Appendices

Appendix A Tactile depth network: data and training

The TDN is trained with TACTO [32] images, minimizing heightmap reconstruction loss, as in the monocular depth estimation [62]. These image-heightmap pairs are shown in Figure 12 [right]. The TACTO interactions are over a diverse set of YCB objects shown in Figure 12 [left].

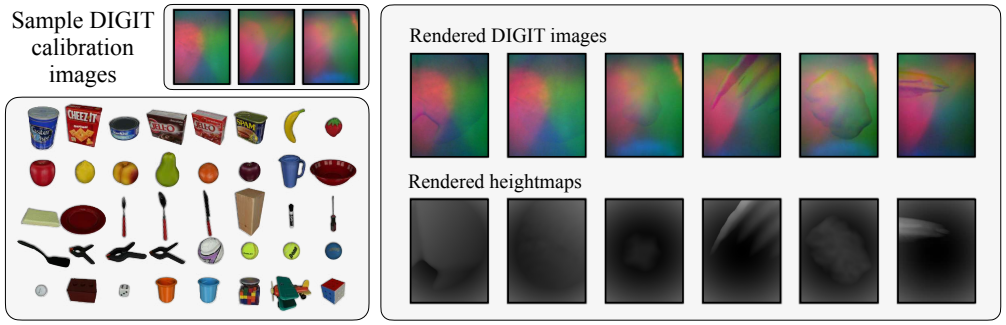

Figure 12: [top-left] Three calibration images captured from different DIGIT sensors. We augment our training data with these calibrations for better sim2real transfer. **[bottom-left]** The 40 YCB [24] objects across food, kitchen, tool, shape, and task classes. **[right]** Examples of image-heightmap pairs used for supervision.

Generating contact poses: We render realistic DIGIT images from 5000 unique poses on each object. To generate these contact poses, we sample point-normal pairs across the object's mesh. Through rejection-sampling, we can get an approximately even distribution across the surface. Additionally, we prioritize sampling edges with mesh feature angles $> 10°$. This gives us the uniform spread that we desire, while also capturing the salient features across the object classes.

After sampling these points, we add penetration depth randomly sampled between 0.5mm to 2mm. To convert a contact location to a pose, we first assign a random orientation angle ϕ around the surface normal direction between $[0°, 360°]$. We add an orientation noise angle $\theta = \mathcal{N}(0, 5°)$ in the cone perpendicular to the surface normal to ensure that the poses aren't always othogonal to the local surface. We randomly assign 2% of all poses to not make contact with the surface.

Image augmentations: For sim2real transfer, the TDN should generalize across different sensor lighting conditions. For example, in our real experiments we use three different DIGIT sensors, and there is some wear and tear of the elastomer over time. Through TACTO, we can calibrate the renderer with respect to real-world images. These images, pictured in Figure 12 [left], are captured when the DIGIT does not make contact with a surface. We collect 10 calibration images over the `YCB-Slide` dataset, and use each as the calibration for a subset of the training data generation.

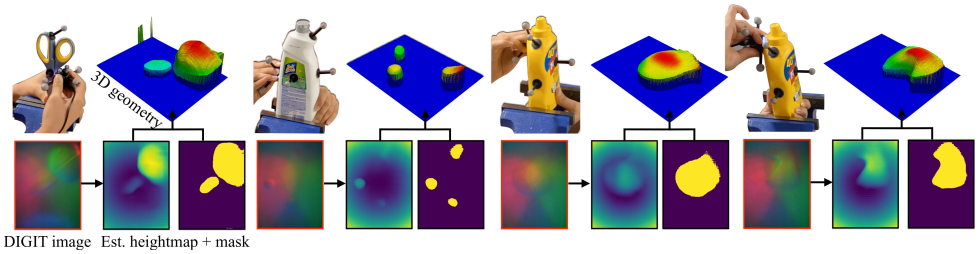

Figure 13: Extended results from the tactile depth network, similar to Figure 2. Given input image, the network predicts heightmap which is reprojected to give 3D geometry.

Appendix B Tactile codes: 3D versus image

In this section, we compare our 3D tactile codes against a baseline tactile image embedding. While Bauza et al. [18] use a similar contrastive strategy to learn image embeddings, they are object-specific. We choose instead to compare against the learned model in ObjectFolder 2.0 [53]. They extract features from the fully-convolutional residual network bottleneck layer, the tactile depth

network we use as our observation model (Section 4.1). They use these embeddings for multi-touch contact location estimation with GelSight images.

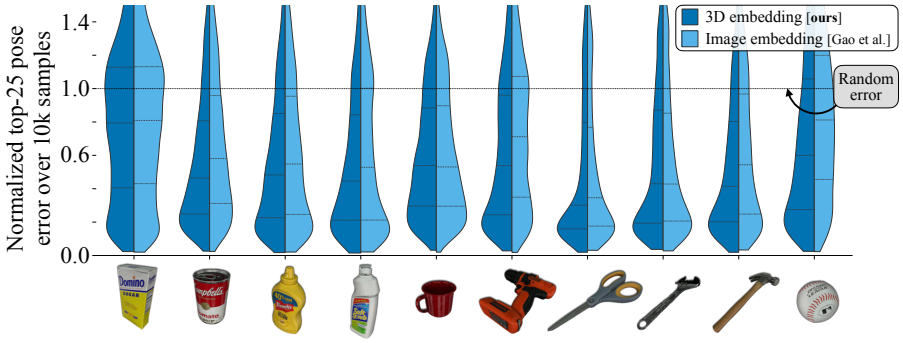

Figure 14: Pose-error for 10k single-touch queries on YCB objects, comparing our 3D tactile codes v.s. image embeddings. For each query, we get the top-25 highest scores from the tactile codebook $\mathcal{C}_o$, and compute their minimum pose-error with respect to ground-truth. This density distribution is plotted as a violin-plot, normalized by the error from a randomly-sampled touch.

We perform the same single-touch localization experiments from Figure 4, using both our tactile codes, and the image embeddings. For each method, we build an object-specific codebook. For each query tactile image, we generate its corresponding embedding and match against the codebook. We then compute the minimum pose-error from the top-25 matches, and repeat this for 10k touch queries. From Figure 14 we observe lower pose-errors for our tactile codes, with an average normalized pose-error of $\mathbf{0.473}$ compared to 0.546 from Gao et al. [53]. Also importantly, our embeddings are low-dimensional, leading to a codebook size 300 times smaller than that of the baseline. Examples of top single-touch query results for `sugar_box` are shown in Figure 15.

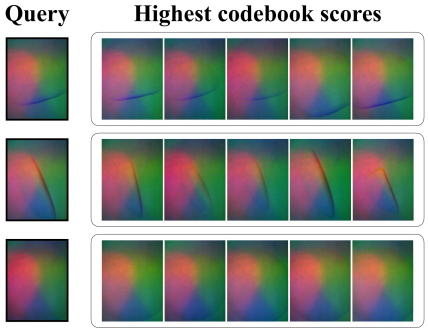

Figure 15: Examples of query tactile images matched against the tactile codebook of the `sugar_box` object. We see that the top scores in each case are images whose sensor pose is also similar to the query.

Appendix C Particle filter: Implementation details

Particle count: We initialize liberally, with $N_0 = 50$k so as to better capture poses close to the ground-truth. With too few particles, we run the risk of not capturing good 6D pose candidates initially. In practice, reducing N_t greatly improves computation time, but is a trade-off on tracking accuracy. Alongside resampling, we dynamically adjust the number of particles N_t based on filter convergence. We track the average standard deviation of the hypothesis set σ_{h_t}, and inject or remove particles according to the ratio $\sigma_{h_t}/\sigma_{h_{t-1}}$. When injecting particles, we replicate those with the largest weights, and when removing particles, we delete those with the lowest. To prevent particle depletion, we do not let it drop below 1k particles.

Pruning: We leverage our on-surface assumption, to prune particles that drift too far away from the objects surface. Given the object meshes, we construct a k-d tree of all vertices, and at each iteration we perform a nearest neighbor search for the particles using this tree. When the distance check exceeds 2mm, which we consider the sensor's maximum penetration distance, we set the corresponding particle's weight to zero. This "soft" on-surface constraint allows for some noise in the odometry, while gradually pruning candidate hypotheses.

Real-world experiment: With real DIGIT images, we've found exponential time smoothing over predicted heightmaps gives stable local geometry and removes outlier effects. Further, we reduce the frequency of particle resampling to every five iterations. This prevents particle depletion in the presence of erroneous heightmaps. In the real-world experiments, there are instances where the DIGIT can slide off the object's surface. In those cases, the TDN does not produce a point-cloud and we instead generate a randomized $\mathbb{R}^{256}$ embedding.

Appendix D Additional `YCB-Slide` details

Figure 16: [left] Our data-collection setup with the YCB object, motion capture, DIGIT sensor, and recording camera. **[right]** The 10 YCB test objects during interactions, with representative tactile images.

Our real-world dataset was collected in the indoor environment pictured in Figure 16. Each object is tightly clamped onto a heavy-duty bench vise, while the DIGIT sensor is slid across its surface. The full set of 50 simulated interactions are shown in Figure 17. Each trajectory has a fixed geodesic length of approximately 0.5m.

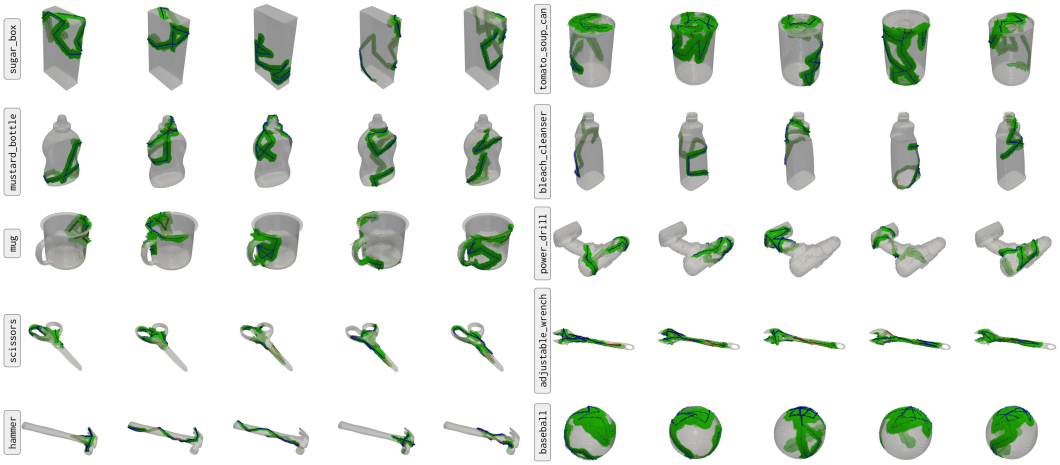

Figure 17: All 50 sliding trajectories from tactile simulation on the 10 YCB objects. Overlaid in green are the local 3D geometries captured by the tactile sensor, and the contact poses as RGB coordinate axes.

The 50 collected real-world interactions are shown in Figure 18. While these human-designed sequences are not random, each covers different sections of the object geometry.

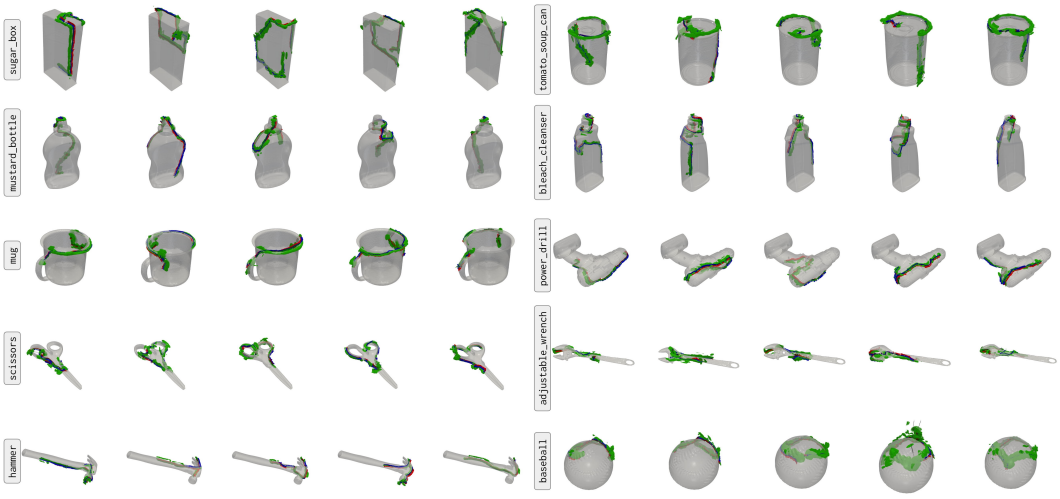

Figure 18: All 50 sliding trajectories from real-world interactions on the 10 YCB objects. Overlaid in green are the local 3D geometries captured by the tactile sensor, and the contact poses as RGB coordinate axes.

Appendix E Additional MidasTouch results

Closest hypothesis error: In Section 6, we present final RMSE for the pose particles with respect to ground truth. In Figure 8 and 9 we plot these accumulative statistics at the final timestep T:

$$e_{\text{trans}} = \sqrt{\frac{1}{N_T} \sum_{\mathbf{x} \in \mathbf{X}_T} \|\mathbf{x}_{\text{trans}} - \mathbf{x}^{\text{gt}}{}_{\text{trans}}\|_2^2} \;, \quad e_{\text{rot}} = \sqrt{\frac{1}{N_T} \sum_{\mathbf{x} \in \mathbf{X}_T} \|\mathbf{x}_{\text{rot}} - \mathbf{x}^{\text{gt}}{}_{\text{rot}}\|_2^2} \tag{3}$$

This is a general error metric for the particle filter, but penalizes a multi-modal pose distribution. We present an additional metric that computes RMSE with respect to hypothesis set h_T, and uses the error associated with the closest hypothesis to ground-truth:

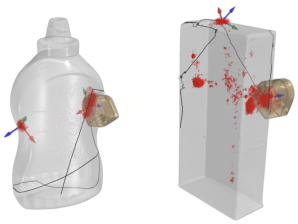

Figure 19: Examples of cases where particle pose error is high, but we still capture the true pose in our multi-modal distribution.

$$\texttt{min\_cluster}(e_{\text{trans}}) = \sqrt{\min_{\mathbf{x} \in h_T} \|\mathbf{x}_{\text{trans}} - \mathbf{x}^{\text{gt}}{}_{\text{trans}}\|_2^2} \;, \quad \texttt{min\_cluster}(e_{\text{rot}}) = \sqrt{\min_{\mathbf{x} \in h_T} \|\mathbf{x}_{\text{rot}} - \mathbf{x}^{\text{gt}}{}_{\text{rot}}\|_2^2} \tag{4}$$

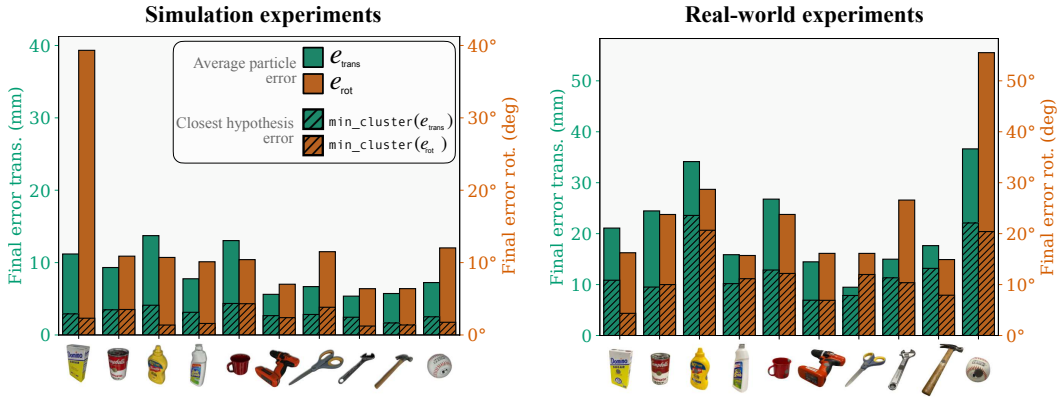

Figure 20: Plots for the 500 simulated **[left]** and real-world **[right]** trials comparing e_{trans}, e_{rot} against $\texttt{min\_cluster}(e_{\text{trans}})$, $\texttt{min\_cluster}(e_{\text{rot}})$. These serve as a complement to Figure 8 and 9, and highlight the multi-modality of the filtering problem. Given knowledge of ground-truth, we pick the cluster closest to it and plot the RMSE statistics with respect to it (Equation 4). We observe lower errors across all objects in both simulated and real settings, empirically indicating the true mode is captured in most cases.

While this assumes we have access to the ground-truth, it can better demonstrate if we capture the true pose in our multi-modal distribution. In Figure 20, we see the `min_cluster` statistics plotted alongside the original final RMSE. Across all experiments, we end up with lower error: with a median of 0.28cm + 2.03° in simulation and 1.11cm + 10.76° in the real-world.

Further qualitative results: Finally, we highlight some visualization similar to Figure 7 and 10, for the remaining YCB test objects. In Figures 21 and 22 we show snapshots of MidasTouch on the remaining YCB objects. Alongside the snapshots is the translation and rotation RMSE over time, averaged over 10 trials. We see filter convergence across different YCB objects, along with the failure mode of the baseball in Figure 22. Please refer to the supplementary video for further visualizations.

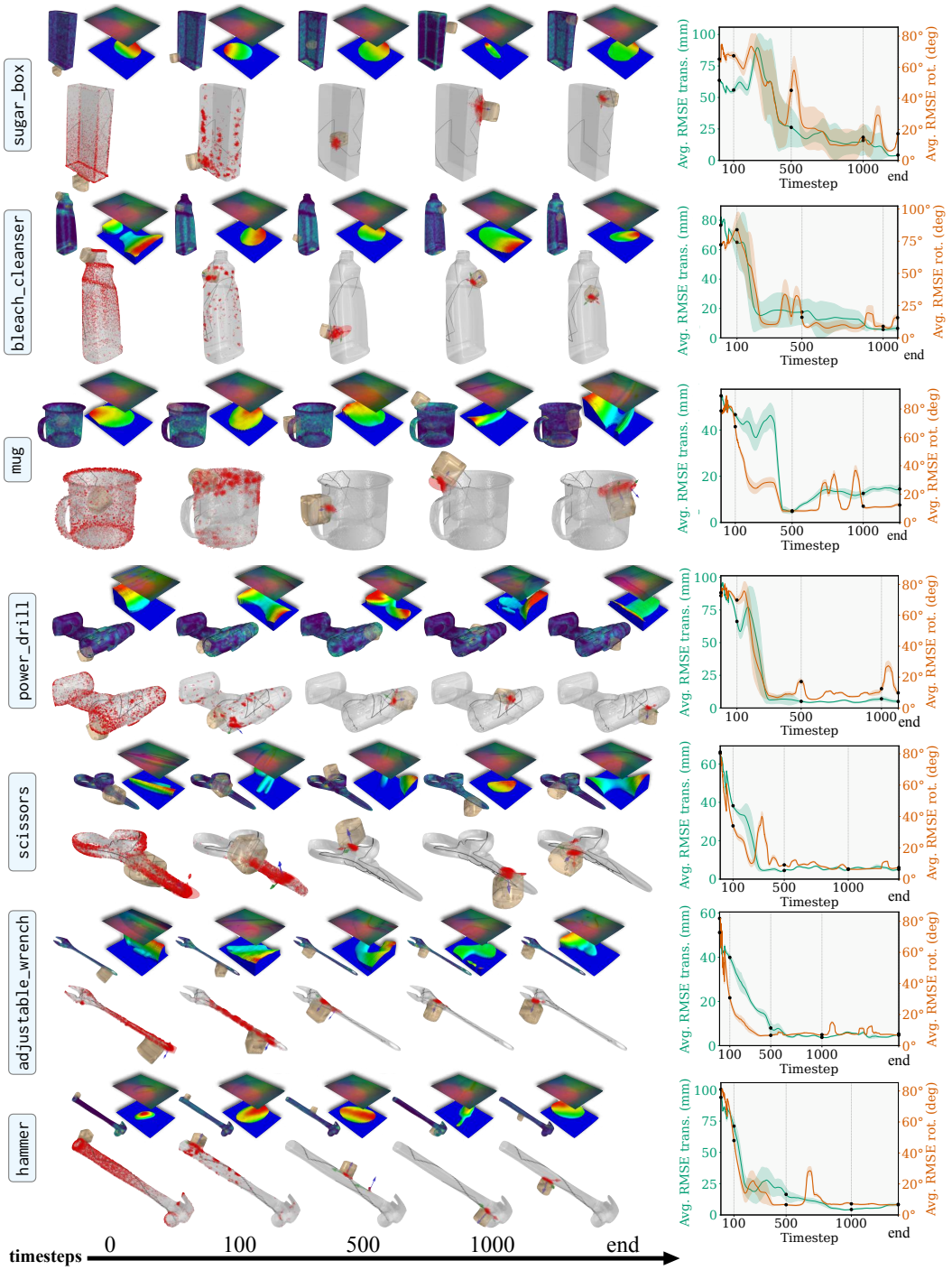

Figure 21: Select simulation results from seven objects not shown in Figure 7. For each row: **[top]** the tactile images, local geometries, and heatmap of pose likelihood with respect to the tactile codebook, **[bottom]** pose distribution evolving over time, **[right]** average translation/rotation RMSE of the distribution over time with variance over 10 trials.

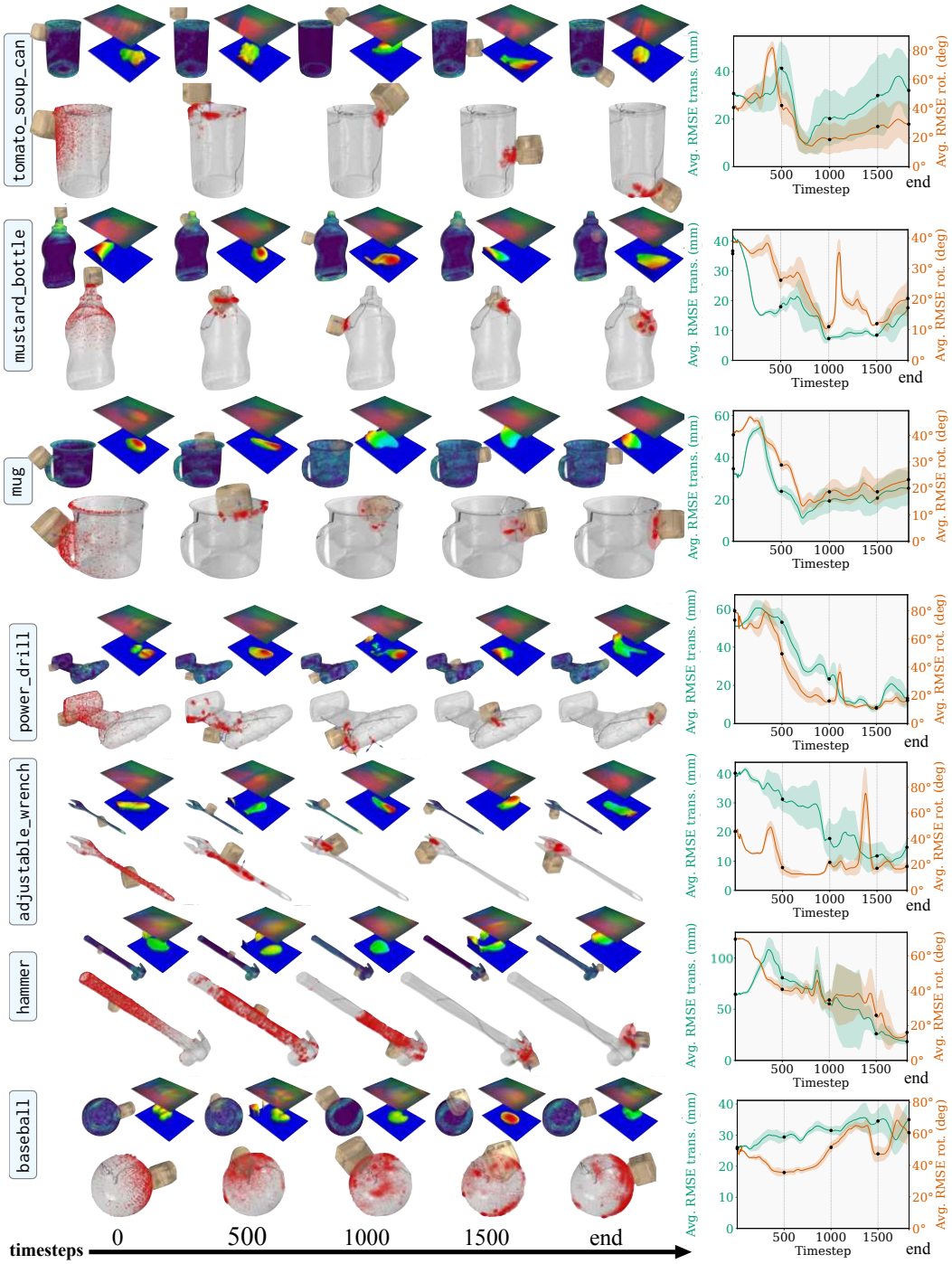

Figure 22: Select real-world results from seven objects not shown in Figure 10. For each row: **[top]** the tactile images, local geometries, and heatmap of pose likelihood with respect to the tactile codebook, **[bottom]** pose distribution evolving over time, **[right]** average translation/rotation RMSE of the distribution over time with variance over 10 trials.

Appendix F Study on contact patch area

We analyze the correlation of surface contact patch area with the performance of our filter. During interaction, it is crucial to maintain forceful contact with the surface area impinging the sensor. This gives us a larger contact area, and more 3-D surface geometry to match against the tactile codebook. For the DIGIT, this is theoretically between 0 to 6 cm^2, and can be obtained as the pixel area of $\mathbf{C}_t$ (Section 4.1).

To show the importance of larger contact areas, we record a single simulated trajectory on `power_drill`, ablated over five different penetration depth ranges. We randomly sample penetration depth in the range of $\omega \times \mathcal{N}(0.5, 2\text{ mm})$, where $\omega \in [0.1, 0.325, 0.55, 0.775, 1.0]$. Figure 24 shows the same interaction with five different ω values. We observe that large penetration captures more surface geometry.

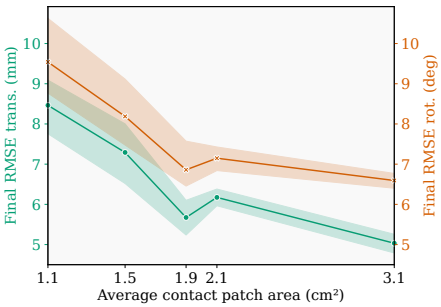

Figure 23: Plot over 50 trials of the pose error v.s. average contact area of the trajectory; larger contact areas lead to lower downstream error in tracking.

For each profile, we average the results of 10 filtering trials, and plot the final pose error v.s. average trajectory contact area. Figure 23 shows that more 3-D surface geometry can lead to lower downstream error in finger pose tracking. Intuitively, this is analogous to a depth-camera with a larger depth range, generating more complete scans of the scenes it perceives.

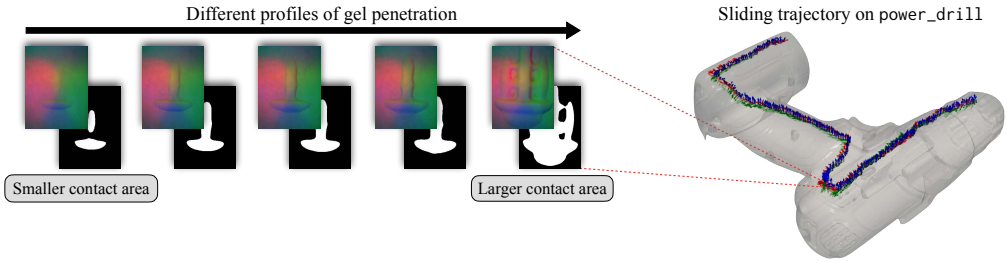

Figure 24: [right] Fixed trajectory on `power_drill` for which we apply different penetration profiles. **[left]** For the same local surface, different penetration profiles observe very different contact shapes and tactile images.

Appendix G Experiments on small parts

MidasTouch focuses on YCB-sized objects that we encounter in household and assistive robotics contexts. These are considerably larger than the robot finger, which is relevant for our desired applications of in-hand and tabletop manipulation. `YCB-Slide` spans objects with surface areas ranging from 109cm^2 (`adjustable_wrench`) to 643cm^2 (`bleach_cleanser`), while the sensor has a footprint of 6cm^2. In this section, we show simulated experiments for small parts with large sensor-model overlap, similar to prior work [16, 18].

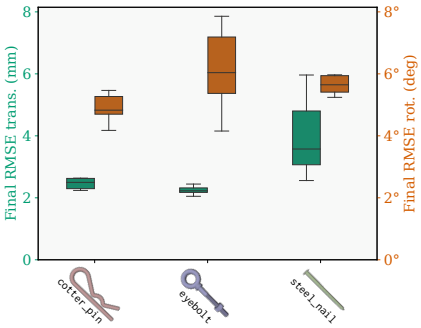

Figure 25: Boxplot of final error over 30 simulated trials on the McMaster-Carr small parts.

We select three objects from McMaster-Carr [74], the `cotter_pin`, `eyebolt`, and `steel_nail`, each of 2" length. For each we generate a tactile codebook, and record a short simulated trajectory along the object's length (just as in Section 5). In Figure 26 we show results for all three, where the filter quickly converges to the true mode. We run each experiment 10 times, and show the accumulative statistics in Figure 25. We observe a final error of $\approx [4\text{ mm}, 5°]$, which is roughly twice as accurate as results in Section 6. Moreover, this requires trajectories $10\times$ smaller, with $5\times$ less particles. This is due to the small size of objects, and larger relative field-of-view.

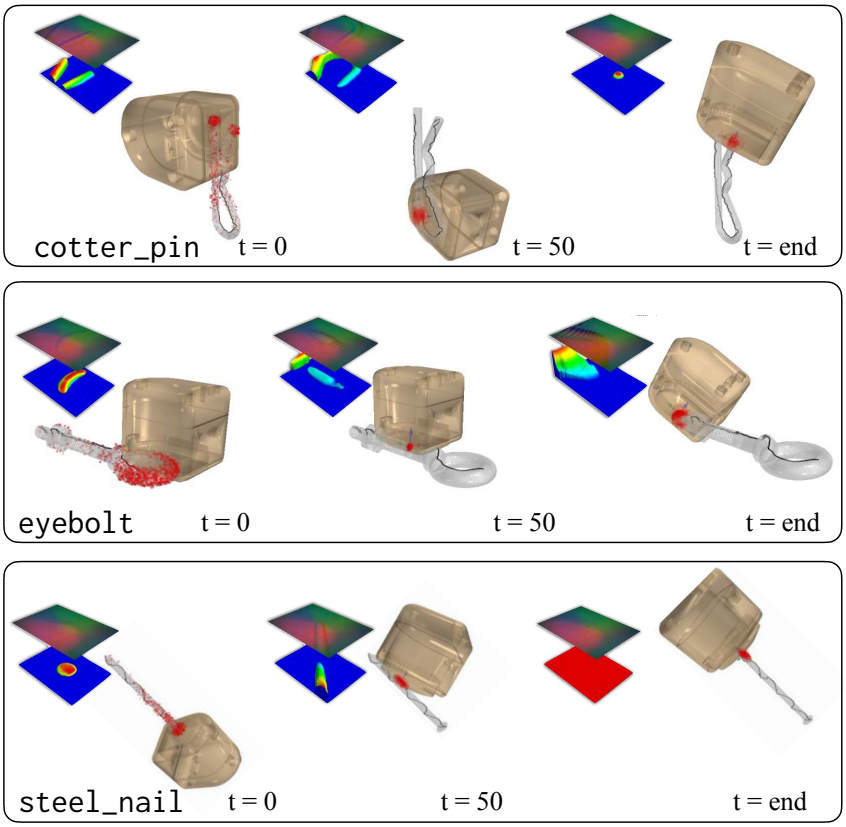

Figure 26: Filtering results from the three McMaster-Carr small parts. We visualize the tactile images, local geometries, and pose distribution evolving over time.