTO (*24*). **(B)** Each of these samples are generated from dense, random interactions with 40 different YCB objects. **(C)** In our training, we augment the data with background images collected from 25 unique DIGIT sensors (*20*).

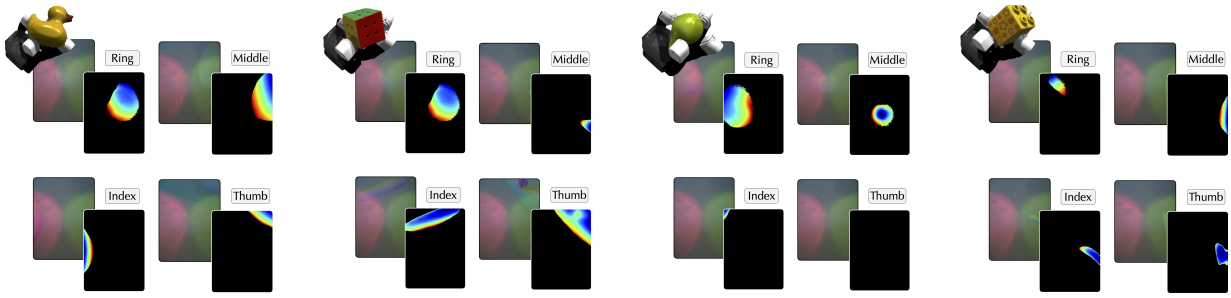


Figure S5: Image to depth predictions by the tactile transformer on simulated contacts. Our tactile transformer shows good performance in simulated interactions, capturing both large contact patches, as well as smaller edge features. These objects are unseen during training, but we demonstrate an average prediction error of 0.042 mm on simulated test images (refer to the "Frontend" section of Materials and Methods in the main text).

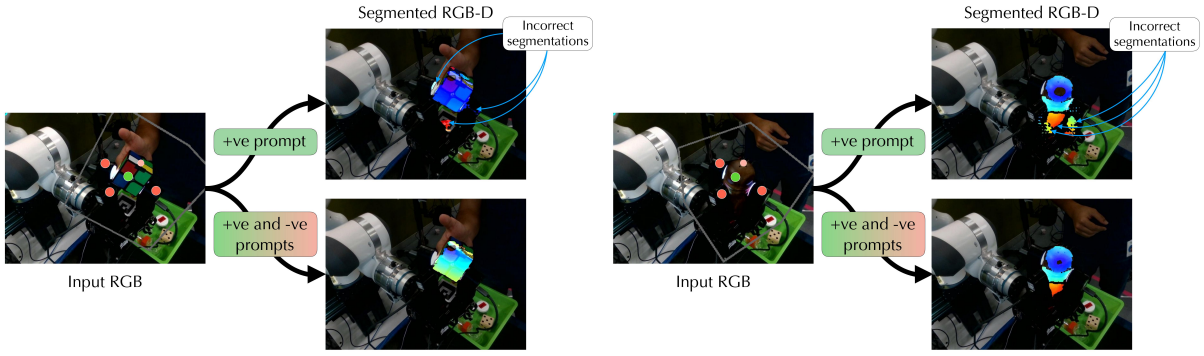


Figure S6: SAM prompts for object segmentation. With our segmentation method in Fig. 8 (A), we can robustly singulate the object in-hand from positive and negative prompts. Without knowledge of robot finger pixels, the output often contains false positives. By rendering our shape estimate in the camera frame, we can discard any negative prompts that are occluded by the object.

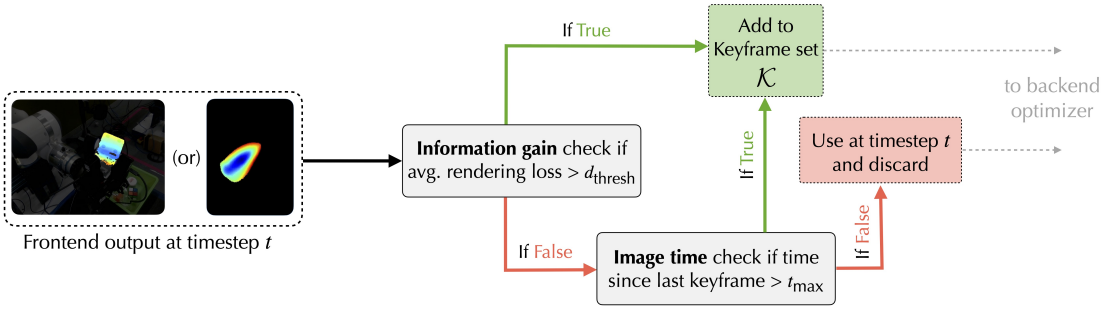


Figure S7: Keyframing schema. Based on prior work (*31, 32*) our keyframing strategy has two different checks for an input set of visuo-tactile frames. The first determines if the SDF rendering loss with respect to the current depth frame is high. If not, we verify if there has been a keyframe added in the last t_{max} seconds, else we force a new keyframe. The keyframe set $\mathcal{K}$ is then used by the shape and pose optimizers (refer to the "Backend: shape and pose optimizer" section of Materials and Methods in the main text)

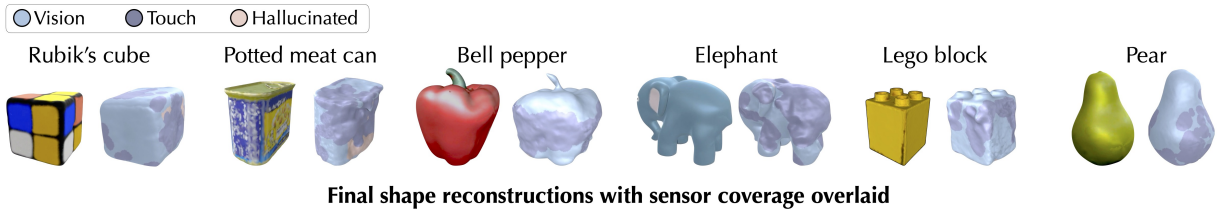


Figure S8: Sensor coverage illustrated in final mesh reconstructions of select objects—indicating vision, touch, and hallucinated regions.

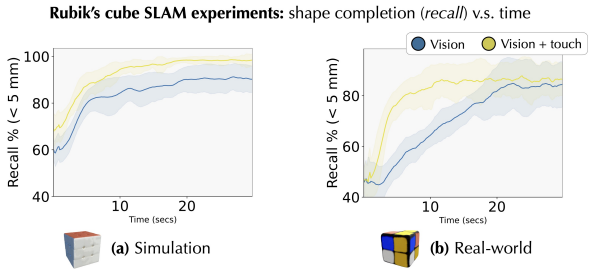


Figure S9: Shape completion with vision and touch, average over five Rubik's cube trials. The shaded regions represent one standard deviation from the mean recall. We see that incorporating touch gives a more complete shape model quicker, which is better for pose tracking.

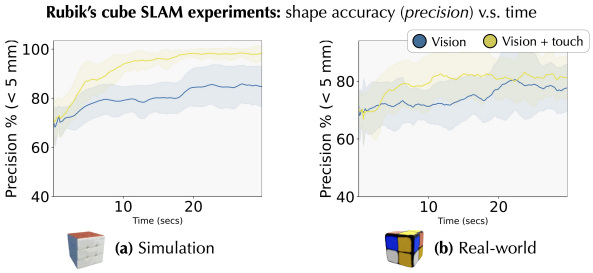


Figure S10: Shape accuracy with vision and touch, over five Rubik's cube trials. The shaded regions represent one standard deviation from the mean precision. Although vision and touch start the same, we get more accurate over time through optimization of tactile depth.

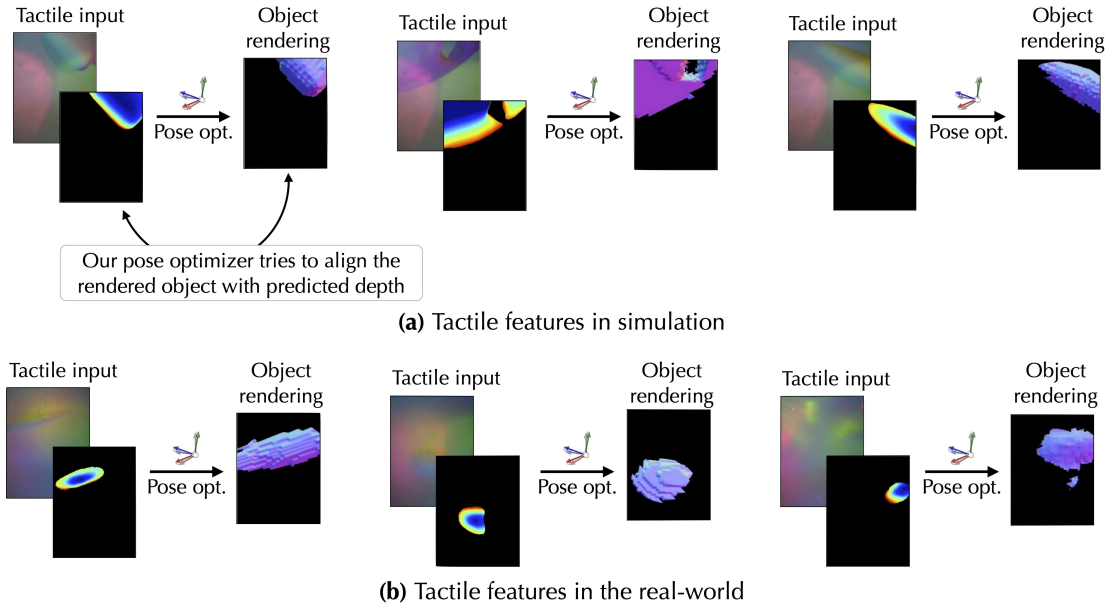


Figure S11: Six examples of tactile images compared against the neural field. We see that our tactile pose optimizer matches the predicted local geometry with the neural surface rendering. Thus, patches and edges predicted by touch appear in the rendering as well.

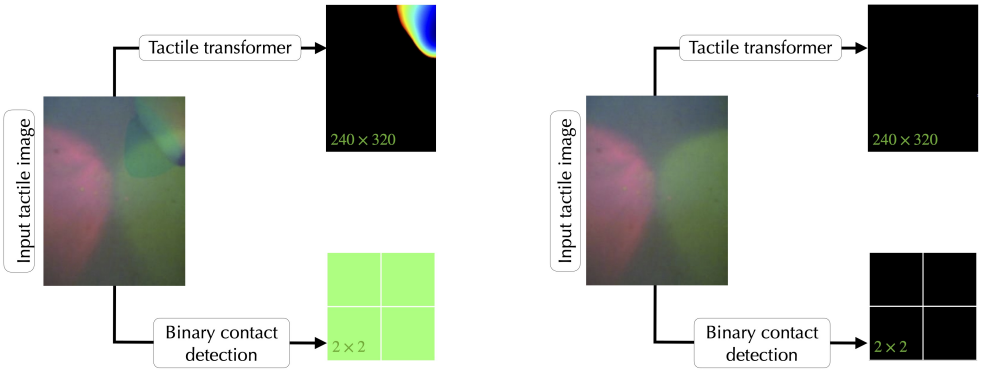
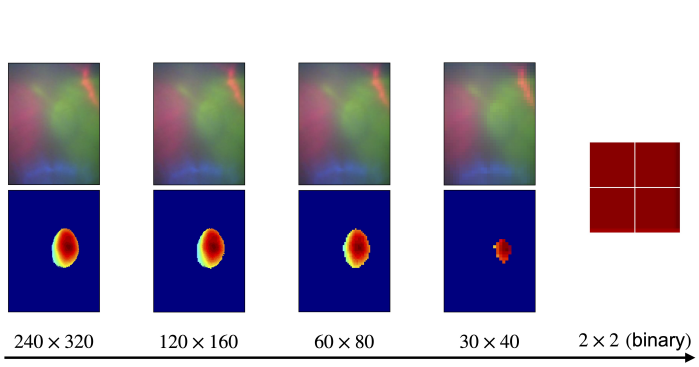
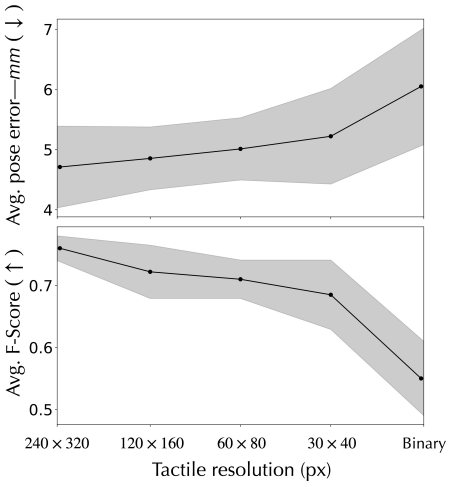


Figure S12: Vision-based touch versus binary contact. We use image thresholding for an approximate binary contact, which we use to compare against our tactile transformer output in SLAM experiments. Our SDF sampling method prevents us from using point contacts (1×1), so instead upsample them by a factor for 2×2 grids, with all depth values set to the maximum detected depth.



(a) Downsampled tactile images with their corresponding depth (*rescaled to full resolution for visualization*)



(b) Effect of downsampling on SLAM results

Figure S13: Benefits of high-resolution touch. **(A)** We downsample the input tactile images, and their corresponding depth, by factors of $2\times$, $4\times$, $8\times$, and binary contact from Fig. S12. **(B)** We compute and plot the pose and shape metrics across 5 visuo-tactile SLAM trials of real-world bell pepper rotation (Fig. 4). The shaded regions represent one standard deviation from the mean error. We see a gradual deterioration of shape/pose metrics with lower resolution, ending with a sharp drop for the binary contact.

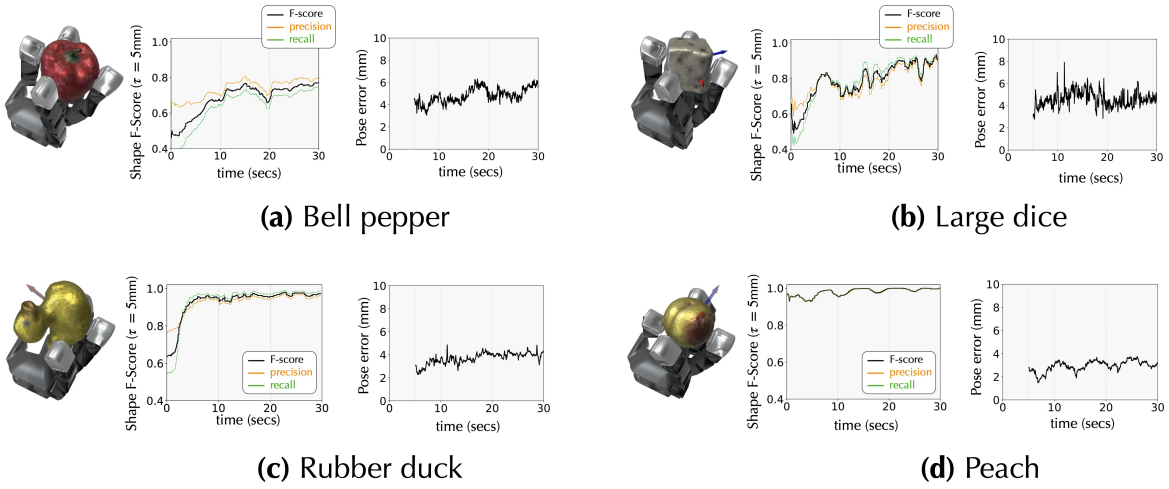


Figure S14: Shape and pose metrics over time for in-hand SLAM. Here, we plot the time-varying metrics for experiments visualized in Fig. 4. First, we note the gradual increase in F-score over time with further coverage. Additionally, we have bounded pose drift over time—for each experiment we omit the first five seconds as the metric is ill-defined then.

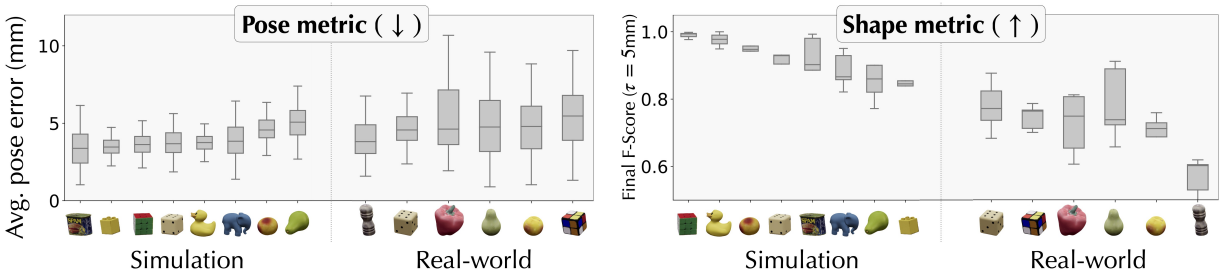
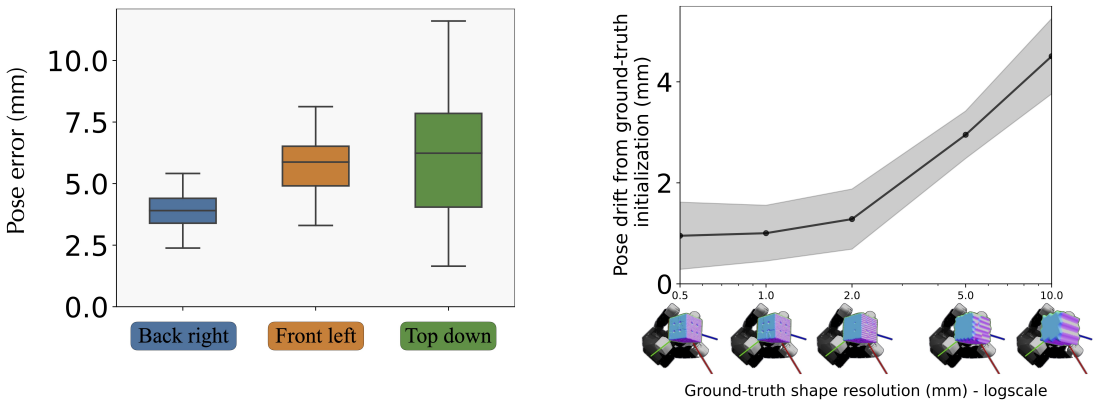


Figure S15: Pose and shape metrics from visuo-tactile SLAM of each object class in Fig. 3, sorted from best-to-worst performance. In the boxplots, the central line is the median error, extents of the box are the upper and lower quartiles, and whiskers represent $1.0\times$ the interquartile range (IQR)



(a) SLAM errors based on camera viewpoint

(b) Pose drift v.s. quality of object shape

Figure S16: (A) We plot the average pose error from 5 trials of the Rubik's cube SLAM experiment in the real-world. In each boxplot, the central line is the median, extents of the box are the upper and lower quartiles, and the whiskers represent $1.0\times$ the interquartile range (IQR). The front (27cm) camera is generally occluded by the fingers more than the back (28cm), the top-down camera is further away (49cm). **(B)** Shape quality versus pose drift: We vary the quality of our SDF by voxelizing the ground-truth mesh at different resolutions. This ranges from 0.5mm to 10mm, with deteriorating tracking results. The shaded region represents one standard deviation from the median pose drift.

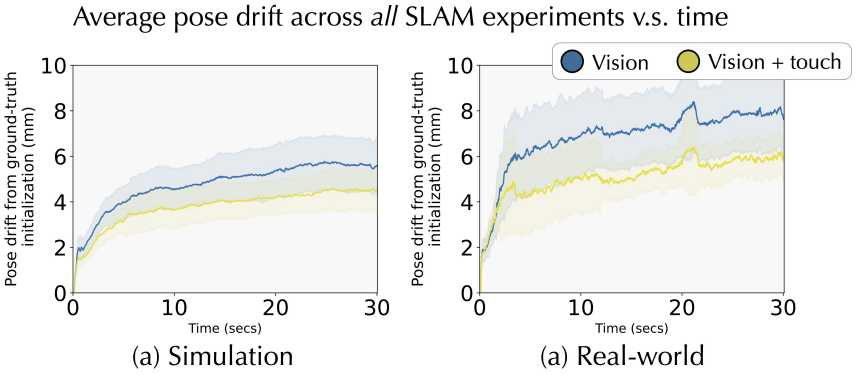


Figure S17: Average object drift drift over time. This is computed across all simulation and real-world SLAM experiments from the main text. The shaded regions represent one standard deviation from the mean pose drift.

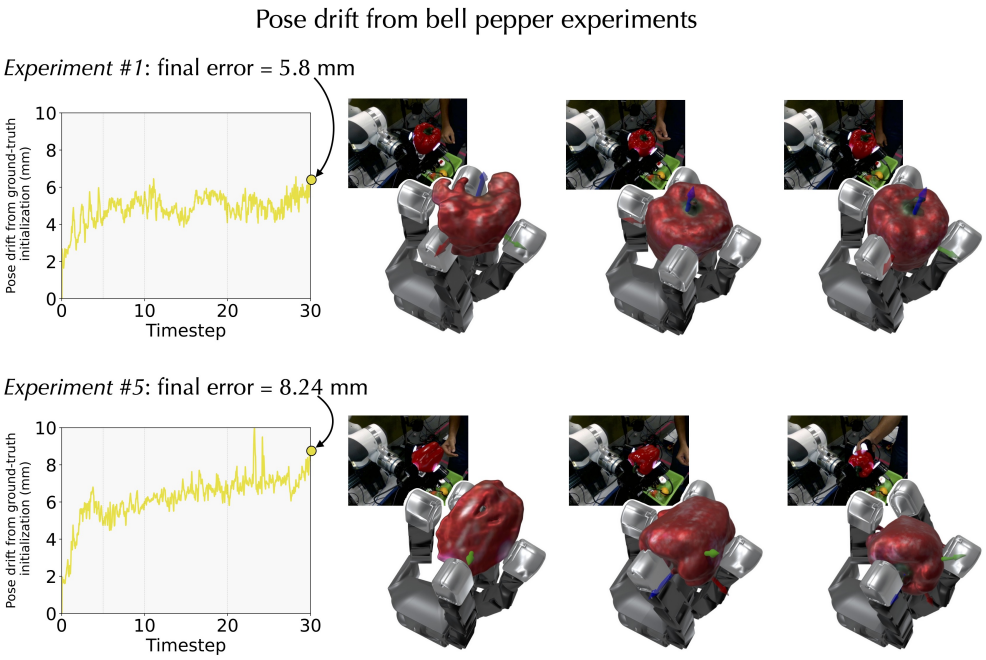


Figure S18: Visualization of pose drift. We illustrate the pose drift in two separate bell pepper experiments, with final pose metrics of 5.8mm and 8.24mm respectively. In the second experiment, the object is upside down with poor visibility, and this is reflected in the growing pose drift.

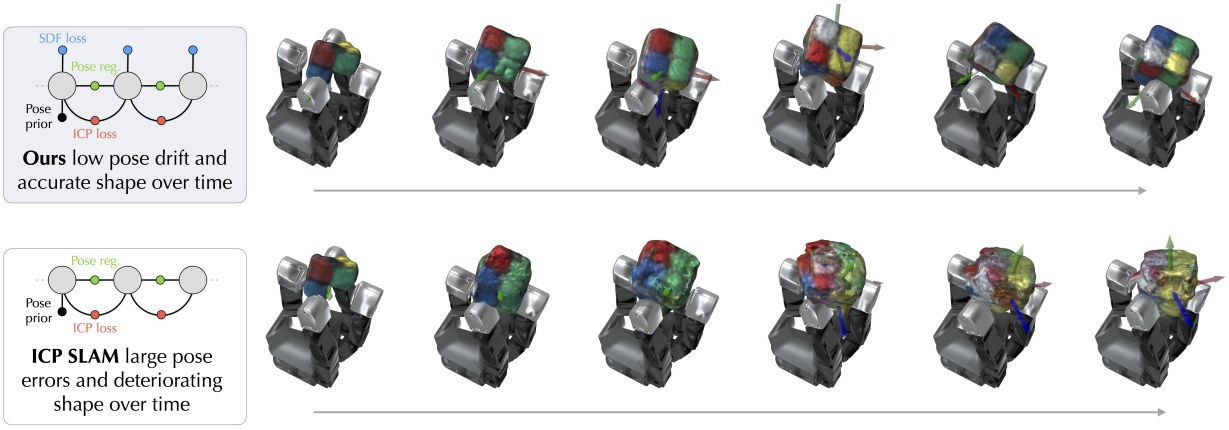


Figure S19: Importance of neural SDF loss. Example visualization of ours v.s. ICP SLAM (Table S4) for in-hand rotation of the Rubik's cube. Without the frame-to-model constraint such as the neural SDF loss, we accrue large pose errors and deteriorating shape over time.

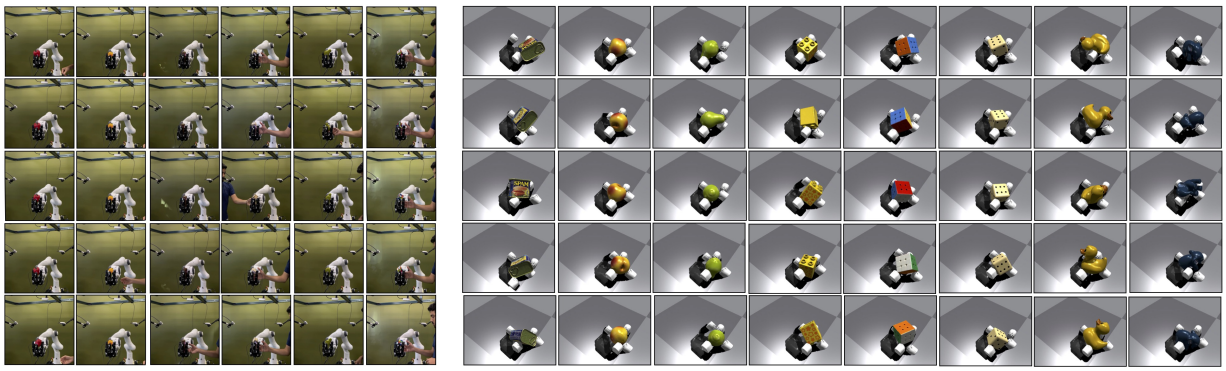


Figure S20: A collage depicting the entirety of the FeelSight dataset. We collect five sequences each in the real-world across six different objects, and five sequences each in simulation across eight different objects. In this collage each column is a unique object, and each row is a unique sequence.

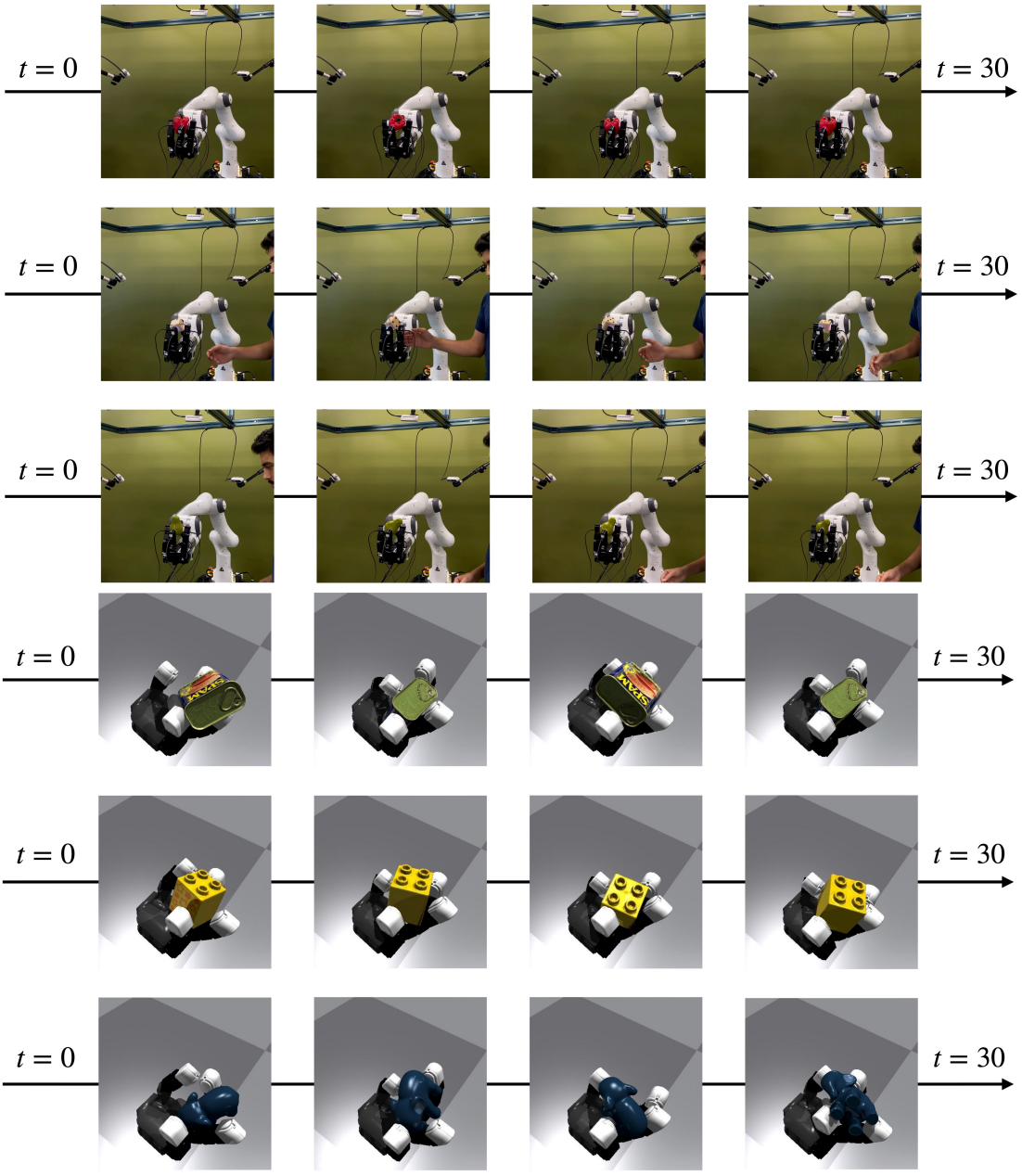
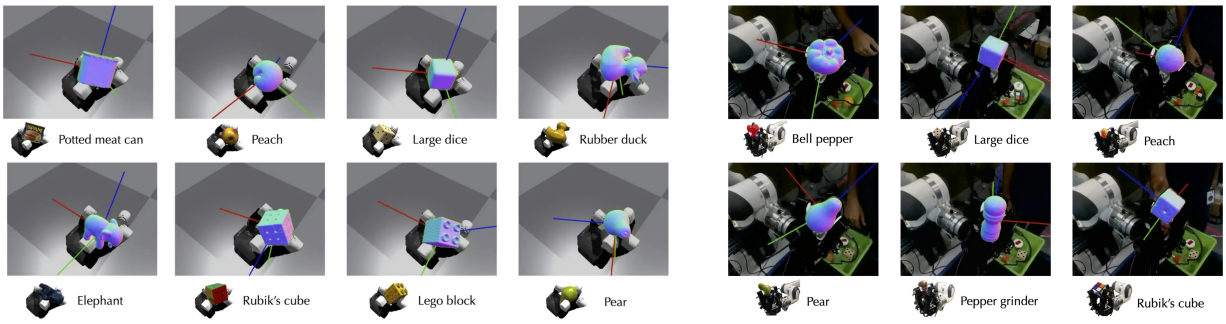
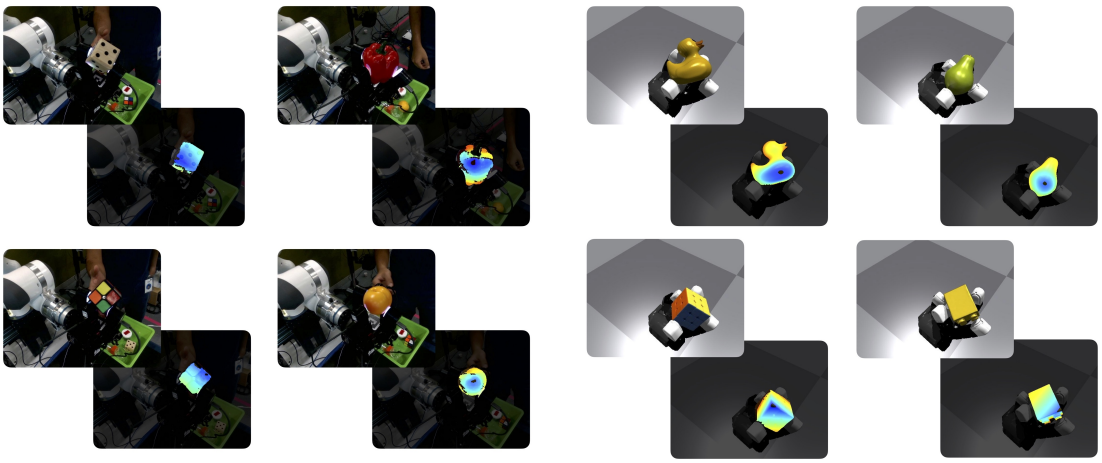


Figure S21: Sequences in the FeelSight dataset. Zoomed-in viewpoint of sequences from the collage in Fig. S20, in the real-world and simulation.



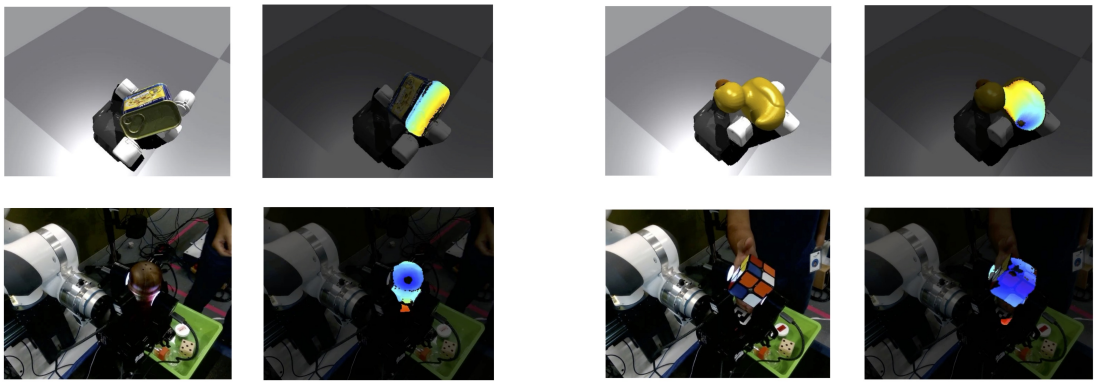
(a) Simulated pose tracking **(b)** Real-world pose tracking

Figure S22: Further visualizations of neural tracking experiments. These complement the qualitative results from the main text for both **(a)** simulated and **(b)** real-world experiments.



(a) Visual segmentation in the real-world **(b)** Visual segmentation in the simulation

Figure S23: Additional results on visual segmentation. Our segmentation module can accurately singulate the in-hand object in both **(a)** real-world and **(b)** simulated image sequences.



Examples of imprecise visual segmentation in simulation and the real-world

Figure S24: Imprecise segmentation of objects. We highlight some frames where the visual segmentation fails. We composite each Realsense RGB image with the segmented depth map.

Table S1: Statistics for our in-hand exploration policy. We show the episode length (in seconds), object rotation done in one episode (in radians), and success rate of the policy.

Object name	Episode length (s) ↑	Rotation (rad) ↑	Success rate (%) ↑
Large dice	18.13	12.79	84.32
Rubber duck	18.33	9.39	86.34
Elephant	15.93	5.56	63.14
Rubik's cube	19.09	13.12	92.78
Pear	12.45	7.35	42.42
Potted meat can	14.78	3.66	53.68
Lego block	17.32	9.89	77.14
Peach	18.73	14.37	91.16
Overall	16.85	9.52	73.87

Table S2: Binary contact ablation for pose. Avg. pose metric (↓) over 5 visuo-tactile SLAM experiments show DIGIT sensing outperforms binary contact in both sim/real ($p < 0.001$)

Sensing	**Avg. pose error ± SD (mm)**	
	Simulation (Rubik's cube)	Real-world (Bell pepper)
Binary approx.	4.46 ± 1.66	6.05 ± 0.97
Digit sensor	**3.07 ± 0.61**	**4.71 ± 0.68**

Table S3: Binary contact ablation for shape. Avg. shape metric (↑) over 5 visuo-tactile SLAM experiments show DIGIT sensing outperforms binary contact in both sim/real ($\mathbf{p} < 0.001$)

Sensing	**Avg. F-Score ± SD**	
	Simulation (Rubik's cube)	Real-world (Bell pepper)
Binary approx.	0.67 ± 0.01	0.55 ± 0.06
Digit sensor	**0.98 ± 0.00**	**0.76 ± 0.02**

Table S4: Ours v.s. iterative closest point (ICP) SLAM ($\mathbf{p} < 0.001$) for five trials of the real-world Rubik's cube experiment from the results section titled "Neural SLAM: object tracking and shape estimation" of the main text.

Modality	Pose metric (mm) (↓)	Shape metric (F-Score) (↑)
Ours	**5.862**	**0.883**
ICP SLAM	13.486	0.545