Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

MultiSCOPE: Disambiguating In-Hand Object Poses with Proprioception and Tactile Feedback

Andrea Sipos
Robotics Department
University of Michigan
Ann Arbor, USA
asipos *at* umich *dot* edu

Nima Fazeli
Robotics Department
University of Michigan
Ann Arbor, USA
nfz *at* umich *dot* edu

***Abstract*—In this paper, we propose a method for estimating in-hand object poses using proprioception and tactile feedback from a bimanual robotic system. Our method addresses the problem of reducing pose uncertainty through a sequence of frictional contact interactions between the grasped objects. As part of our method, we propose 1) a tool segmentation routine that facilitates contact location and object pose estimation, 2) a loss that allows reasoning over solution consistency between interactions, and 3) a loss to promote converging to object poses and contact locations that explain the external force-torque experienced by each arm. We demonstrate the efficacy of our method in a task-based demonstration both in simulation and on a real-world bimanual platform and show significant improvement in object pose estimation over single interactions. Visit www.mmintlab.com/multiscope/ for code and videos.**

I. Introduction

Dexterous object manipulation is a challenging open problem in robotics. Inaccurate in-hand pose estimation of grasped objects is an important cause of failure during robust robot manipulation. Inaccurate pose estimates can lead to misalignment (that can cause slip), unforeseen contacts, and jamming. These failure modes result in unreliable interactions between the robot and its environment when performing tasks such as tool use and assembly.

Many current state-of-the-art approaches to this problem use visual feedback to identify objects in the scene and estimate their poses [6, 9, 11, 12, 15, 19, 21, 25, 26, 29]. These approaches produce poor or unreliable estimates in vision-deprived environments. Vision deprivation occurs naturally due to occlusions from the robot or environment during task execution (e.g., during fine assembly or operation in narrow spaces) and is often unavoidable [27, 30, 32, 33]. Recent progress in high-resolution collocated tactile sensing has enabled in-hand pose estimation [1, 3, 16, 18, 28, 31, 34]. While these novel sensors present many new possibilities for tactile feedback, they also present several challenges including high cost, components that easily fatigue, difficult to model dynamics between the robot and grasped object, and data processing complexity.

Here, we present MultiSCOPE: a method to simultaneously estimate the poses of two objects grasped in unknown configurations by two collaborative arms using only proprioception and 6-DOF force-torque sensing at the wrists, shown in Fig. 1.

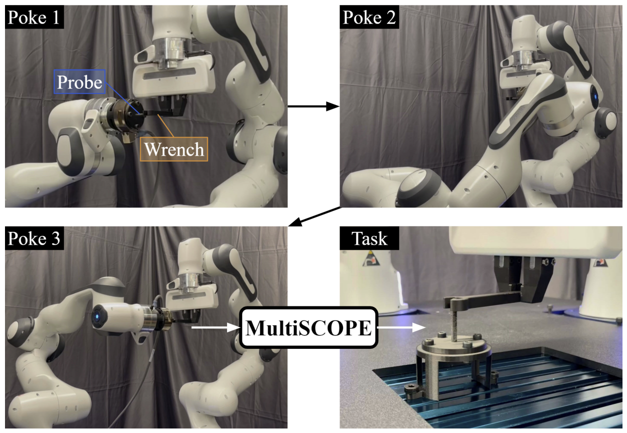


Fig. 1. Each robot in this bimanual system is grasping an object (probe and wrench) in unknown configurations. Using our method, MultiSCOPE, the robot is able to localize both simultaneously using a sequence of information-gathering ”poking actions”, its own proprioception and external wrench measurements, and object models. After running MultiSCOPE to estimate object poses, the robot will attempt to use the wrench tool.

Our method works by iteratively bringing the two objects into contact with one another and uses two complementary particle filters to accurately estimate in-hand object poses. These filters exploit mutual information between known object geometries and detected contact events. Specifically, we contribute:

- A framework for sequential bimanual interactions to reduce pose ambiguity through iterative Bayesian filtering,
- A memory loss to promote consistency between sequential information-gathering physical interactions,
- A tool segmentation algorithm to improve object pose and contact location estimation, and
- A wrench loss to promote converging to object poses and contact locations that explain the external force-torque experienced by each arm.

We demonstrate the efficacy of our proposed approach on simulated and real data using 2 Franka Emika Panda robots. We emphasize that our approach does not require visual feedback or high-resolution tactile sensing and is compatible with most existing robot hardware.

II. Related Work

In-hand object pose estimation is an important challenge in robotics. The extensive literature on the subject can be grouped into two approaches: vision-based and tactile. More recently, there is an increasing amount of overlap in visuotactile approaches that we discuss below:

A. *Vision-Based State Estimators*

Many works on in-hand object pose estimation use vision-based approaches [9, 11, 15]. Vision-based methods are significantly impacted by occlusion. Aside from occlusion, two main challenges for these methods are i) multi-object interactions (i.e. cluttered scenes) [21, 29] and ii) reasoning about contact [6, 25, 26]. One of the main advantages of these methods is that they often mitigate the need for object models via learning because vision is a much more global sense than touch [19, 25, 26].

B. *Tactile and Visuotactile-Based State Estimators*

These works use a variety of tactile sensors and visuotactile methods to reason about pose at the interface between the robot and the object. There are many different types of tactile and visuotactile sensors including robotic skin [8], tactile fingertips [3, 14, 31], Soft Bubbles [33], and GelSlim [16, 28] that are being used in robotic manipulation research. While these sensors hold a lot of promise, they are often expensive and fragile. In our method, we use robot-estimated wrenches and an off-the-shelf force-torque sensor to gain tactile feedback. These methods are more accessible than most current tactile and visuotactile sensors. Two other works that also use robot proprioception to estimate contact are [17, 34]. However, these methods estimate contact on the surface of the robot rather than an object grasped by the robot. [23] extends this idea to an object grasped by a robot, as well as to objects grasped in a bimanual system. Several other methods use finger/robot position combined with RGB(-D) data to estimate object shape and pose [1, 12, 22]. Additionally, several works have explored using particle filtering methods [2, 4, 10, 13, 24] to estimate contact between a robot and its environment. Generally, these methods consider single robot arms or known in-hand object pose but an uncertain environment. Our method could reasonably be integrated with the above vision-based and tactile techniques discussed here.

III. Methodology

In this paper, we extend recent work from Sipos and Fazeli [23] on single-action object pose estimation with proprioception and tactile feedback called Simultaneous Contact and Object Pose Estimation, or SCOPE. Our method extends this framework to multiple contact-rich interactions in order to achieve object pose estimates that are consistent across actions and result in more accurate object pose estimates. We emphasize that in this framework, the robot is estimating the pose of both grasped objects (e.g., the probe and tool in Fig. 1).

A. *Assumptions*

We assume that the grasped objects are rigid and that we have access to their geometric models. We further assume that the robot is capable of estimating externally applied wrenches. This is a common functionality for most existing cobots (e.g., Franka Emika Panda and Kuka lbr iiwa) and can be implemented using an external force-torque sensor at the wrist for arms without this capability. We also assume that we have access to robot proprioception. We assume that contacts are well-approximated by the point contact model and that no moments are transmitted through the contact location. Finally, we assume that the robot's exploratory actions (pokes) result in contact with no relative slip. These assumptions are the same as SCOPE [23] and no additional information is required.

B. *SCOPE Review*

SCOPE [23] leverages two complementary particle filters to jointly estimate the contact location and grasped object poses (for two objects grasped in unknown configurations) using robot proprioception and tactile feedback from a single interaction. The first filter is called the Contact Particle Filter for Grasped Objects, or CPFGrasp. For each arm, this filter takes as input the measured robot wrench at the end-effector $\boldsymbol{\Gamma}_{EE} \in \mathbb{R}^6$ and an estimate of the pose of the grasped object represented as a transform between the object and end-effector frames $\mathrm{H}_O \in \mathbb{SE}(3)$. In this work, we estimate grasped object poses in $\mathbb{SE}(2)$ because the pose is constrained by the parallel jaw grippers of the Frankas. The output of this filter is a belief distribution over the estimated contact locations on the objects' surfaces and the forces applied at these locations conditioned on $(\boldsymbol{\Gamma}_{EE}, \mathrm{H}_O)$. This distribution is approximated by a set of particles referred to as the Contact Location Particles (CLPs): $\mathrm{R}_c = \{\boldsymbol{r}_i \in \mathbb{R}^3 \mid i \in \mathbb{N} < n_{clp}\}$ where $\boldsymbol{r}$ denotes the contact location w.r.t. to the end-effector frame and where n_{clp} denotes the number of particles. Once converged, CPFGrasp is able to provide a score $\mathrm{S} = \{s_i \in \mathbb{R} \mid i \in \mathbb{N} < n_{clp}\}$ for how well an estimated pose is able to explain the measured robot wrenches:

$$(\mathrm{R}_c, \mathrm{S}) = \text{CPFGrasp}(\mathrm{H}_O, \Gamma_{EE})$$

The second filter, SCOPE, maintains a belief distribution over object poses and updates this belief distribution using the scores computed by CPFGrasp. This distribution is approximated using the Object Pose Particles (OPPs): $\mathrm{H} = \{\mathrm{H}_O^i \in \mathbb{SE}(2) \mid i \in \mathbb{N} < n_{pp}\}$ where n_{pp} denotes the number of particles. Given an initial distribution of object pose particles, CPFGrasp provides scores for how well each particle explains the measured wrench, and SCOPE updates these particles based on their computed scores. These two filters are run sequentially until convergence. Each object in the bimanual system has its own OPP distribution. OPPs from each grasped object in the bimanual interaction are scored in pairs using the losses described below.

Losses The losses introduced in SCOPE are the penetration loss, force alignment loss, and contact consistency loss. Penetration loss penalizes OPP pairs that are in collision. Contact consistency loss penalizes OPP pairs that have spatially distant contact locations. Finally, force alignment loss penalizes OPP pairs that have misaligned applied force. These losses are formalized in Eqs. 1, 2, and 3:

$$\mathcal{L}_P = \max\{0, N_{pp} - \epsilon_{pp}\} \tag{1}$$

$$\mathcal{L}_C = \sum s_t s_p \|\boldsymbol{r}_t - \boldsymbol{r}_p\|_2 \tag{2}$$

$$\mathcal{L}_F = \sum s_t s_p \| - \boldsymbol{w}_t - \boldsymbol{w}_p\|_2 \tag{3}$$

where we introduce the subscripts t and p for clarity of notation to describe the objects (e.g., tool and probe). Here, N_{PP} is the number of points in penetration for a given OPP pair and ϵ_{PP} is a threshold for penetration in the ground truth. $\boldsymbol{r}_t$ and $\boldsymbol{r}_p$ are the CLPs for the tool and probe respectively. n_{clp} is the number of CLPs for each of the objects. s_t and s_p are the scores of each CLP, and $\boldsymbol{w}_t$ and $\boldsymbol{w}_p$ are the external force estimates at each contact location in the world frame. The summations are taken pairwise.

C. *MultiSCOPE*

To extend SCOPE for sequential contacts, we introduce a tool segmentation algorithm that improves contact location initialization and convergence properties, a memory loss to allow our method to find temporally consistent solutions, and a wrench loss to promote consistency in wrench measurements. In the following, we provide the details of each contribution and summarize our proposed method in Algs. 2 and 3. Alg. 2 contains extensions to the algorithm presented in [23].

Algorithm 1 Tool Segmentation

```
procedure SEGMENTTOOL($\mathbf{n}_\mathcal{S}, n_{clusters}, \varepsilon, n_{min}$)
    $\mathbf{n}_{unique} \leftarrow$ getUnique($\mathbf{n}_\mathcal{S}$)
    centroids $\leftarrow$ KMeans($\mathbf{n}_{unique}, n_{clusters}$)
    groups$_\mathbf{n} \leftarrow$ groupNorms($\mathbf{n}_\mathcal{S}$, centroids)
    faces $\leftarrow$ DBSCAN(groups$_\mathbf{n}, \varepsilon, n_{min}$)
    return faces
end procedure
```

1) Tool Segmentation: In order to ensure better convergence properties from the CPFGrasp introduced in [23], we propose a tool segmentation method outlined in Alg. 1 that is designed to improve CLP initialization and consequently pose estimation. The original CPFGrasp method uniformly sampled from the tool surface, and this means that smaller faces are less likely to be sampled than large ones. However, in the case of the grey face in Fig. 2a, these faces may be intended to make contact with the environment. If there are no CLPs on this face at initialization, it is unlikely that the filter will converge to it because the particles would have to traverse high-error faces to get there. To address this, we propose a tool segmentation algorithm that samples N_{face} particles on each face, ensuring better convergence properties for CPFGrasp. In order to do this, we segment the tool into faces, then sample points from each. We define a face as a group of points that have similar surface normals and are spatially close together. To find groups of similar surface normals, we use K-Means clustering [20] of

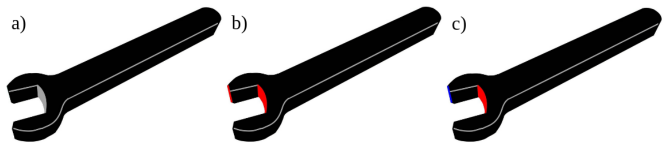


Fig. 2. a) Filled gray: useful face of the wrench tool, also one of the smaller faces b) Filled red: faces of the wrench that share the same surface normal c) Filled red and blue: faces that share the same surface normal segmented into different faces based on spatial proximity.

unique surface normals. After grouping by surface normal, we use DBSCAN [7] to further separate faces that share the same surface normal but are spatially distant from one another, an example of which is shown in Fig. 2b-c. Tool segmentations produced by this method for all tools are shown in Fig. 4. This method's parameters are easy to modify for new tools. This contribution improves the performance and reliability of MultiSCOPE over random CLP initialization, as we show in Fig. 11.

2) Wrench Loss: While the penetration, force alignment, and contact consistency losses in [23] encode many properties of contact-rich interactions, they don't explicitly enforce that the algorithm converges to the object pose(s) that best explains the wrench the system is experiencing at the end-effector Γ_{EE}. To this end, we add a wrench loss $\mathcal{L}_\Gamma$ to our method. We define ε_Γ in Eq. 4 as the wrist wrench error of the CLP distribution:

$$\varepsilon_\Gamma = \sum_{i=0}^{n_{clp}} s_i |\Gamma_i - \Gamma_{EE}| \tag{4}$$

$\mathcal{L}_\Gamma$ in Eq. 5 is the summed difference between ε_Γ and ε_{min} for the tool and probe (denoted $_t$ and $_p$ respectively), where ε_{min} is the lowest error found so far across SCOPE steps for this action:

$$\mathcal{L}_\Gamma = \varepsilon_t - \varepsilon_{min,t} + \varepsilon_p - \varepsilon_{min,p} \tag{5}$$

The inclusion of this loss metric ensures that in addition to the contact, alignment, and penetration constraints, the pose estimates for the objects also explain the experienced wrist wrench. This idea is shown in Fig. 3. On the left, the CPFGrasp solution for the given object pose does not resolve the wrist wrench experienced by the robot because the object surface does not contain the ground truth contact location. The object pose on the right is a better estimate because the CPFGrasp solution resolves the wrench that the robot is experiencing at the wrist.

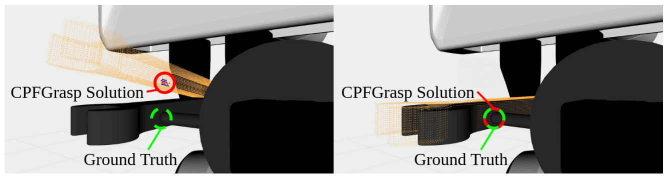


Fig. 3. Left: The CPFGrasp solution does not resolve the wrist wrench experienced by the robot. Right: The CPFGrasp solution is able to resolve the wrist wrench. $\mathcal{L}_\Gamma$ allows us to distinguish between these cases.

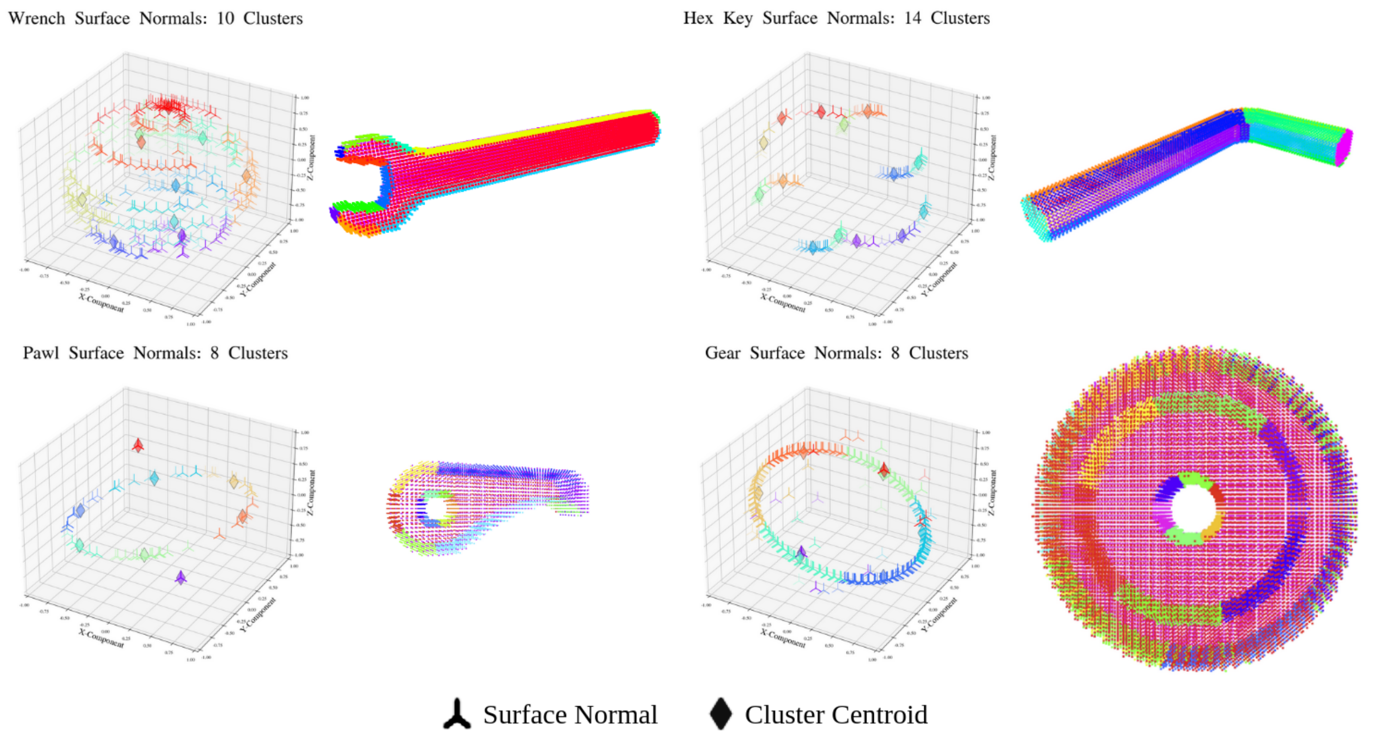


Fig. 4. The result of K-Means clustering of unique surface normals alongside the resulting tool segmentation for each of the wrench, hex key, pawl, and gear tools. Each tri-mark is the 3D coordinate representing the X-, Y-, and Z-component of the surface normal unit vector. In the resulting tool segmentation, each face is distinctly colored.

3) Memory Loss: To build MultiSCOPE, we introduce a memory loss $\mathcal{L}_M$ to our algorithm. The goal of this loss is to leverage the results of previous interactions and reduce uncertainty in our object pose estimates. This idea is shown in a simple 2-DOF translational case in Fig. 5. The key challenge we address in our implementation is assessing consistency between pose estimates and sequential actions. In order to do this, we introduce the contact clouds r_{cc} for the tool and the probe that contain estimated contact points. The contact clouds grow over time as more actions are taken.

In order to construct the contact cloud with multiple actions, we take the process depicted in Fig. 6. We define two scoring functions in Eqs. 6 and 7: S_{OPP} and S_C.

$$S_C = \eta_P \mathcal{L}_P + \eta_C \mathcal{L}_C + \eta_F \mathcal{L}_F + \eta_\Gamma \mathcal{L}_\Gamma \tag{6}$$

$$S_{OPP} = \eta_P \mathcal{L}_P + \eta_C \mathcal{L}_C + \eta_F \mathcal{L}_F + \eta_\Gamma \mathcal{L}_\Gamma + \mathcal{L}_M \tag{7}$$

The difference between these functions is that S_C does not include memory loss and is used to directly compare OPP pairs to one another across actions. At the last step of SCOPE for Action A_n, we find the top n_{OPP} tool and probe pose pairs using S_{OPP}. Without applying the noise model to these pairs, we move to Action A_{n+1} and run the first step of SCOPE. Using S_C rather than S_{OPP}, we again find the top n_{OPP} pairs. By doing this, we are rewarding the OPP pairs that score highly in both A_n and A_{n+1} and penalizing the OPP pairs that score well in only A_n. In order to update the contact cloud, we use Eq. 8 to estimate the contact location $\boldsymbol{r}_{cc,i}$ for each OPP in the top pairs, then project $\boldsymbol{r}_{cc,i}$ back into the frame of A_n. We weight $\boldsymbol{r}_{cc,i}$ by the score of the OPP it belongs to, $S_{c,i}$. For the remaining SCOPE steps in A_{n+1}, we use S_{OPP} to take advantage of the contact cloud that we have just updated for each tool. After we update the contact cloud, we project it into the frame of A_{n+1} and use S_{OPP} finish SCOPE for A_{n+1}.

$$\boldsymbol{r}_{cc,i} = S_{C,i} \sum_{j=0}^{N_{clp}} s_j \boldsymbol{r}_j \tag{8}$$

When we use the contact cloud r_{cc} to score memory at each step, we compute the weighted average distance from each point in the contact cloud to the surface of the object using a signed distance function (SDF) as shown in Eq. 9.

$$\mathcal{L}_M = \sum_{i=0}^{N_{cc,t}} w_i SDF_t(r_{cc,t_i}) + \sum_{i=0}^{N_{cc,p}} w_i SDF_p(r_{cc,p_i}) \tag{9}$$

We additionally encourage consistency between actions by introducing what we call "memory dropout". There are some situations in which none of the OPPs provided by A_n are consistent with A_{n+1}. We check for this by comparing S_C after the last step of A_n, $S_{C_{-1}}$, with S_C after the first step of A_{n+1}, S_{C_0}. Since the OPPs for each of these steps are the same, we can directly compare their scores. We expect that if the OPPs are consistent between A_n and A_{n+1}, $S_{C_0} \approx S_{C_{-1}}$. If $S_{C_0} \gg \Delta_C S_{C_{-1}}$, we exclude A_n from memory scoring in all future scoring runs and increase the magnitude and spread of the noise model for the rest of A_{n+1}.

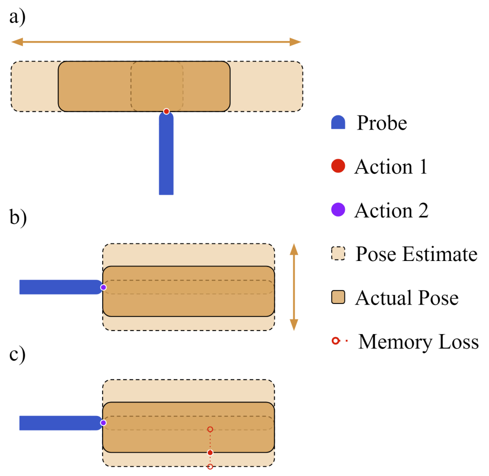


Fig. 5. Simple 2-DOF translational example of the intuition behind $\mathcal{L}_M$. a) Probe takes Action 1, resulting in object pose estimates that have horizontal uncertainty. b) Probe takes Action 2, resulting in object pose estimates that have vertical uncertainty. c) By combining Action 1 and Action 2, we can score object pose estimates on how well they satisfy both actions that have been taken and reduce our uncertainty.

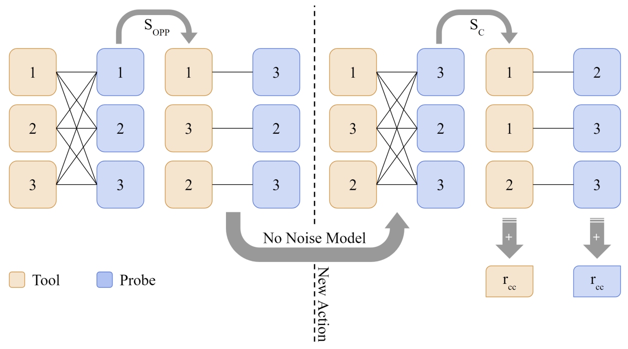


Fig. 6. OPPs are represented as blocks, colored orange for the tool and blue for the probe. At the end of one action we use S_{OPP}, which includes $\mathcal{L}_M$, to score our top OPP pairs. We then move to the next action without applying the noise model so that we can directly compare the consistency of the last and current OPP pairs using S_C, which does not include $\mathcal{L}_M$. We take the most consistent particles and use them to update each object's contact cloud r_{cc} for use in memory scoring.

4) Preventing Divergence: When we inject noise after each SCOPE step, we present the opportunity for the object pose estimate to get worse. Therefore, when it is time to score the OPP pairs we pick the top n_{OPP} object pose pairs between the last and current SCOPE steps.

IV. Experiments

A. Simulation Environment

We implement our method first in simulation. We used Pybullet [5] and configured the environment to reflect our hardware setup. We use two Franka Emika Panda robots mounted on the same table. In simulation, both robots use the Panda Hand. We use four tools (wrench, hex key, pawl, and gear) and a probe. We set the ground truth pose to be $[0,0,0]_{EE}$ for each tool and the probe. To generate each action, we sample a surface point of the tool and its surface

Algorithm 2 SCOPE: $_t \rightarrow$ tool, $_p \rightarrow$ probe

```
procedure SCOPE($\mathrm{H}, \mathbf{\Gamma}, f, \bar{\boldsymbol{S}}_{C_{-1}}, r_{cc}, n_A$)
    $\mathrm{H}_t, \mathrm{H}_p \leftarrow \mathrm{H}$
    $\mathbf{\Gamma}_{EE,t}, \mathbf{\Gamma}_{EE,p} \leftarrow \mathbf{\Gamma}$
    $f_t, f_p \leftarrow f$                                  ▷ Segmented object faces
    $N_{os} \leftarrow$ Number of SCOPE steps
    for $i \leftarrow 0;\ i < N_{os};\ i++$ do
        for $j \leftarrow 0;\ j < N_{opp};\ j++$ do
            $(\mathrm{H}_{t,j}, \mathrm{H}_{p,j}) \leftarrow \mathrm{H}_j$
            $\mathrm{R}_{c,t} \leftarrow$ CPFGrasp($f_t, \mathrm{H}_{t,j}, \mathbf{\Gamma}_{EE,t}$)
            $\mathrm{R}_{c,p} \leftarrow$ CPFGrasp($f_p, \mathrm{H}_{p,j}, \mathbf{\Gamma}_{EE,p}$)
        end for
        if $i$ is 0 then
            $\boldsymbol{S}_{OPP} \leftarrow$ scoreConsistency($\mathrm{H}_t, \mathrm{H}_p, \mathrm{R}_{c,t}, \mathrm{R}_{c,p}$)
            $r_{cc} \leftarrow$ updateContactCloud($\mathrm{H}_t, \mathrm{H}_p, \mathrm{R}_{c,t}, \mathrm{R}_{c,p}$)
            if given $\bar{\boldsymbol{S}}_{C_{-1}}$ and $\bar{\boldsymbol{S}}_{OPP} > \Delta_C \bar{\boldsymbol{S}}_{C_{-1}}$ then
                dropMemory($n_A - 1$)
            end if
        else
            $\boldsymbol{S}_{OPP} \leftarrow$ scoreOPPs($\mathrm{H}_t, \mathrm{H}_p, \mathrm{R}_{c,t}, \mathrm{R}_{c,p}, r_{cc}$)
            $\mathrm{H} \leftarrow$ preventDivergence($\mathrm{H}$)
        end if
        if $i + 1 < N_{os}$ then
            $\mathrm{H} \leftarrow$ Importance-Resample($\mathrm{H}, \boldsymbol{S}_{OPP}$)
            $\mathrm{H} \leftarrow$ Noise-Model($\mathrm{H}, i, n_A$)
        end if
    end for
    return $\mathrm{H}, \mathrm{S}$
end procedure
```

normal. We disable collision between the probe and tool then use the surface point and normal to create a target pose for the probe. To obtain force-torque data, we use Pybullet's applyExternalForce API to apply 3N of normal force to the contact point. We do this because the force-torque generated through contact resolution in Pybullet can be unpredictable and at a different scale from real-world data. We selected 3N of normal force based on similar real-world interactions, where the normal force generally falls between 2N and 5N. We take the joint states of each robot directly from Pybullet as our proprioception data.

B. Real-World Environment

We further test our method in the real world using the wrench tool. We again use two Franka Emika Panda robots mounted on the same table. However, for the real robots we use one robot with a Panda Hand and the other with an ATI Gamma FT sensor mounted at the wrist with the probe attached directly, as shown in Fig. 1. This shows flexibility in the source of force-torque data. While running MultiSCOPE, we still treat the probe pose as unknown by using the geometry of a graspable probe and simulated gripper fingers at the origin of the Gamma. This is shown in Fig. 7. The 3D-printed wrench tool has a crosshair pattern that interfaces with the Panda

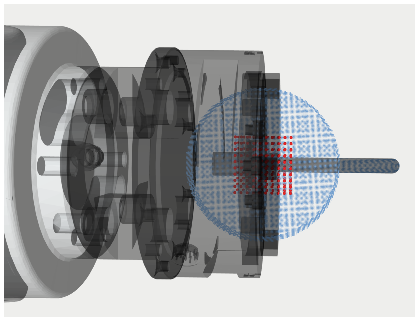


Fig. 7. Despite the probe being fixed on the real robot, we treat its pose as being unknown in MultiSCOPE. To do this, we create a graspable probe and use its mesh in MultiSCOPE. The surface points of this mesh are shown in blue. To check grasp validity we create gripper points, shown in red, at the origin of the ATI Gamma FT.

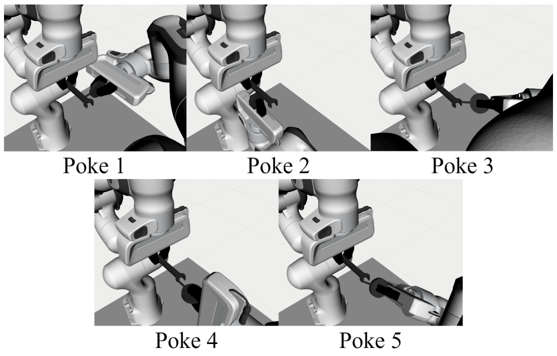


Fig. 8. 5 poking actions selected for MultiSCOPE trials in simulation.

Hand fingers to enforce that the ground truth pose is again $[0,0,0]_{EE}$.

One of the assumptions that we make in this method is that we have access to robot proprioception. We do not reason about noise or error in it. However, in the real world, we find that this assumption does not hold to such a high standard. In using our bimanual robot platform, we have found that it is difficult to precisely calibrate the robots with respect to one another even though they are rigidly mounted to the same dimensioned surface. We have developed a simple calibration routine in which we rigidly attach the end-effectors to one another then manually guide the attached arms through the workspace while collecting joint state data. Using this joint state data and geometric information, we map one end-effector frame to the other and create a point cloud for this frame on each robot model. If the robots were perfectly calibrated the point clouds would be identical; however, we have observed that with no calibration we often find 1-1.5 cm of RMS error between the two point clouds. We perform ICP to find the transformation that best maps one point cloud to the other, then apply this transformation to the base of that robot. We find that this method reduces our RMS error to 3-7 mm.

In addition to robot proprioception noise and error, we additionally confront force-torque signal noise and error. To address this, we collect static signal before moving into contact in order to zero the sensor and fit the signal noise to a Gaussian distribution. We use the detected noise to form Σ_m, the calibrated sensor noise used in CPFGrasp.

C. Noise Injection in Simulation

To mimic what we observe in real-world hardware, we inject noise into the MultiSCOPE input $\Gamma_{EE} = [F;T] \in \mathbb{R}^{6\times1}$ for each arm. For each action, we separately calculate the magnitude of the force $||F||$ and torque $||T||$. We define two Gaussian distributions $\mathcal{N} \sim (0,\sigma)$ where σ_F is defined as $n\%$ of $||F||$ and σ_T as $n\%$ of $||T||$. We modify the MultiSCOPE input to be $\Gamma_{EE,noise} = \Gamma_{EE} + [\delta_F;\delta_T]$ where $\delta_F \sim \mathcal{N}(0,\sigma_F) \in \mathbb{R}^{3\times1}$ and $\delta_T \sim \mathcal{N}(0,\sigma_T) \in \mathbb{R}^{3\times1}$. Results with the wrench tool for $n = 5\%$ and $n = 8\%$ are shown in Sec. V-A3.

Algorithm 3 MultiSCOPE: $_t \rightarrow$ tool, $_p \rightarrow$ probe

```
procedure MultiSCOPE($\boldsymbol{\Gamma}_{EE,t}, \boldsymbol{\Gamma}_{EE,p}$)
    $\mathrm{H}_t \leftarrow \{\mathrm{H}_{t,0}, \cdots, \mathrm{H}_{t,N_{opp}}\}$    ▷ Init OPPs
    $\mathrm{H}_p \leftarrow \{\mathrm{H}_{p,0}, \cdots, \mathrm{H}_{p,N_{opp}}\}$    ▷ Init OPPs
    $\mathrm{H} \leftarrow (\mathrm{H}_t, \mathrm{H}_p)$
    $\boldsymbol{\Gamma} \leftarrow (\boldsymbol{\Gamma}_{EE,t}, \boldsymbol{\Gamma}_{EE,p})$
    $\boldsymbol{S} \leftarrow 1/N_{opp}$    ▷ Initially uniform OPP scores
    $r_{cc} \leftarrow \{\}$    ▷ Initially empty contact cloud
    $N_A \leftarrow$ Number of actions
    $f_t \leftarrow \mathrm{segmentTool}(\mathbf{n}_{\mathcal{S},t}, n_{clusters,t}, \varepsilon_t, n_{min,t})$
    $f_p \leftarrow \mathrm{segmentTool}(\mathbf{n}_{\mathcal{S},p}, n_{clusters,p}, \varepsilon_p, n_{min,p})$
    $f \leftarrow (f_t, f_p)$
    for $i \leftarrow 0;\ i < N_A;\ i++$ do
        if $i$ is 0 then
            $\mathrm{H}, \mathrm{S} \leftarrow \mathrm{SCOPE}(\mathrm{H}, \boldsymbol{\Gamma}, f, r_{cc}, i)$
        else
            $\mathrm{H}, \mathrm{S} \leftarrow \mathrm{SCOPE}(\mathrm{H}, \boldsymbol{\Gamma}, f, \bar{\boldsymbol{S}}, r_{cc}, i)$
        end if
    end for
    return $\mathrm{H}, \mathrm{S}$
end procedure
```

D. Task-Based Demonstration

The goal of MultiSCOPE is to estimate in-hand tool poses that are sufficiently accurate to use in manipulation tasks. In order to demonstrate the efficacy of our method, we use the wrench tool to approach a screw in our environment,

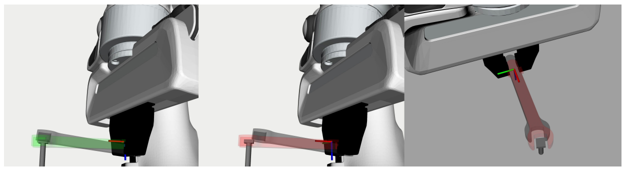


Fig. 9. Examples of success and failure cases from the task-based demonstration of the Action 1 results shown in Table II. The grey wrench tool is the underlying ground truth used for collision checking. In green on the far left is a wrench tool pose estimate that succeeded in surrounding the screw head. The middle and right examples, in red, show wrench tool pose estimates that failed to complete the task. We observe that X-error is very influential in determining task success.

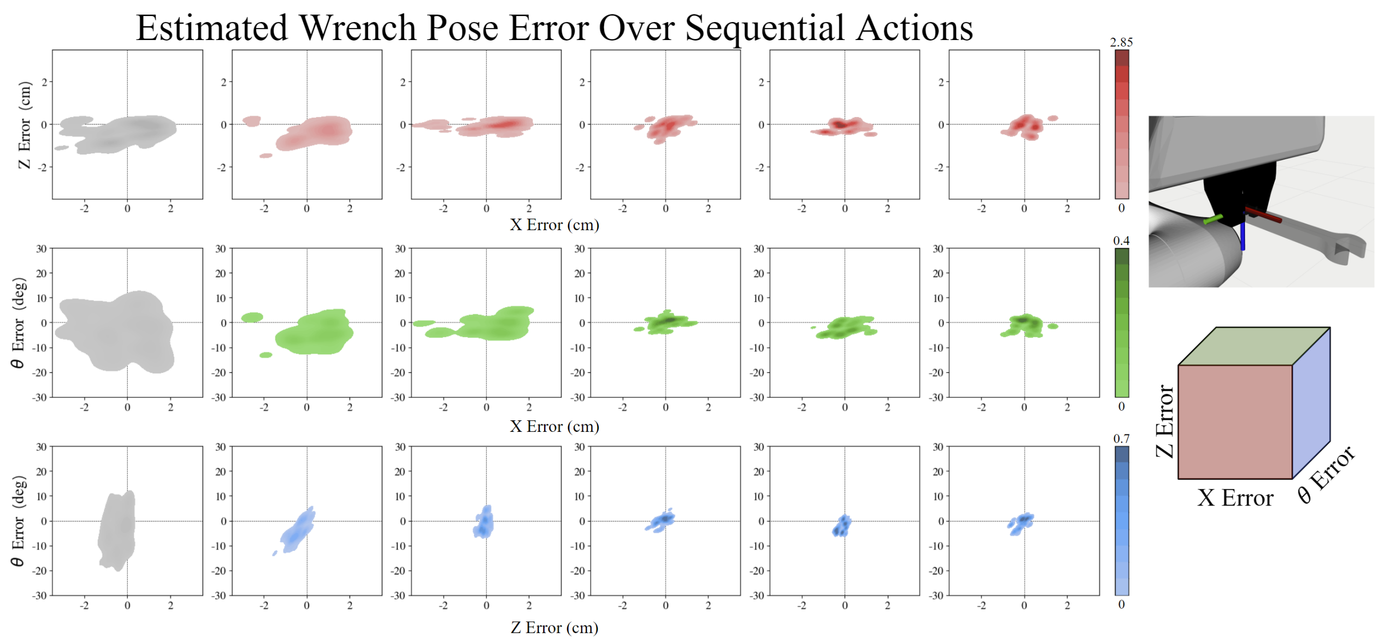


Fig. 10. These results are the combined output of all 10 MultiSCOPE trials with the wrench tool. The error is presented in $[x, z, \theta]_{EE}$, which is shown on the upper right with x in red, y in green, and z in blue. We visualize this 3D error as 3 separate planes: X-Z (red), X-θ (green), and Z-θ (blue). For clarity, this is shown in the cube on the lower right. The first column (grey) shows the error at initialization and every subsequent column shows the results after taking an additional action. We can see that as MultiSCOPE takes more actions, the pose estimate error shrinks and has higher density about $[0, 0, 0]_{EE}$.

in preparation to tighten the screw. We assign a planning frame based on the estimated wrench tool pose and plan to an approach point. We then use the Pilz Industrial Motion Planner to plan a linear Cartesian path into contact between the wrench tool and the screw head. We qualify the trial as a success if the wrench tool is able to surround the screw head. Since we are using the ground truth geometry as the collision body, our estimate of wrench tool pose must agree with the ground truth in order to plan successfully.

We conduct this demonstration in the simulation environment as well as the real world, but with slightly different tolerances. The opening of our wrench tool (which is the 3D printed version of a wrench tool sold by McMaster-Carr) measures $\frac{1}{2}$" (12.7 mm). Therefore, a square-headed bolt must have a minimum side length of 0.35" (8.9 mm) in order to catch in the mouth of the wrench when the wrench is turned. This leaves at most 0.075" (1.9 mm) tolerance on either side of the screw head. We created a mesh for this square-headed screw and added it to our environment for the simulation demonstration. For the real-world demonstration, we increased the maximum tolerance to 3.5 mm on each side by foregoing a screw head and using just a cylinder.

V. Results

A. Simulated Experiments

For brevity, we present only the results for the wrench tool here. The pose estimation results for the hex key, gear, and pawl tools can be found in Appendix A. For the simulated experiments, we generated a set of actions according to Section IV-A and chose 5 of them to be advantageous for reachability. We ran 10 MultiSCOPE trials on this set of actions. We show these 5 actions in Fig 8 and present the results of pose estimation and task completion here.

1) Pose Estimation: The robot grasping the wrench tool achieved 0.47 $\pm$ 0.27 cm translation error and -0.76 $\pm$ 2.01○ rotation error. For the probe, the robot achieved 0.49 $\pm$ 0.28 cm translation error and -2.67 $\pm$ 4.76° rotation error. We visualize the progression of wrench tool pose estimation results over several actions in Fig 10.

2) Task Completion: In the task-based demonstration with tight tolerance, the robot succeeded in 8 of 10 trials to surround the screw head with the wrench. In the 2 trials that failed, we found that after Action 1 the pose estimate of the wrench tool maintained relatively large X-error. These poses were maintained in the contact cloud for the tool because they did not meet the qualification for memory dropout mentioned in Sec. III-C3. Because there was a similar trial that succeeded, we hypothesize that a better-chosen value of Δ_C would improve our success rate.

3) Noise Injection: We sampled force-torque noise at 5% and 8% of each action's magnitude according to Sec. IV-C and tested in 5 trials. At 5% noise, we found that the robot grasping the wrench tool achieved 0.52 $\pm$ 0.31 cm average translation error and -0.1 $\pm$ 3.08° rotation error. The robot grasping the probe achieved 0.58 $\pm$ 0.48 cm of translation error and 0.76 $\pm$ 5.38° of rotation error. Additionally, the robot was able to complete 4 of 5 task-based demonstrations. At 8% of each action's magnitude, we found that MultiSCOPE was no longer able to achieve results that were sufficiently accurate to complete any demonstrations successfully. The results at

TABLE I
RESULTS OF ABLATION STUDY ACROSS LOSS METRICS $\mathcal{L}_P, \mathcal{L}_C, \mathcal{L}_F, \mathcal{L}_\Gamma$, AND $\mathcal{L}_M$ OVER 5 TRIALS.

Loss Ablation Study Results									
Loss	H_t **Trans Error (cm)** ↓		H_t **Rot Error (deg)** ↓		H_p **Trans Error (cm)** ↓		H_p **Rot Error (deg)** ↓		**Task Success** (%) ↑
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	
$\mathcal{L}_P$	1.32	0.74	6.55	18.26	0.89	0.35	3.67	26.77	0
$\mathcal{L}_C$	0.80	0.31	-0.21	2.60	0.74	0.40	2.63	5.30	40
$\mathcal{L}_F$	1.19	0.59	-2.95	3.26	1.29	0.70	-7.29	14.68	40
$\mathcal{L}_\Gamma$	0.46	0.27	-1.26	2.10	1.06	0.64	-5.36	14.83	80
$\mathcal{L}_M$	1.30	0.41	-1.98	2.47	1.65	0.66	-6.06	20.64	20
$\mathcal{L}_P, \mathcal{L}_C, \mathcal{L}_F$	0.66	0.37	-2.13	2.64	0.82	0.44	0.08	11.63	N/a
$\mathcal{L}_P, \mathcal{L}_C, \mathcal{L}_F, \mathcal{L}_\Gamma$	0.61	0.31	-0.47	2.92	0.84	0.47	2.45	11.64	N/a
$\mathcal{L}_P, \mathcal{L}_C, \mathcal{L}_F, \mathcal{L}_M$	0.51	0.31	-1.15	2.79	0.78	0.49	-0.07	11.80	N/a
Full Method	0.48	0.16	-0.50	1.63	0.50	0.25	-1.71	4.23	100

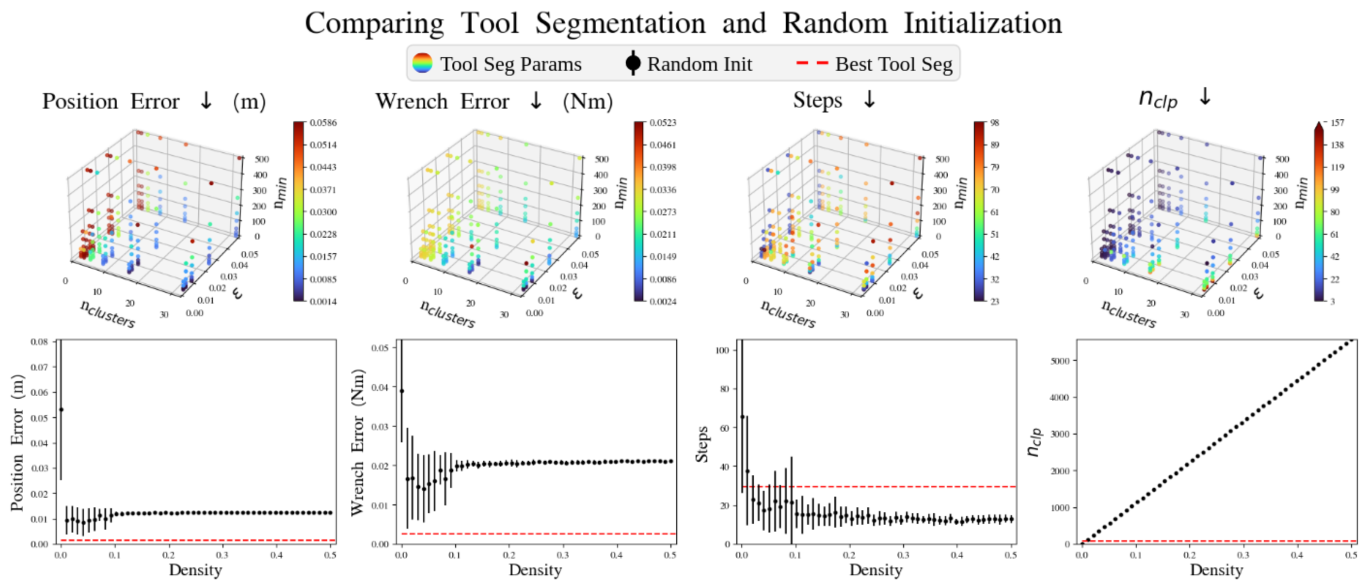


Fig. 11. In this figure we compare the results of our tool segmentation algorithm vs. random CLP initialization when using CPFGrasp to estimate contact location with Poke 4 from Fig. 8. We use 4 metrics: the position error, wrench error, number of steps to converge, and the number of CLPs. These results are averaged across 20 random seeds: error bars are shown in the bottom row for random initialization. The 3 parameters for tool segmentation are $n_{clusters}$, ε, and n_{min} while the single parameter for random initialization is the density of the random sampling (fraction of surface points). We find that our tool segmentation method is able to reduce the number of CLPs needed to converge to the ground truth contact location precisely, as shown with red dashed lines.

TABLE II
RESULTS OF ABLATION STUDY ACROSS STANDALONE ACTIONS OVER 10 TRIALS.

Action Ablation Study Results									
Action	H_t **Trans Error (cm)** ↓		H_t **Rot Error (deg)** ↓		H_p **Trans Error (cm)** ↓		H_p **Rot Error (deg)** ↓		**Task Success** (%) ↑
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	
1	1.15	0.48	-3.71	3.62	0.51	0.27	-2.36	5.58	20
2	1.72	1.11	-0.23	2.46	0.58	0.23	-1.79	5.59	10
3	0.84	0.33	-2.25	2.94	0.6	0.36	-0.71	6.99	70
4	0.88	0.71	-2.27	2.67	0.94	0.42	2.77	9.28	50
5	0.83	0.46	-0.86	2.56	0.75	0.30	3.51	10.93	50
Mean	1.08	0.62	-1.86	2.85	0.68	0.32	0.28	7.67	40
Full Method	0.47	0.27	-0.76	2.01	0.49	0.28	-2.67	4.76	80

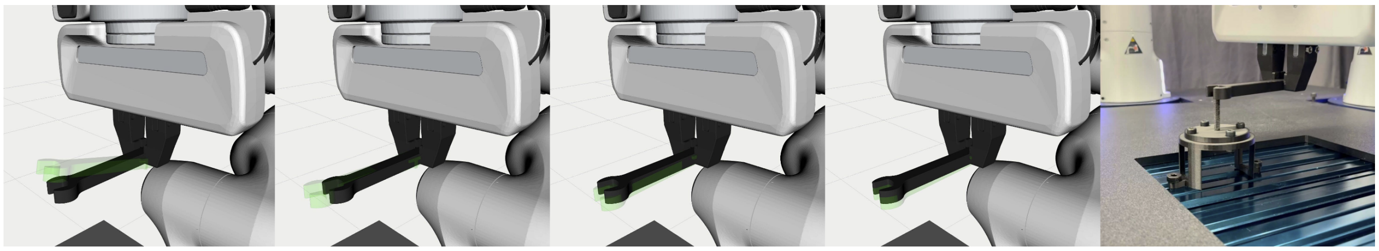


Fig. 12. Estimated wrench pose (green, transparent) compared to ground truth (black, opaque) after Action 1 through after Action 4. At the far right, the result of planning using the estimated wrench pose after Action 4 in the real world. The robot succeeded in the task in all 5 real-world trials.

this 8% noise were 1.13 $\pm$ 0.33 cm translation error and 0.87 $\pm$ 1.27° rotation error for the wrench tool and 0.94 $\pm$ 0.62 cm translation error and -1.31 $\pm$ 5.09° rotation error for the probe.

4) Ablation Studies: We conducted several ablation studies to examine how our selected loss metrics impacted our results. In addition to examining the effect of each loss metric by itself, we also compared the results of performing only a single action. We present these results both in terms of their $[x, z, \theta]_{EE}$ errors as well as their task success rate. Results of these ablation studies are presented in Table I and Table II.

B. Hardware Demonstrations

For the real robot demonstration, we chose and collected data for 4 actions and ran 5 MultiSCOPE trials. 3 of the 4 actions are shown in Fig. 1. The last action is in the same direction as Poke 3 on a different face of the wrench. We present the results of the task-based demonstration trials in Fig. 12. We successfully completed the task in 5 of 5 trials despite the robot proprioception error discussed in Sec. IV-B.

VI. Conclusion

In this paper, we demonstrated the efficacy of MultiSCOPE in disambiguating in-hand object poses using robot proprioception and tactile feedback. We evaluated our method in both simulated and real world environments using Franka Emika Panda robots and saw improved performance from using multiple actions rather than single actions. These improvements enabled us to use the grasped tool for task-based demonstrations.

Building on the success of this work, we look ahead to more principled action selection to reduce the number of actions needed to converge to accurate object pose estimates. We also consider integrating vision or visuotactile sensing as relevant extensions to our method to increase efficiency and applicability.

References

[1] David Álvarez, Máximo A Roa, and Luis Moreno. Tactile-based in-hand object pose estimation. In *Iberian Robotics conference*, pages 716–728. Springer, 2017.

[2] Robert Andre, Michael Jokesch, and Ulrike Thomas. Reliable robot assembly using haptic rendering models in combination with particle filters. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1134–1139, 2016.

[3] Joao Bimbo, Shan Luo, Kaspar Althoefer, and Hongbin Liu. In-hand object pose estimation using covariance-based tactile to geometry matching. *IEEE Robotics and Automation Letters*, 1(1):570–577, 2016.

[4] S.R. Chhatpar and M.S. Branicky. Localization for robotic assemblies using probing and particle filtering. In *Proceedings, 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.*, pages 1379–1384, 2005.

[5] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[6] Guoguang Du, Kai Wang, Shiguo Lian, and Kaiyong Zhao. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review. *Artificial Intelligence Review*, 54(3):1677–1734, 2021.

[7] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 1996.

[8] Xiaoran Fan, Daewon Lee, Lawrence Jackel, Richard Howard, Daniel Lee, and Volkan Isler. Enabling low-cost full surface tactile skin for human robot interaction. *IEEE Robotics and Automation Letters*, 2022.

[9] Steffen Haidacher and Gerd Hirzinger. Estimating finger contact location and object pose from contact measurements in 3d grasping. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 2, pages 1805–1810. IEEE, 2003.

[10] Katharina Hertkorn, Maximo A. Roa, Carsten Preusche, Christoph Borst, and Gerd Hirzinger. Identification of contact formations: Resolving ambiguous force torque information. In *2012 IEEE International Conference on Robotics and Automation*, pages 3278–3284, 2012.

[11] Kyuhei Honda, Tsutomu Hasegawa, Toshihiro Kiriki, and Takeshi Matsuoka. Real-time pose estimation of an object manipulated by multi-fingered hand using 3d stereo vision and tactile sensing. In *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No. 98CH36190)*, volume 3, pages 1814–1819. IEEE, 1998.

[12] Mia Kokic, Danica Kragic, and Jeannette Bohg. Learning to estimate pose and shape of hand-held objects from rgb images. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3980–3987. IEEE, 2019.

[13] Michael C. Koval, Matthew Klingensmith, Siddhartha S. Srinivasa, Nancy S. Pollard, and Michael Kaess. The manifold particle filter for state estimation on high-dimensional implicit manifolds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4673–4680, 2017.

[14] Nathan F. Lepora, Yijiong Lin, Ben Money-Coomes, and John Lloyd. DigiTac: A DIGIT-TacTip hybrid tactile sensor for comparing low-cost high-resolution robot touch. *IEEE Robotics and Automation Letters*, 7(4):9382–9388, Oct 2022.

[15] Shile Li, Haojie Wang, and Dongheui Lee. Hand pose estimation for hand-object interaction cases using augmented autoencoder. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 993–999. IEEE, 2020.

[16] Daolin Ma, Siyuan Dong, and Alberto Rodriguez. Extrinsic contact sensing with relative-motion tracking from distributed tactile measurements. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11262–11268. IEEE, 2021.

[17] Lucas Manuelli and Russ Tedrake. Localizing external contact using proprioceptive sensors: The contact particle filter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5062–5069. IEEE, 2016.

[18] Miquel Oller, Mireia Planas, Dmitry Berenson, and Nima Fazeli. Manipulation via membranes: High-resolution and highly deformable tactile sensing and control. In *Conference on Robot Learning, CoRL 2022*, volume 205 of *Proceedings of Machine Learning Research*, pages 1850–1859. PMLR, 2022.

[19] Shuvo Kumar Paul, Muhammed Tawfiq Chowdhury, Mircea Nicolescu, Monica Nicolescu, and David Feil-Seifer. Object detection and pose estimation from rgb and depth data for real-time, adaptive robotic grasping. In *Advances in Computer Vision and Computational Biology*, pages 121–142. Springer, 2021.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[21] Javier Romero, Hedvig Kjellström, Carl Henrik Ek, and Danica Kragic. Non-parametric hand pose estimation with object context. *Image and Vision Computing*, 31 (8):555–564, 2013.

[22] Brad Saund and Dmitry Berenson. Clasp: Constrained latent shape projection for refining object shape from robot contact. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1391–1400. PMLR, 08–11 Nov 2022.

[23] Andrea Sipos and Nima Fazeli. Simultaneous contact location and object pose estimation using proprioception and tactile feedback. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3233–3240, 2022.

[24] Yuichi Taguchi, Tim K. Marks, and Haruhisa Okuda. Rao-blackwellized particle filtering for probing-based 6-dof localization in robotic assembly. In *2010 IEEE International Conference on Robotics and Automation*, pages 2610–2617, 2010.

[25] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[26] Jonathan Tremblay, Stephen Tyree, Terry Mosier, and Stan Birchfield. Indirect object-to-robot pose estimation from an external monocular rgb camera. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4227–4234. IEEE, 2020.

[27] Mark Van der Merwe, Dmitry Berenson, and Nima Fazeli. Learning the dynamics of compliant tool-environment interaction for visuo-tactile contact servoing. In *Conference on Robot Learning, CoRL 2022*, volume 205 of *Proceedings of Machine Learning Research*, pages 2052–2061. PMLR, 2022.

[28] Maria Bauza Villalonga, Alberto Rodriguez, Bryan Lim, Eric Valls, and Theo Sechopoulos. Tactile object pose estimation from the first touch with geometric contact rendering. In *Conference on Robot Learning*, pages 1015–1029. PMLR, 2021.

[29] Bowen Wen, Chaitanya Mitash, Sruthi Soorian, Andrew Kimmel, Avishai Sintov, and Kostas E Bekris. Robust, occlusion-aware pose estimation for objects grasped by adaptive hands. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6210–6217. IEEE, 2020.

[30] Youngsun Wi, Pete Florence, Andy Zeng, and Nima Fazeli. Virdo: Visio-tactile implicit representations of deformable objects. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3583–3590, 2022.

[31] Akihiko Yamaguchi and Christopher G Atkeson. Tactile behaviors with the vision-based tactile sensor fingervision. *International Journal of Humanoid Robotics*, 16 (03):1940002, 2019.

[32] Sheng Zhong, Zhenyuan Zhang, Nima Fazeli, and Dmitry Berenson. Tampc: A controller for escaping traps in novel environments. *IEEE Robotics and Automation Letters*, 6(2):1447–1454, 2021.

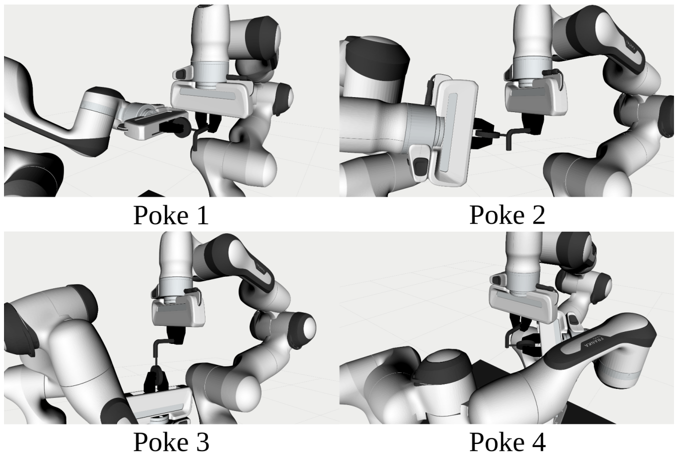
[33] Sheng Zhong, Nima Fazeli, and Dmitry Berenson. Soft tracking using contacts for cluttered objects to perform blind object retrieval. *IEEE Robotics and Automation*

Letters, 7(2):3507–3514, 2022.

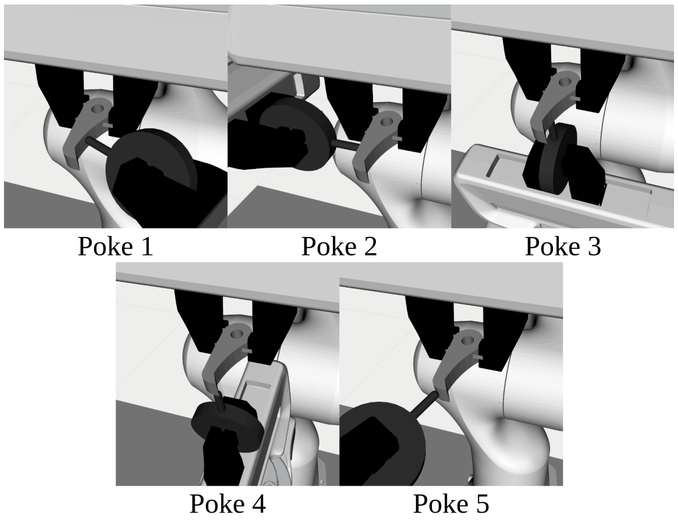
[34] Adrian Zwiener, Christian Geckeler, and Andreas Zell. Contact point localization for articulated manipulators with proprioceptive sensors and machine learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 323–329. IEEE, 2018.

APPENDIX A
EXTENDED RESULTS

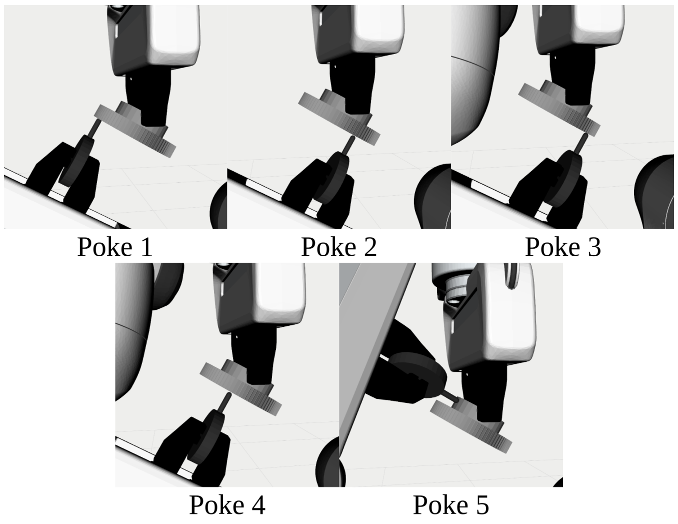
For brevity, we excluded these results from the main body of the paper. We demonstrate that MultiSCOPE is effective at disambiguating in-hand object poses for three additional tools: a hex key, pawl, and gear. We take the simulated actions shown in Fig. 13 and conduct 10 trials with each tool. These results are compared to the MultiSCOPE results with the wrench tool in Table III. We find that the results are consistent across tools. We visualize the pose estimation results for the additional tools in Figs. 14, 15, and 16 in the same style as Fig. 10.



(a) Hex Key Actions



(b) Pawl Actions



(c) Gear Actions

Fig. 13. Selected poking actions for MultiSCOPE trials in simulation with the hex key, pawl, and gear tools.

TABLE III
RESULTS OF MULTISCOPE WITH DIFFERENT TOOLS OVER 10 TRIALS.

Tool	H_t **Trans Error (cm)** ↓		H_t **Rot Error (deg)** ↓		H_p **Trans Error (cm)** ↓		H_p **Rot Error (deg)** ↓	
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev
Wrench	0.47	0.27	-0.76	2.01	0.49	0.28	-2.67	4.76
Hex Key	0.31	0.16	0.00	3.03	0.30	0.18	2.29	3.60
Pawl	0.33	0.22	2.23	4.25	0.63	0.57	-2.48	10.96
Gear	0.49	0.40	-2.66	2.49	1.10	0.57	-3.48	12.34

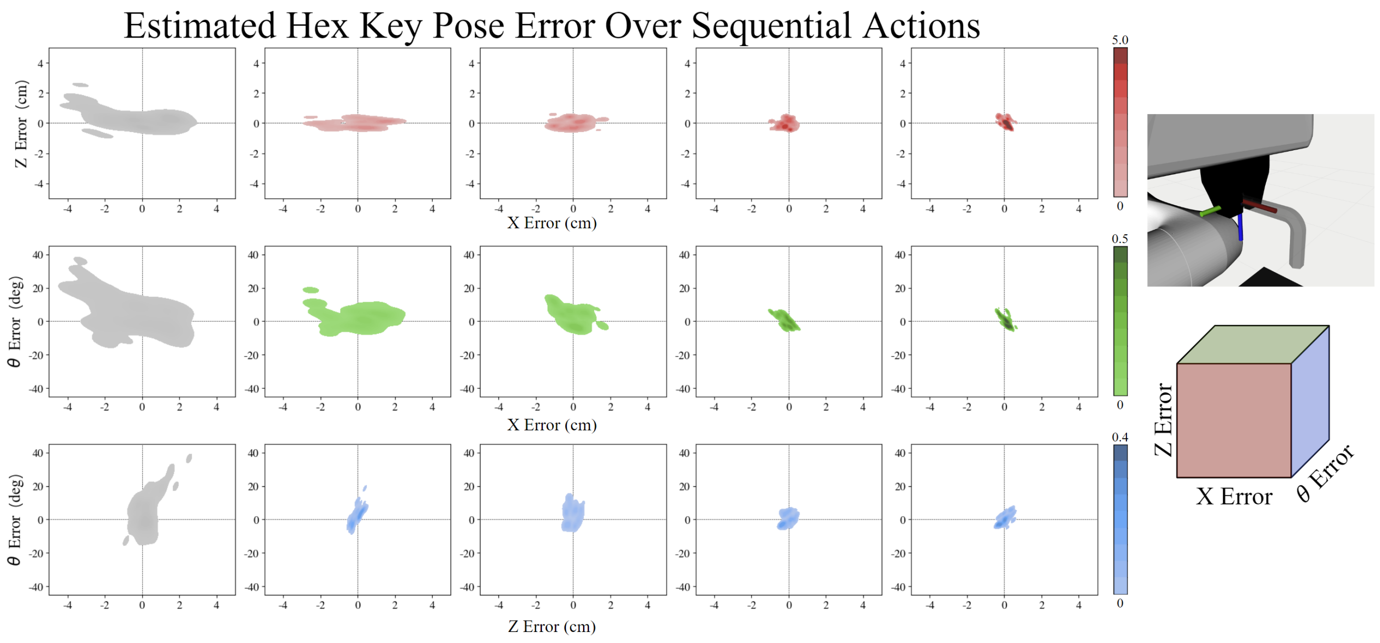


Fig. 14. These results are the combined output of all 10 MultiSCOPE trials with the hex key tool. The error is presented in $[x, z, \theta]_{EE}$, which is shown on the upper right with x in red, y in green, and z in blue. We visualize this 3D error as 3 separate planes: X-Z (red), X-θ (green), and Z-θ (blue). For clarity, this is shown in the cube on the lower right. The first column (grey) shows the error at initialization and every subsequent column shows the results after taking an additional action.

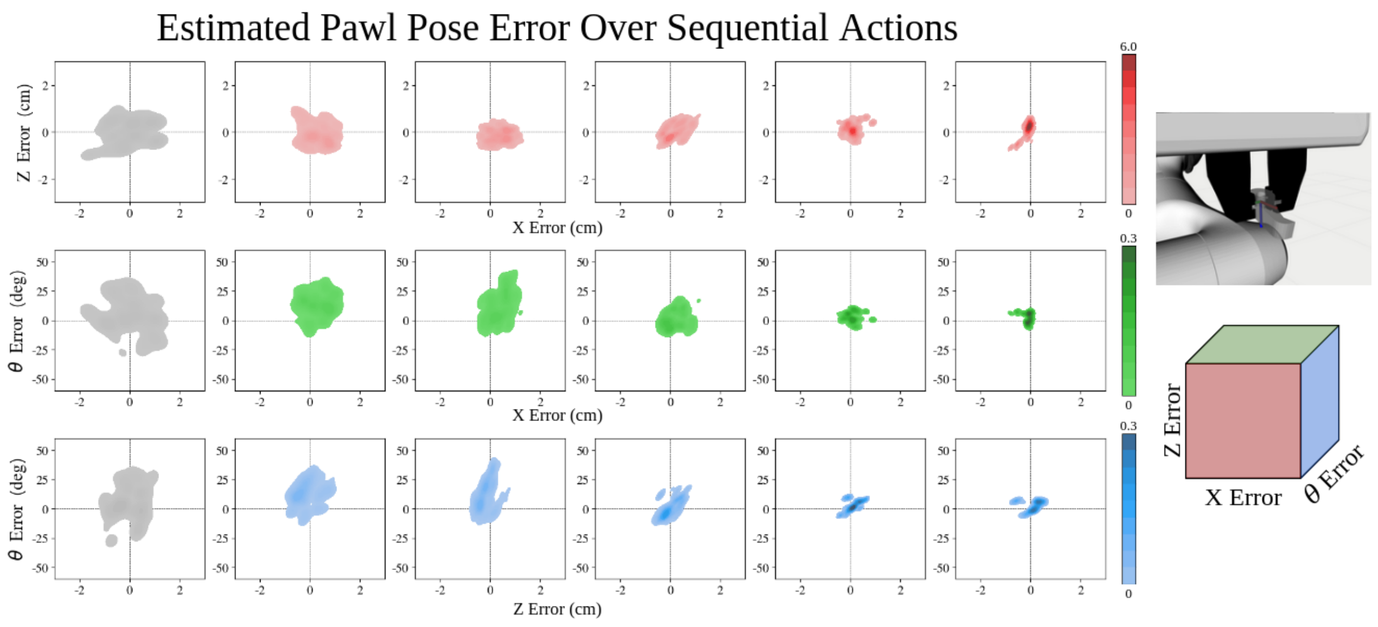


Fig. 15. These results are the combined output of all 10 MultiSCOPE trials with the pawl tool. The error is presented in $[x, z, \theta]_{EE}$, which is shown on the upper right with x in red, y in green, and z in blue. We visualize this 3D error as 3 separate planes: X-Z (red), X-θ (green), and Z-θ (blue). For clarity, this is shown in the cube on the lower right. The first column (grey) shows the error at initialization and every subsequent column shows the results after taking an additional action.

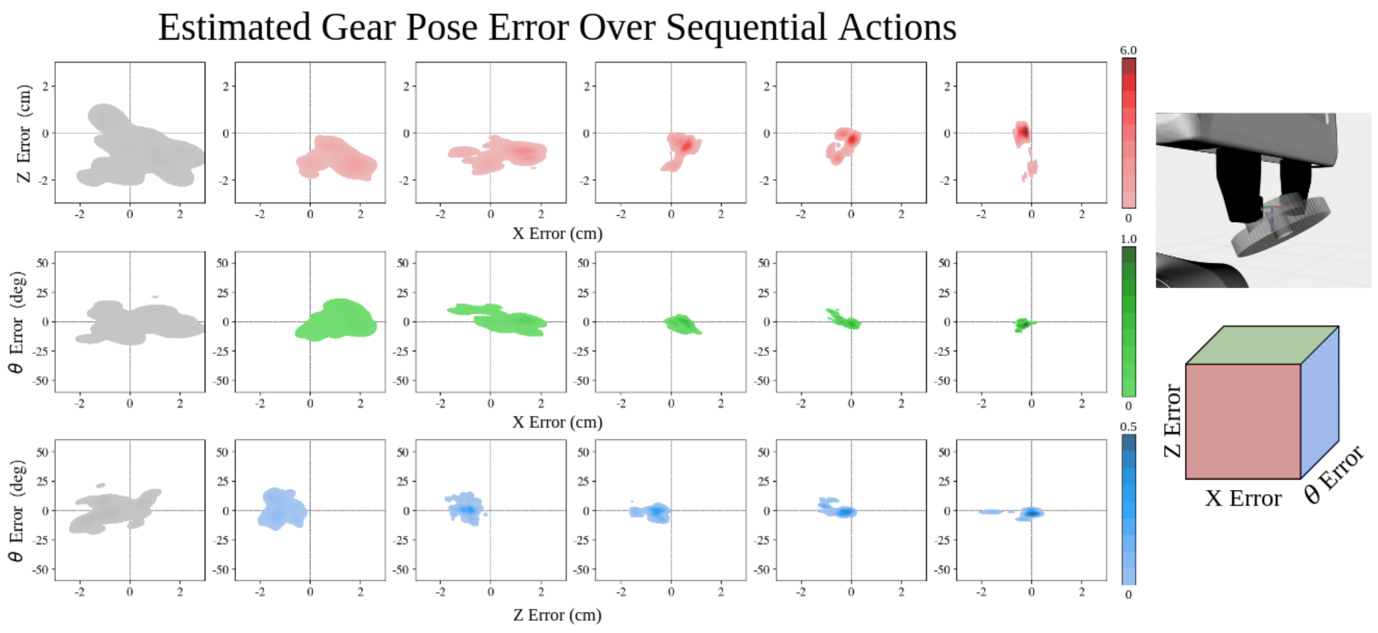


Fig. 16. These results are the combined output of all 10 MultiSCOPE trials with the gear tool. The error is presented in $[x, z, \theta]_{EE}$, which is shown on the upper right with x in red, y in green, and z in blue. We visualize this 3D error as 3 separate planes: X-Z (red), X-θ (green), and Z-θ (blue). For clarity, this is shown in the cube on the lower right. The first column (grey) shows the error at initialization and every subsequent column shows the results after taking an additional action.