Online 3D Edge Reconstruction of Wiry Structures From Monocular Image Sequences

Hyelim Choi, *Graduate Student Member, IEEE*, Minji Lee, *Graduate Student Member, IEEE*, Jiseock Kang, and Dongjun Lee, *Member, IEEE*

***Abstract*—Three-dimensional (3D) reconstruction of wiry structures from vision suffers from thin geometry, lack of texture, and severe self-occlusions. We propose an online 3D edge reconstruction framework that uses monocular image sequences to reconstruct the wiry structures whose skeletons are mainly straight as commonly found in the real world. To reconstruct such structures in an efficient manner, we employ straight edges constructed from points as underlying primitives of the representation. This is to address the harsh geometric nature of wiry objects (e.g., severe self-occlusion) and also to avoid a typically expensive line matching process. Specifically, we first construct sparse 3D points by tracking feature points, while simultaneously refining the camera poses via a robust maximum a posteriori (MAP) inference. These sparse points are then used to generate edge candidates and the belief of each candidate is updated in a Bayesian fashion using a likelihood evaluated on the image observation. Finally, we take the set of 3D edges with beliefs greater than a threshold and apply a post-processing step to reject false edges. We experimentally validate our framework using real-world wiry objects and demonstrate a manipulation task using the reconstruction. The proposed framework exhibits superior performance over state-of-the-art algorithms for the class of wiry structures and the potential to be easily used for subsequent robotic tasks.**

***Index Terms*—Multi-view stereo, structure from motion, edge reconstruction, monocular vision, wiry objects.**

I. Introduction

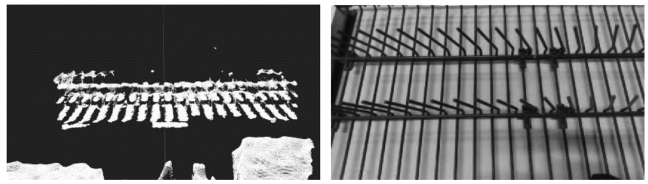
VISION provides abundant opportunities for perception in a variety of applications such as hand tracking [1] and three-dimensional (3D) reconstruction. Especially, 3D reconstruction is a way to understand environments and it poses an important, yet, challenging problem in robotics and computer vision research. It is especially challenging for wiry structures which are not easily captured even with commercial depth cameras (see Fig. 1), and the image is a promising sensing modality to capture the wiry appearance. However, the reconstruction using images is still not straightforward since the wiry objects are low-textured and frequently self-occluded.

Manuscript received 26 May 2023; accepted 4 September 2023. Date of publication 27 September 2023; date of current version 5 October 2023. This letter was recommended for publication by Associate Editor M. Li and Editor M. Vincze upon evaluation of the reviewers' comments. This work was supported by Samsung Research. *(Corresponding author: Dongjun Lee.)*

The authors are with the Department of Mechanical Engineering, IAMD and IOER, Seoul National University, Seoul 08826, South Korea (e-mail: helmchoi@snu.ac.kr; mingg8@snu.ac.kr; jskang0894@snu.ac.kr; djlee@snu.ac.kr).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3320022, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3320022



(a) (b)

Fig. 1. Snapshots of (a) a point cloud and (b) an image of IKEA Kungsfors dish drainer captured with RealSense D435i. The wiry structures are visible in the image but they do not appear in the point cloud.

The 3D reconstruction of such wiry structures based on images can be addressed from many perspectives, and these can be categorized mainly into four distinct classes with respect to the representation each approach adopts: points, lines, curves, and implicit representation.

1) Point Reconstruction: In this method, points are extracted from images, matched, and triangulated into the 3D space, as done in most structure from motion (SfM) approaches [2], [3]. The points are favorable due to their easy matching based on visual appearance and straightforward evaluation of the reprojection residual. This class of methods produces a 3D point cloud, which, yet, is usually very sparse and thus not typically informative enough to fully reconstruct low-textured scenes (e.g., wiry structures) which are of our interest. To achieve a more informative model, a dense point cloud can be generated using subsequent dense reconstruction algorithms such as multi-view stereo (MVS) [4], [5]. However, this is inefficient in memory and time because a massive number of points and a significant amount of time are required.

2) Line Reconstruction: Lines contain richer information than points and thus can represent the scene in a more compact way. Most of the approaches use line segments extracted from images whose correspondences are established to reconstruct 3D lines. But their correspondences are hard to establish using the appearance because they have unstable endpoints and line descriptors are costly to compute and match. As a result, many studies avoid visual matching of lines and call upon other correspondence cues (e.g., epipolar geometry from camera pose estimates [6], [7]). However, the use of line segments detected in images is disadvantageous for scenes of our interest (i.e., wiry structures) because the line segments are often severely fragmented, resulting in unsuccessful matching. An alternative approach was utilized in [8] where a 3D wireframe was reconstructed using points and edges connecting them. Although the overall direction shares some similarities, our proposed framework differs

from it, particularly in that we offer a systematic methodology that produces 3D points and constructs scored edges from the points, yielding a probabilistic 3D edge inference. More precisely, [8] relied on 2D graphs of edges and vertices detected in images, and evaluated their three-dimensional validity using counts and some heuristic rules based on vertex types.

3) Curve Reconstruction: Curves are general representation primitives that can fully depict arbitrary scenes, but as for lines, they are hard to match and most of the approaches in this class exploit epipolar geometry. Wire arts were reconstructed in [9] from images with known camera poses, and a depth sensor was additionally utilized in [10]. An optimization framework was introduced in [11] where camera poses and sample points of curves were jointly optimized. In [12], curves were matched with the additional guidance of sparse points from SfM. Despite their capability to reconstruct general scenes, these works require segmentation masks [9], [10], [11], and their computation is typically expensive since general curves can only be handled by a sampled set of points on them.

4) Implicit Representation: These approaches adopt neural networks to perceive 3D scenes given multi-view 2D images, where the networks themselves are the implicit representations. Most of the methods train either a neural radiance field (NeRF) or a neural signed distance function (SDF) on images with known camera poses, which can render images in novel views [13], [14]. However, the implicit representation requires additional processing to be used for other tasks (e.g., another neural network to obtain manipulation actions or conversion to an explicit form such as a mesh). The methods also typically involve offline learning with a large number of images that requires high-end GPUs and takes a huge amount of time.

In this letter, we propose a novel online 3D edge reconstruction strategy for wiry structures that uses monocular image sequences. We employ lines as primitives of the reconstruction for a compact and informative representation, yet, we adopt straight edges connecting points rather than lines detected in images. This is to circumvent difficulties coming from the geometric nature of the wiry structures (e.g., line segments fragmented by severe self-occlusion) and the methodology so devised would obviate the typically expensive and challenging line matching process. Specifically, we first construct sparse 3D points by tracking feature points, while simultaneously refining the camera poses via a robust maximum a posteriori (MAP) inference. These sparse points are used to generate 3D edge candidates, and the belief of each being a real 3D edge is updated in a Bayesian fashion with a likelihood evaluated on the image observation. Then we take the set of edges with beliefs greater than a threshold and a post-processing is applied to filter out false edges. The proposed algorithm is experimentally validated using real-world wiry objects with a camera mounted on a robot manipulator, and we demonstrate a manipulation task using the reconstruction. This shows that our framework is able to effectively produce a more complete representation of wiry structures than state-of-the-art methods, rendering it promising for a variety of robotic tasks.

Our framework is advantageous over existing algorithms in that: 1) the reconstruction is more informative than sparse point-based methods; 2) it is more efficient in time/memory than dense point/curve-based methods; 3) it circumvents the challenging line matching and is robust to self-occlusion unlike line-based methods; 4) it does not require segmentation which is needed in many curve-based methods; and 5) it can be easily used for subsequent robotic tasks (e.g., household manipulation) without laborious offline learning and high-end hardware, in contrast to implicit representation methods.

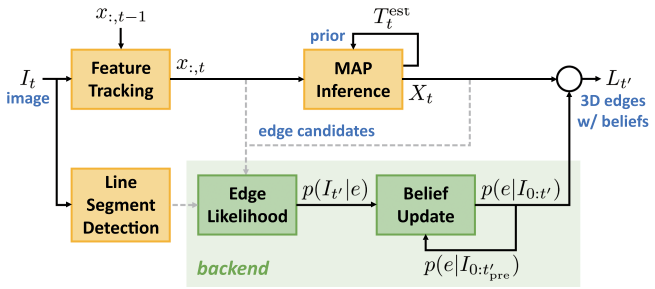


Fig. 2. Overall architecture of our proposed 3D edge reconstruction framework of wiry structures.

The rest of the letter is organized as follows. In Section II, detailed methods of the framework are described. Section III presents the experimental evaluation results on various wiry objects and a manipulation demonstration. Finally, Section IV concludes the letter and suggests future research directions.

II. Methods

In our setting, a camera is mounted on a platform (e.g., robotic manipulator) equipped with sensors (e.g., joint encoder) so that the camera pose can be estimated. We assume that camera intrinsic parameters are known, which can be easily achieved using the manufacturer-provided interface or through an offline calibration process beforehand. The overall architecture of our method is illustrated in Fig. 2. Our method is composed of two components: 1) sparse 3D point generation; and 2) edge inference. In the first component, 2D feature points are detected and tracked, and 3D points corresponding to these 2D points in multiple views are obtained while refining the camera poses simultaneously. The second component runs in the backend and is triggered when it is not running and there is a new image and pose measurements processed in the first component. Using the new information, edge candidates are generated by connecting the feature points, and the beliefs of them being real edges are updated in a Bayesian way. After the algorithm is terminated, the obtained edges are inspected in a post-processing step, where the edges are projected onto selected images saved during the algorithm operation and the ones that are not likely are rejected. We would like to note that by considering edges generated from points, we can circumvent a burdensome process of line segment matching required in other works on line reconstruction.

A. Sparse 3D Point Generation

For the initial image of the sequence, we extract feature points on it using image gradient-based methods (e.g., [15]), with the distances between them greater than a certain value. Then the points are tracked in subsequent images via optical flow (e.g., [16]). Using the feature correspondences obtained from the tracking, we formulate a maximum a posteriori (MAP) inference as a joint optimization to generate 3D points from the feature points and to correct the camera poses. To regulate

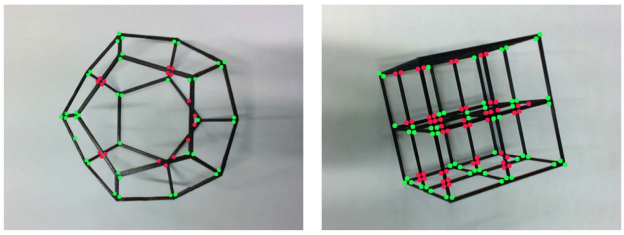


Fig. 3. Feature points detected in images of wiry objects. The red points are fake points artificially created by self-occlusion.

the computational load, we confine the estimation in a sliding window of size N_w to optimize for the last $N_{w,\text{opt}}$ camera poses and the related 3D points while leaving the rest of them fixed to the previously optimized values.

In case there are many incorrectly tracked features, the optimization is corrupted, which leads to inaccurate results. This is especially significant for wiry structures since there frequently occurs a severe self-occlusion that creates many fake points, as depicted in Fig. 3. These points do not actually correspond to physical 3D points, thus they must be excluded from the optimization for correct results. To exclude them, we need to identify them from the inconsistency of their correspondence given the accurate camera poses (i.e., the fake points would not triangulate to a single point). In turn, the camera poses can only be precisely estimated given the right feature correspondences. Since we know neither the right feature correspondences nor the accurate camera poses, we estimate both simultaneously using an expectation-maximization (EM) algorithm [17] for the MAP inference as a means of robust estimation. This is achieved by introducing a latent variable (z in (1)), which indicates whether each feature point is a good feature to be included in the optimization.

The MAP inference is as the following equation

$$\Theta = \underset{\Theta}{\text{argmax}}\, p(\Theta|x) = \underset{\Theta}{\text{argmax}} \sum_{z} p(z, \Theta|x) \tag{1}$$

where $\Theta = [X, T]$, $X = \{X_i \in \mathbb{R}^3 \mid i = 1, 2, \ldots, N_{\text{pts}}\}$ is the set of 3D points, $T = \{T_t \in SE(3) \mid t = 1, 2, \ldots, N_t\}$ is the set of camera poses, $x = \{x_{it} \in \mathbb{R}^2 \mid i = 1, 2, \ldots, N_{\text{pts}}, t = 1, 2, \ldots, N_t\}$ is the set of feature points (x_{it} is the point on the t-th image corresponding to X_i), and $z = \{z_{it} \in \{0, 1\} \mid i = 1, 2, \ldots, N_{\text{pts}}, t = 1, 2, \ldots, N_t\}$ is the set of latent variables with z_{it} associated to x_{it} ($z_{it} = 1$ if x_{it} is a good feature). Using independence relations, $p(z, \Theta|x)$ is factorized as

$$\begin{aligned} p(z, \Theta|x) &= \frac{p(x|z, \Theta)p(z|\Theta)p(\Theta)}{p(x)} \\ &\propto p(x|z, \Theta)p(\Theta) \\ &= \prod_{i,t} p(x_{it}|z_{it}, \Theta) \cdot \prod_{t} p(T_t) \end{aligned} \tag{2}$$

where each term is modeled as follows:

1) The conditional probability $p(x_{it}|z_{it}, \Theta)$ is defined as

$$p(x_{it}|z_{it}, \Theta) = \begin{cases} \frac{1}{A} \exp\left(-\frac{\delta_{it}\|\pi(X_i, T_t) - x_{it}\|^2}{2\sigma_x^2}\right), & z_{it} = 1 \\ \eta, & z_{it} = 0 \end{cases} \tag{3}$$

where $\pi(X_i, T_t) \in \mathbb{R}^2$ is the projection of the 3D point X_i on the image with the camera pose T_t, $\delta_{it} \in \{0, 1\}$ is an indicator of the existence of the corresponding feature of X_i on the t-th image ($\delta_{it} = 1$ if exists), $\sigma_x \in \mathbb{R}$ is the standard deviation related to the uncertainty of the feature tracking, $A = \sigma_x\sqrt{2\pi}$ is the normalizer, and η is a constant. It implies that if the feature is a correctly-tracked one, the probability is greater with a smaller reprojection error; otherwise, the probability is just uniform (i.e., no implication).

2) The prior probability $p(T_t)$ is modeled as

$$p(T_t) = \frac{1}{\sigma_T\sqrt{2\pi}} \exp\left(-\frac{\|T_t - T_t^{\text{est}}\|^2}{2\sigma_T^2}\right) \tag{4}$$

where $T_t^{\text{est}} \in SE(3)$ is the current best-estimated camera pose (i.e., measured camera pose in case it has not been optimized before, and previously optimized pose otherwise) and $\sigma_T \in \mathbb{R}$ is the standard deviation related to the measurement accuracy.

To solve the MAP inference, we adopt the EM algorithm that iterates over E- and M-steps detailed below.

1) E-Step: We estimate the distribution of z using the old parameter Θ_{old} from the last M-step to evaluate the expectation of the log-likelihood $\mathbb{E}_{z\sim p(z|x,\Theta^{\text{old}})}[\log p(z, \Theta|x)]$. The distribution $p(z|x, \Theta^{\text{old}})$ is factorized as

$$q(z) := p(z|x, \Theta^{\text{old}}) = \prod_{i,t} q(z_{it}) \tag{5}$$

where $q(z_{it}) := p(z_{it}|x_{it}, \Theta^{\text{old}}) \propto p(x_{it}|z_{it}, \Theta^{\text{old}})$. Thus we can calculate the distribution using the conditional distribution defined in (3) as follows.

$$\begin{aligned} q(z_{it} = 1) &= \frac{p(x_{it}|z_{it} = 1, \Theta^{\text{old}})}{p(x_{it}|z_{it} = 0, \Theta^{\text{old}}) + p(x_{it}|z_{it} = 1, \Theta^{\text{old}})} \\ &= \frac{\frac{1}{A}\exp\left(-\frac{\delta_{it}\|\pi(X_i, T_t) - x_{it}\|^2}{2\sigma_x^2}\right)}{\eta + \frac{1}{A}\exp\left(-\frac{\delta_{it}\|\pi(X_i, T_t) - x_{it}\|^2}{2\sigma_x^2}\right)} \end{aligned} \tag{6}$$

2) M-Step: Here, we solve for the new parameter Θ^{new} that maximizes the expectation of the log-likelihood evaluated in the E-step using the distribution $q(z)$ in (5) as follows.

$$\begin{aligned} \Theta^{\text{new}} &= \underset{\Theta}{\text{argmax}}\, \mathbb{E}_{z\sim q(z)}[\log p(z, \Theta|x)] \\ &= \underset{\Theta}{\text{argmax}} \sum_{i,t} \mathbb{E}_{z_{it}\sim q(z_{it})}[\log p(x_{it}|z_{it}, \Theta)] \\ &\quad + \sum_{t} \mathbb{E}_{z\sim q(z)}[\log p(T_t)] \\ &= \underset{\Theta}{\text{argmin}} \sum_{i,t} \frac{q(z_{it} = 1)}{2\sigma_x^2}\delta_{it}\|\pi(X_i, T_t) - x_{it}\|^2 \\ &\quad + \sum_{t} \frac{1}{2\sigma_T^2}\|T_t - T_t^{\text{est}}\|^2 \end{aligned} \tag{7}$$

The M-step boils down to a weighted bundle adjustment (BA) with a prior on camera poses, and the entire EM algorithm falls into the form of iteratively reweighted least squares. This formulation offers robustness to camera pose uncertainties

and incorrect feature tracking, and can be easily solved with off-the-shelf optimization tools since it is basically a weighted least squares problem. The solution is guaranteed to converge to a stationary point, but it is not guaranteed to converge to the global optimum. There could be certain hard cases for the algorithm to find the optimal solution (e.g., when many fake feature points coincidentally agree with the same 3D structure), but it rarely happens in practical cases. The MAP is better suited to challenging scenes of our interest over other robust estimation methods such as RANSAC because our method directly provides joint optimization of 3D points and camera poses from multiple images and also allows us to straightforwardly incorporate prior knowledge of the camera pose. Robust kernels (e.g., Huber loss) are also not sufficient for the settings considered here because they just weight down the outliers heuristically rather than excluding them.

In practice, we anneal σ_x by decaying them with a constant ratio τ to prevent getting stuck in a bad stationary point. To regulate the computational complexity, we limit the number of iterations of the EM algorithm and of the optimization in the M-step. The initial values of new (i.e., not optimized before) variables are set to the measured camera poses and 3D points triangulated from the measured relative pose. Note that this initialization of variables, with the introduction of prior on camera poses, allows the scale to be recovered even though we only use monocular vision. After the optimization, the optimized points whose reprojection errors are all smaller than a threshold $r_{\max}$ (e.g., $2\,[\mathrm{px}]$) are considered acceptable points. For points with large errors, if such points had been acceptable before, they are determined to have lost good track and are excluded from future optimizations. But if such a point had been optimized more than a certain number of times (e.g., 4), it is still regarded as acceptable and included in the edge inference using the 2D point projected on the image. Otherwise, if it had been optimized less, it is marked as unacceptable and excluded also from the edge inference. Unacceptable points are excluded from future optimizations because they arise from either fake feature points created by self-occlusion or wrong feature tracking, which can lead to incorrect results. To save resources and gather more information, points that are either unacceptable or sufficiently optimized are removed from the feature points to track. For each new image, we take the neighborhood of the current feature points and the projected points of the 3D points acceptable so far, and extract additional feature points outside this neighborhood. The number of additional feature points is constrained to limit the number of acceptable points and consequently the number of edge candidates to evaluate the likelihood of.

Here the major parameters are σ_x, σ_T, η, τ, and $r_{\max}$. First, the suitable values of σ_x and σ_T do not vary much for each target scenario but can be tuned to adjust the relative significance of reprojection/prior costs. Second, η has a trade-off (i.e., if it becomes larger, there is a greater chance of overfitting and rejecting too many points; and if it becomes smaller, more bad feature tracking may not be excluded) but we found that there is little need to adjust by scenario. Third, the inadequate decay rate τ can degrade the reconstruction by leading the solution to suboptimal solutions, and we found that 2 is a good starting point. Finally, $r_{\max}$ also has a trade-off (i.e., if it becomes larger, more feature points are tolerated that can hinder camera pose and 3D point estimation; and if it becomes smaller, the reconstruction becomes more accurate but many of the points can be rejected) but the optimal value does not vary much for each case and can be set to a constant.

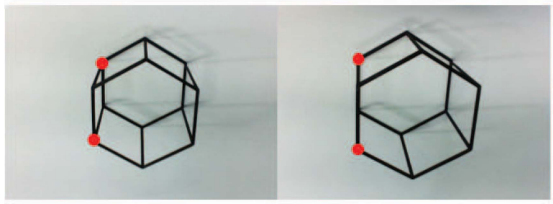


Fig. 4. Misleading appearance of a single image. The red points are actually unconnected (left) but look connected due to the occlusion (right).

B. Edge Inference

As we have generated 3D points, now we infer the edges that connect them using the 2D images. Since a single image can be misleading (e.g., an unconnected set of points could appear to be connected by occlusion as shown in Fig. 4), we fuse the information from multiple images of different views. We achieve this via the Bayesian inference by incrementally updating the beliefs of the connections of two points being real edges. First, we take current acceptable feature points that are successfully tracked. Then the corresponding 3D points of these points are projected on the current image and we generate edge candidates by connecting these projected points. Although we can generate $n(n-1)/2$ candidates from n points, we only make connections of lengths smaller than a predefined value $l_{\max}\,[\mathrm{px}]$ because in many cases long edges can be replaced with multiple shorter edges.

The belief of each candidate is a conditional probability of being a real edge given observations and is updated as

$$p(e|I_{0:t'}) = \frac{p(e|I_{0:t'_{\mathrm{pre}}})p(I_{t'}|e)}{p(I_{t'}|I_{0:t'_{\mathrm{pre}}})} \tag{8}$$

where $e \in \{0,1\}$ is the indicator of being a real edge (i.e., 1 if it is and 0 otherwise), $I_{t'}$ is the t'-th image observation, t'_{pre} is the previous image index, $I_{t_1:t_2}$ is the image observations with indices from t_1 to t_2, $p(e|I_{0:t'_{\mathrm{pre}}})$ is the previous belief, $p(I_{t'}|e)$ is the likelihood of the t'-th observation, and $p(I_t|I_{0:t'_{\mathrm{pre}}})$ is the normalizer $(=\sum_e p(e|I_{0:t'_{\mathrm{pre}}})p(I_{t'}|e))$.

The likelihood of an edge candidate is evaluated using line segments detected in the image. First, if the length of the candidate is shorter than a threshold, the likelihood returns 0.5, which does not affect the belief value. To consider the thickness of the wiry structures, we look into an edge band of the candidate depicted in Fig. 5 to calculate the likelihood. We consider the edge band with margins $\epsilon_i\,[\mathrm{px}]$ as illustrated in Fig. 5, and each margin is calculated as

$$\epsilon_i = \mathcal{T}_{\max} f/d_i, \qquad i = 1,2 \tag{9}$$

where $\mathcal{T}_{\max}\,[\mathrm{m}]$ is the maximum thickness that is assumed to be known, $f\,[\mathrm{px}]$ is the focal length of the camera, and $d_i\,[\mathrm{m}]$ is the depth of the endpoint. The margins are determined in inverse proportion to the depths of endpoints because the farther the endpoint is, the smaller the thickness appears. Among the line segments contained in the edge band, we reject ones whose projections on the edge candidate do not lie on the candidate even partially (see the right side of Fig. 5). If the edge band contains

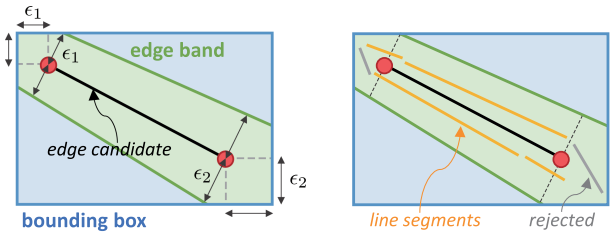


Fig. 5. Edge band of an edge candidate (left) and the line segments in the edge band (right). In the right figure, among line segments in the band (yellow and gray), the segments whose projections on the edge candidate do not intersect with the candidate are rejected (gray).

no line segments, a small constant (e.g., 0.05) is returned as the likelihood value.

Then we evaluate how likely are the survived segments when the candidate is a real edge by defining a metric s.t.,

$$p(I_{t'}|e=1) = \min\{2.5\bar{p}(I_{t'}|e=1) + 0.05,\ 0.99\} \tag{10}$$

$$\bar{p}(I_{t'}|e=1) = \left[\sum_{i=1}^{N_{\text{ls}}}(1-\tanh^2(100\max\{|\sin(\theta_i-\theta^{\text{ref}})| - \sin(\theta^{\text{thk}}),0\}))\cdot(1-0.8\ (\delta_i^{\text{dst}})^2)\cdot l_i\right] \cdot \frac{r_{\text{occ}}^3\cdot(1-0.9\ \min_i(\delta_i^{\text{dst}})^2)}{\sum_{i=1}^{N} l_i} \tag{11}$$

where N_{ls} is the number of line segments in the edge band, θ_i is the angle of the i-th line segment, θ^{ref} is the angle of the edge candidate, $\theta^{\text{thk}} = \text{atan2}(\max\{\epsilon_1,\epsilon_2\}/2, l^{\text{ref}})$ is the tolerance of the angle discrepancy to account for the thickness of objects and different depths of two endpoints, l^{ref} is the length of the candidate, δ_i^{dst} is the minimum distance of the segment to the candidate, l_i is the length of the i-th line segment, r_{occ} is the ratio of the length of union of line segments with respect to ρl^{ref} thresholded below 1, and ρ is a discount rate on the candidate length. The likelihood is set above a small positive value because line segments could be erroneously not detected (e.g., highly cluttered and occluded environments) and once a zero likelihood is returned, there is no chance of restoring a nonzero belief. For the likelihood of not being a real edge, $p(I_{t'}|e=0)$, is defined as

$$p(I_{t'}|e=0) = \max\{1-p(I_{t'}|e=1), 0.5\} \tag{12}$$

which is designed to be greater than 0.5 because candidates that are not real edges may look like edges by occlusion. When the algorithm is terminated, we take the set of 3D edges with beliefs greater than 0.9, and a post-processing step to be described in the next subsection is performed to organize the final reconstruction of the scene. We would like to note that although using line segments to evaluate the likelihood works in general settings, it is not the only way and we can use any likelihood evaluation module that reasonably predicts the likelihood (e.g., neural networks working directly on images).

Some parameters are introduced in the edge inference, including $l_{\max}$ and ρ. First, $l_{\max}$ should be large enough to generate edge candidates while it sacrifices computational load when it is too large. And ρ is introduced to tolerate fragmented line segments and is relevant to the degree to which self-occlusion occurs (i.e., the more occlusion, the smaller ρ should be). These two parameters may be tuned for each target scenario for the best performance while the rest of the parameters can be set to constant values.

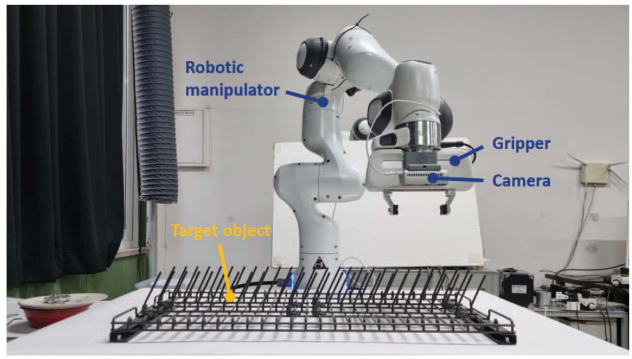


Fig. 6. Experiment setup. A camera (RealSense D435i) is mounted on a gripper (Franka Hand) attached to the end effector of the robotic manipulator (Franka Emika Panda). The target objects are put on the table.

C. Post-Processing

When a point tracking is lost, false edges connected to the point which have looked like real edges mistakenly survive. To reject these edges and also to integrate more information, a post-processing step is introduced. During the operation, we accumulate the number of points that have lost track and check if it exceeds 10% of the number of the current feature points that are being tracked. If exceeded, we store the index of the current frame and if nothing is stored until the operation ends, we store the index of the last frame. Then at the end of the operation, we project the reconstructed 3D edges on the initial image and the images corresponding to the stored indices, evaluate their scores using the likelihood metric, and reject low-scored edges. The threshold used to reject edges is relevant to how accurate the reconstruction should be (i.e., the larger, the more accurate the edges are) and it is empirically tuned to a constant for all of our experiments in Section III. We also check if occlusion takes place for each edge, and do not reject it if it is occluded by other edges. We repeat this process until there is no change because there can be erroneously accepted edges occluded by other rejected edges.

III. Experimental Results

The experiment setup is shown in Fig. 6. The camera mounted on the gripper has its intrinsic parameters known, and the hand-to-eye transformation is calibrated. The image resolution for all evaluations is set to 1280×720 because a state-of-the-art algorithm to test, Line3D++ [6], requires high-resolution images. The algorithm is implemented in C++ and is programmed to run parallel where possible. The visual features are handled with OpenCV (e.g., feature points are detected with Good Features To Track [15], tracked with Lucas-Kanade optical flow [16], and line segments are detected using LSD detector [18]). For the MAP inference we use GTSAM [19], a graph optimization library that is easy to use in vision problems, and Levenberg–Marquardt solver is adopted in the optimization. The evaluations

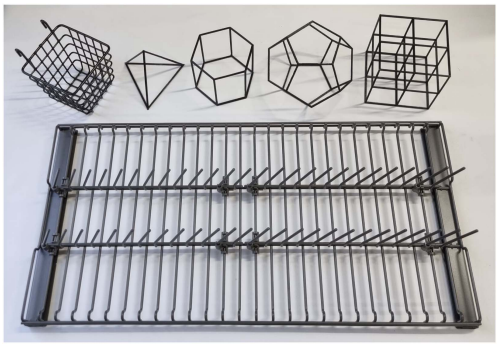


Fig. 7. Wiry objects for the experimental evaluations.

presented in this section are all done on a desktop with an AMD Ryzen 5 3600 6-core $3.59\,[\mathrm{GHz}]$ CPU, a $16\,[\mathrm{GB}]$ RAM, and an NVIDIA GeForce GTX 1660 GPU. Datasets are obtained for target scenarios and our algorithm is run online for every dataset using ROS. The images and joint angle measurements are logged for each loop on which other algorithms are run for comparison.

A. *Comparative Evaluation*

The proposed algorithm is evaluated in comparison with a dense point reconstruction algorithm, COLMAP [5], a line reconstruction algorithm, Line3D++ [6], a curve reconstruction, EdgeGraph3D [12], and an implicit method, NeuS [14]. The wire reconstruction algorithms (e.g., [9]) are excluded here because they require segmentation of objects which renders them not suitable for a fair comparison. All algorithms are fed the same images and camera pose estimates from the robot measurements. Note that COLMAP and NeuS use a GPU while the other methods use a CPU only, and the networks in NeuS are trained for 300000 epochs in all scenarios.

We present evaluation results on wiry objects with known geometric models (a tetrahedron, a hexagonal prism, a dodecahedron, and a $2 \times 2 \times 2$ cube) and practical objects (a wire basket and a dish drainer) shown in Fig. 7. The four objects with known models are made of tubular structures with circular cross-sections of $3\,[\mathrm{mm}]$ in diameter and are manufactured with a 3D printer. The results are depicted in Fig. 8. The points from COLMAP represent the geometry to the level one can recognize it, yet with many outliers. The lines from Line3D++, with many incorrect lines, are not very successful in rendering the geometry (e.g., it struggles to reconstruct the spokes of the dish drainer where the dishes are placed) and many of them are segmented into pieces. EdgeGraph3D generates a much more compact point cloud than COLMAP but it still suffers to depict the geometry in many cases. NeuS also struggles to reconstruct objects in many cases, as the rendered views show. In contrast, the reconstruction using our framework provides a model that mostly describes the object geometry with few outliers. The shortfalls of other algorithms can be attributed to several reasons: 1) thin geometry hinders the depth estimation of the structures; 2) line segments are fragmented into small pieces so that few meaningful reconstructions are made from them; 3) the detected line segments and curves are derived from the boundaries that change as the view changes, making the problem quite ill-posed; 4) a small number of images with a relatively small range of viewpoints are used; and 5) inaccurate camera poses without refinement deteriorate the reconstruction.

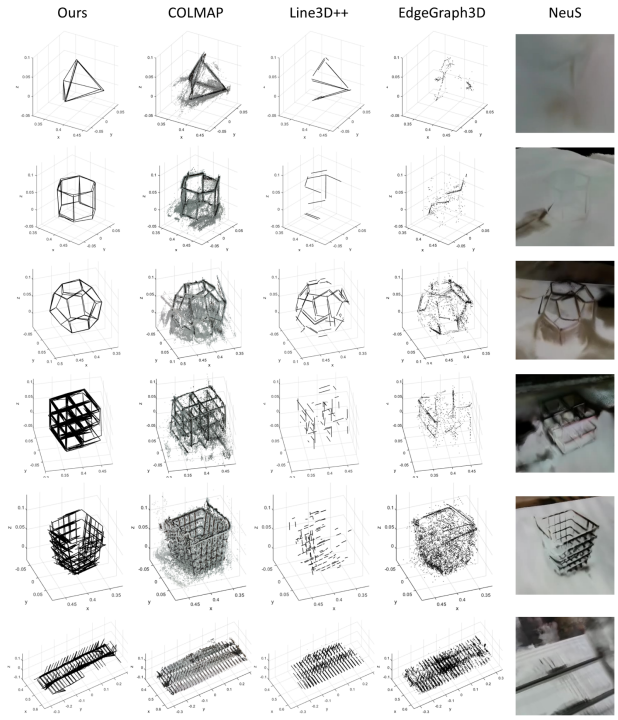


Fig. 8. Reconstruction results on wiry objects. The leftmost column contains the plots of the reconstruction using the proposed framework. The following columns contain the results using COLMAP [5], Line3D++ [6], EdgeGraph3D [12], NeuS [14], respectively. The points of COLMAP are colored using the image colors and the results of NeuS are rendered images given certain viewpoints. Rotating views are available at: https://youtu.be/s1J9GVYt7Fs.

The number of parameters (i.e., 3#points for COLMAP and EdgeGraph3D, 6#lines for ours and Line3D++, and #SDF network parameters for NeuS) and the total execution times are presented in Table I . For a fair comparison, we removed points and lines of the reconstructions (i.e., ours, COLMAP, Line3D++, and EdgeGraph3D) that are outside the tight bounding boxes of the objects. As can be seen in the table, the proposed framework reconstructs compact quantities of geometric primitives that articulate the object geometry and works online with a fast enough post-processing. We argue that running the reconstruction online is beneficial in that we can lower computation and memory requirements and access the reconstructed model anytime during the scanning, bringing an opportunity for active sensing.

Although we have no accurate ground truths of the global poses of the objects, we have the ground truth 3D CAD models of the four objects manufactured with a 3D printer, which are precise up to the accuracy of the 3D printer. We evaluate the errors of the shapes by comparing the reconstructions to the ground truth CAD models. First, we make all reconstructions and the ground truth model into point clouds and then use an iterative closest point (ICP) algorithm to align two point clouds. For the reconstruction of NeuS, we sample points whose SDF values are between $\pm 1\,[\mathrm{mm}]$ and for the rest, we sample a sufficient number of points to uniformly cover the geometry (e.g., 20000 points). After the alignment, the error is computed for each point in the reconstruction as the distance to the closest point in the ground truth, which represents the accuracy of the reconstruction. Although the error can also be calculated for

TABLE I
NUMBER OF PARAMETERS AND TOTAL EXECUTION TIME (UNIT: [s]) FOR RECONSTRUCTION

	Ours		COLMAP		Line3D++		EdgeGraph3D		NeuS	
	# param.	exec. time	# param.	exec. time	# param.	exec. time	# param.	exec. time	# param.	exec. time
Tetrahedron	102	$0.20 * 30 + 0.00$	93216	878.64	54	0.14	1074	3.38	529076	132604.12
Hexagonal prism	348	$0.20 * 30 + 0.00$	133572	846.36	54	0.17	2382	4.31	529076	135092.11
Dodecahedron	576	$0.20 * 30 + 0.00$	252753	898.80	384	0.51	9675	111.15	529076	135488.47
$2 \times 2 \times 2$ *cube*	6600	$0.25 * 30 + 0.11$	149826	867.96	324	0.38	4593	8.57	529076	136295.52
Wire Basket	5010	$0.33 * 50 + 0.03$	300843	1411.98	732	0.87	19419	77.31	529076	137613.54
Dish Drainer	16284	$0.33 * 50 + 0.53$	1694385	1358.70	3756	15.05	68016	307.90	529076	136386.25

The execution time $x * y + z$ in ours means that the algorithm is run at $1/x$ [hz] for y frames and the post-processing takes z [s].

TABLE II
SHAPE ESTIMATION ERROR (RMSE OF SAMPLED POINT CLOUD) OF EACH CASE (UNIT: [mm])

	Ours	CLMP	L3D+	EG3D	NeuS
Tetrahedron	**1.45**	4.04	1.97	7.63	-
Hexagonal prism	**2.31**	3.61	4.69	9.18	5.18
Dodecahedron	**2.77**	3.73	4.34	7.91	6.91
$2 \times 2 \times 2$ *cube*	**2.93**	4.53	5.62	6.41	6.51

COLMAP, Line3D++, and EdgeGraph3D are abbreviated as CLMP, L3D+, and EG3D, respectively.

each point in the ground truth, we do not report this because the dominant source of error captured by this metric (e.g., missing geometry) is already illustrated in Fig. 8. For COLMAP and NeuS, to exclude points from other objects and the background, we align only the points inside a tight bounding box of the object, take points with errors smaller than 10 [mm], and align those points again. The root mean square error (RMSE) values are presented in Table II , and our framework achieves the smallest errors in all cases, which implies that we obtain more accurate models with few outliers. Note that the error of the tetrahedron reconstruction using NeuS is omitted because no meaningful point cloud is obtained from the trained SDF.

B. Diverse Scenes

We study the generalizability and robustness of our framework by evaluating on different object colors, a textured background, and natural/hand-crafted objects.

1) Object Colors: Since we use images converted to grayscale, we experiment with three colors of dodecahedrons: white, gray, and black. The results (Fig. 9(a)) show that when the grayscale object color is similar to that of the background, the reconstruction can face difficulties because feature points and line segments are not extracted well. This is rather a fundamental limitation of using images and could be alleviated by carefully adjusting lighting conditions.

2) Textured Background: We print a painting and use it as the background to test how the texture in the background influences the reconstruction. The result in Fig. 9(b) shows that the hexagonal prism is still reconstructed and the texture of the background is reconstructed as well, providing a holistic reconstruction of the scene.

3) Natural and Hand-Crafted Objects: We also test our algorithm on natural and hand-crafted objects: tree branches, wire art, and handwriting. The wire art was handcrafted from a wire with a circular cross-section of 1.5 [mm] in diameter and the handwriting was written with a whiteboard marker on a piece of paper. The results are presented in Fig. 9(c), (d) and (e), which show that the proposed framework is able to tolerate non-perfectly straight structures of varying thickness and even approximate curves.

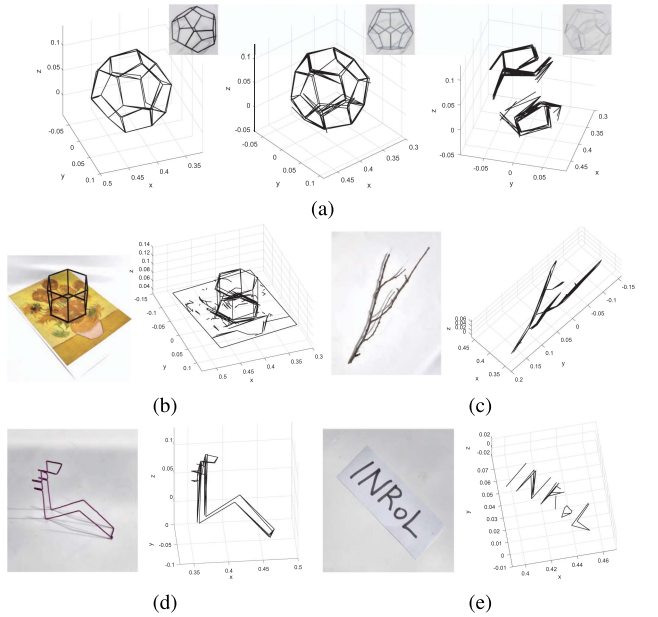


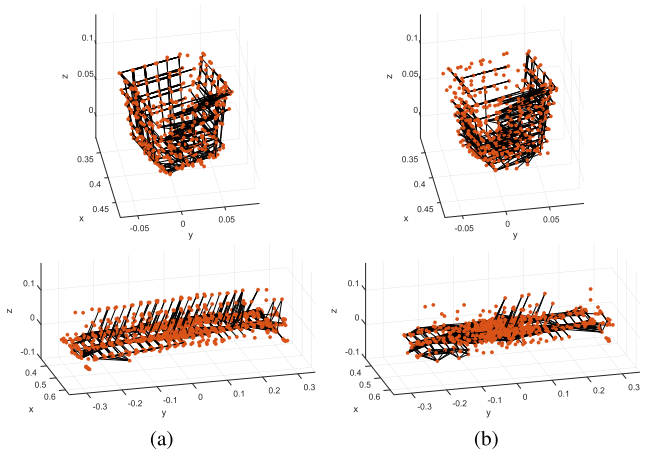
Fig. 9. Reconstruction results of diverse scenes: (a) three different object colors (black, gray, and white from left to right); (b) textured background; (c) tree branches; (d) wire art; and (e) handwriting. The figures on the top right of (a)–(c) and on the left side of (b)–(e) are pictures of each scene.

C. Ablation Study

To highlight the necessity of our MAP inference presented in Section II-A, we present ablation studies on the two practical objects. We compare the proposed inference with a joint optimization in (1) with no latent variable z (i.e., standard BA with pose priors). The computational budget is kept the same by setting the total maximum number of iterations equal. For both objects, the reconstructions from a simple joint optimization without a latent variable have more missing points and lines as can be seen in Fig. 10.

D. Manipulation Demonstration

We demonstrate how our framework can enhance perception for robotic manipulation tasks (e.g., [20]). A tableware manipulation task is performed, where a robot scans a dish drainer and places two dishes in their respective target slots on the wiry dish rack. For the demonstration, the robot is installed with a gripper and an FT sensor (ATI Gamma) for admittance control.



(a) (b)

Fig. 10. Lines and points of the wire basket reconstruction (a) from the proposed; and (b) from a joint optimization with no consideration on the latent variable. The lines are in black and the points are in red.

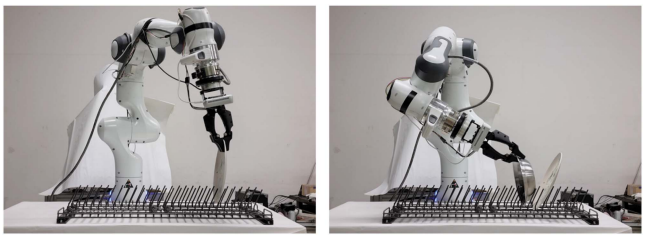


Fig. 11. Snapshots of a robotic tableware manipulation task. Two dishes are successfully placed in a row in the designated slots of the drainer.

We exploit the motion planner used in [21] to place dishes, and the experiment snapshots are presented in Fig. 11. This can only be possible with our framework which accurately estimates the configuration of the wire spokes of the rack and, consequently, the target slot for each dish.

IV. Conclusion

In this letter, we propose an online 3D edge reconstruction framework that recovers scenes with wiry structures from monocular image sequences. For a compact and informative representation, we employ lines as primitives of the reconstruction, but rather than lines detected in images, we adopt straight edges connecting points. This not only enables circumventing a line matching process but also is advantageous for addressing the wiry nature of target scenes because line segments in these scenes are generally fragmented into pieces due to severe self-occlusion. We first detect feature points and track them on the image sequence to construct sparse 3D points while simultaneously revising the camera poses via a robust MAP inference. A set of edge candidates is generated by connecting the feature points and the belief of each candidate is updated in a Bayesian manner using a likelihood metric evaluated on image observations. For the final reconstruction, we obtain a set of probable 3D edges with beliefs greater than a threshold, and a post-processing step is executed to reject false edges. The framework is experimentally validated on the reconstruction of wiry objects with a camera mounted on a robotic manipulator and is proven to offer a more efficient and complete reconstruction compared to state-of-the-art approaches. As our demonstration on a manipulation task shows, the resulting reconstruction can be easily utilized in subsequent robotic tasks.

Although our framework successfully recovers the 3D lines of the structures, our work, as presented here, is not so suitable for reconstructing general curves. By extracting appropriate anchor points on curves, we may better reconstruct general structures and enhance the quality of the resulting model. Additionally, the likelihood calculation in the edge inference may be a bottleneck if the target geometry is more complex. To address this, switching the likelihood estimator to a neural network is a promising direction to speed up the process and also to optimize the metric.

References

[1] Y. Lee, W. Do, H. Yoon, J. Heo, W. Lee, and D. J. Lee, "Visual-inertial hand motion tracking with robustness against occlusion, interference, and contact," *Sci. Robot.*, vol. 6, no. 58, 2021, Art. no. eabe1315.

[2] C. Wu, "Towards linear-time incremental structure from motion," in *Proc. Int. Conf. 3D Vis.*, 2013, pp. 127–134.

[3] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4104–4113.

[4] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards internet-scale multi-view stereo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1434–1441.

[5] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, "Pixelwise view selection for unstructured multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 501–518.

[6] M. Hofer, M. Maurer, and H. Bischof, "Efficient 3D scene abstraction using line segments," *Comput. Vis. Image Understanding*, vol. 157, pp. 167–178, 2017.

[7] D. Wei, Y. Wan, Y. Zhang, X. Liu, B. Zhang, and X. Wang, "ELSR: Efficient line segment reconstruction with planes and points guidance," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 15807–15815.

[8] M. Stephens and C. Harris, "3D wire-frame integration from image sequences," *Image Vis. Comput.*, vol. 7, no. 1, pp. 24–30, 1989.

[9] L. Liu, D. Ceylan, C. Lin, W. Wang, and N. J. Mitra, "Image-based reconstruction of wire art," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–11, 2017.

[10] L. Liu, N. Chen, D. Ceylan, C. Theobalt, W. Wang, and N. J. Mitra, "CurveFusion: Reconstructing thin structures from RGBD sequences," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–12, 2018.

[11] P. Wang, L. Liu, N. Chen, H.-K. Chu, C. Theobalt, and W. Wang, "Vid2Curve: Simultaneous camera motion estimation and thin structure reconstruction from an RGB video," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 132-1–132-12, 2020.

[12] A. Bignoli, A. Romanoni, M. Matteucci, and P. di Milano, "Multi-view stereo 3D edge reconstruction," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2018, pp. 867–875.

[13] B. Mildenhall, P.P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 99–106.

[14] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, 2021, pp. 27171–27183.

[15] J. Shi and C. Tomasi, "Good features to track," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 1994, pp. 593–600.

[16] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, 1981, pp. 674–679.

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Stat. Soc.: Ser. B. (Methodol.)*, vol. 39, no. 1, pp. 1–22, 1977.

[18] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, "LSD: A line segment detector," *Image Process. On Line*, vol. 2, pp. 35–55, 2012.

[19] F. Dellaert et al., "borglab/gtsam," May 2022. [Online]. Available: https://github.com/borglab/gtsam

[20] Z. Liu et al., "OCRTOC: A cloud-based competition and benchmark for robotic grasping and manipulation," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 486–493, Jan. 2022.

[21] J. Lee, M. Lee, and D. J. Lee, "Uncertain pose estimation during contact tasks using differentiable contact features," in *Proc. Robot.: Sci. Syst.*, 2023.