2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Object-based SLAM Using Superquadrics

Yifan Xing[1], Noe Samano[1], Wen Fan[2], and Andrew Calway[1]

***Abstract*— Visual SLAM uses visual information, typically point features, to localise a camera and, at the same time, map the environment. In recent years, there has been interest in using scene-understanding capabilities to enhance the mapping process and object-level SLAM systems have appeared in response. However, most of the previous work is limited to pre-stored object models or pre-trained networks to represent the objects, which limits working scenarios or uses representations with limited scope, such as cubes or quadrics. To address this, we propose to use superquadrics as the object representation and, in this paper, present a proof of principle SLAM system in which object-based mapping is fully integrated with camera tracking via keyframe optimisation. The system was tested on simulated and real datasets, and the results show that the system can achieve lightweight and comparatively good object representation whilst also giving good camera trajectories estimates under certain scenarios.**

I. INTRODUCTION

We present an object-based simultaneous localisation and mapping (SLAM) system using RGB-D sensing. The majority of visual SLAM systems use point features to build a map of an environment and simultaneously track the 3D pose of a vision sensor. These systems have been incredibly successful, finding their way from basic research to real-world applications in a couple of decades. Many operate in real-time and provide robust tracking and mapping capabilities.

Despite this progress, the use of point features can be problematic, not least in terms of the size of the maps that are generated, especially when looking to scale up such systems to operate in very large environments. This has led people to consider alternative map representations, which look to provide more efficient representations of surroundings, as well as seek to address the inherent challenges in matching point features, such as occlusions and lighting or perspective changes. This has ranged from early work on using planar features [1], [2], [3] and a priori known CAD models [4] to more recent work on using object representations such as quadrics [5] or learned class models [6].

We seek to build on the latter by investigating the use of superquadrics (SQs) [7] to represent objects within a SLAM system. SQs have been around for many years and are regularly turned to as a compact and generalised way of representing objects, either in terms of a single SQ or more interestingly, in terms of multiple SQs per object [8]. However, their use in SLAM systems has been far less, with only some initial investigations without full integration into a complete SLAM framework [9], [10].

[1]Y.Xing, N. Samano, A.Calway are with the School of Computer Science, Visual Information Laboratory, University of Bristol, United Kindom
[2]W.Fan is with the Department of Bioengineering, Imperial College London, United Kindom

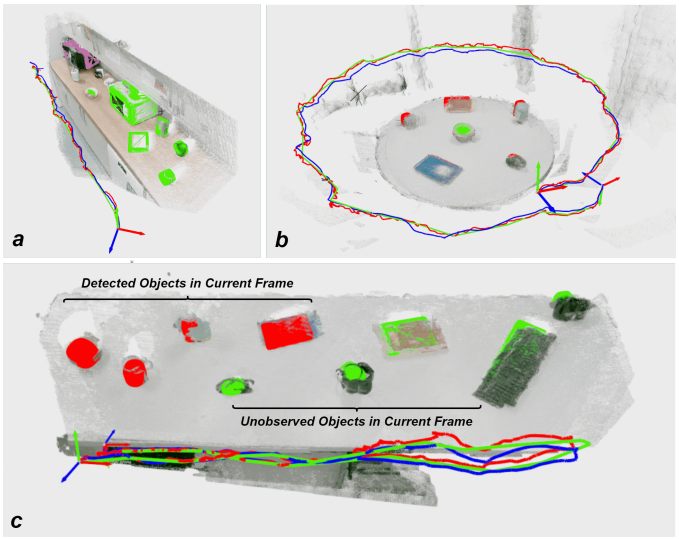

Fig. 1. Example of our SLAM trajectories compare with ORBSLAM3[11]. Three scenarios, including (a) kitchen, (b) round table, and (c) long dining table. The trajectories estimated by our system and ORBSLAM3 are represented in the red and blue lines, respectively, and the green trajectories are the optimised keyframe poses. The background point clouds are based on ORBSLAM3 poses and the keyframe RGBD images. SQs with red color are detected objects with the current frame, while the green ones are unobserved objects in the current moment.

We address this by presenting an initial proof of principle system in which single SQs are used to represent objects in the scene and their parameters optimised simultaneously with tracking 3D sensor pose in a principled manner. The advantage of using SQs for object-based SLAM is that they provide the potential for being more flexible than limited representations such as QuadricSLAM [5] but more generalised than class-dependent learned models as in NodeSLAM [6].

Inspired by the latter, we implement full object-level camera tracking using SQs within an object map representation and probabilistic rendering as used in [6]. We also use SQ-based bundle adjustment, which enables the system to incrementally optimise 3D sensor pose and SQ parameters simultaneously during the tracking process. We show that we can obtain accurate tracking (<1cm Absolute Pose Error, APE) in simulation with SQ-like objects and within 5cm APE compared with ORBSLAM3 [11] for a real scene containing everyday objects. We also show that the quality of the SQ representation is comparable to that obtained by a recent algorithm for optimised fitting of SQs to point clouds [12].

979-8-3503-7769-9/24/$31.00 ©2024 IEEE

II. Related Works

The traditional map representation for SLAM consists of two typical cases. One is the sparse mapping with feature points [13], [14], [11], [15], [16], [17], and the other one is dense surfaces[18], [19], [20], [21]. In comparison, feature points are confusing when describing the specific shape of the scene, while the dense surface is sufficiently detailed but difficult to optimise due to the computational load. Therefore, the current visual SLAM aims to pursue a precise and detailed reconstruction while, at the same time, achieving a more compact and lightweight representation. NeRF[22], with a novel differentiable-rendering technique in recent years, enables the representation of the scene as a neural network, laying great advantages in lightness and efficiency of optimisation and detailed reconstruction. However, most of the NeRF-based SLAMs[23], [24], [25], [26], [27], lack comprehension of the objects in the scene. So, some works explored object-level SLAM with different representations.

In terms of object representation, object-level SLAMs can be classified into four types: pre-stored models, latent-code-based, parametric model-based and NeRF-based. SLAM++[4] pioneered object-level SLAM, which incrementally replaces the detected objects with pre-scanned CAD models. This avoids saving duplicate object models in the map, but it is limited to pre-scanned object models. NodeSLAM[6] compresses the objects' shapes of a particular class in a latent code by pre-training the VAE network in the given class and decompresses them into a representation of dense voxel grids with a decoder. Although its reconstruction can be generalised towards objects of the same category, it is still restricted to the pre-trained classes.

Additional works explored using parametric shapes, including 3D bounding boxes[28], quadrics[5], [29] (mostly ellipsoids) and superquadrics[9], [30] to describe objects. These shapes benefit from representing with few parameters but lack a fine-grained reconstruction of the shapes. CubeSLAM [28] and QuadricSLAM[5], as representatives, describe the object's pose with 3D bounding boxes or ellipsoids. They achieved only parametric mapping and joint optimisation of camera poses but no implementation of object-level visual odometry.

The latest works, RO-MAP[31] and vMap[32], employ NeRF's technique of using neural networks to represent objects and train them while tracking. This enables lightweight object representations and breaks the limitations of pre-trained object classes. However, neither of them achieves camera positioning, and the underlying shape of the object is not obtained from a few observations. Inspired by NodeSLAM and QuadricSLAM, our motivation is to build a complete lightweight object-level SLAM system which is not constrained by pre-trained decoders while providing relatively fine-grained shape reconstruction. Recent works[12], [33], [34], [35] demonstrated the effectiveness of SQs to fit the object's shape from range data or point clouds. They gained impressive results, proving that SQ's parameters are both compact and explainable since each parameter implies a physical meaning. Therefore, we chose SQ, a parametric representation that incorporates multiple shapes. More information about SQs is presented in section III. SQ-SLAM[9] uses SQs as a map representation but not for object-level tracking. Similarly, [30] built a SLAM system with SQs but no camera tracking was implemented at the SQ level, nor was it tested with a real-world dataset. So, we aim to build a SLAM system based only on SQs to achieve camera tracking, SQ mapping, and joint optimisation of both of them.

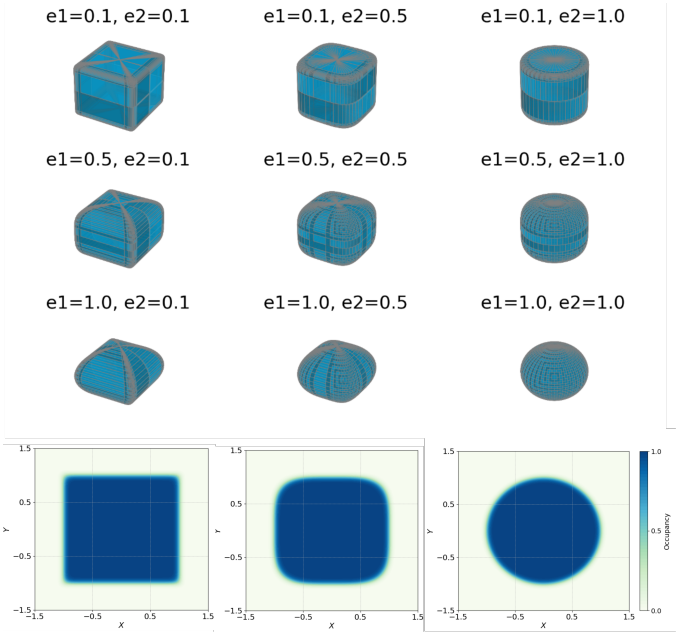

Fig. 2. Superquadric shape examples. The top 3 rows show the effects of the two shape parameters, e_1 and $e2$, from 0.1 to 1.0. The bottom row shows the corresponding occupancy value distribution of e_2 in the range [0.1,1.0] with our modified occupancy function (2).

III. Superquadrics

Superquadric(SQ) is a unified mathematical expression that can be applied to represent cuboids, cylinders, spheres, ellipsoids, etc. Its surface can be represented by an implicit equation named the inside-outside function:

$$f(x,y,z) = \left(\left(\frac{x}{a}\right)^{\frac{2}{e_2}} + \left(\frac{y}{b}\right)^{\frac{2}{e_2}} \right)^{\frac{e_2}{e_1}} + \left(\frac{z}{c}\right)^{\frac{2}{e_1}} \qquad (1)$$

where a,b,c indicate the SQ's size in each of the three dimensions and $e1$,$e2$ define its shape, as shown in Fig. 2. Relying on a,b,c,$e1$,$e2$, a SQ can be represented by its coordinate. In (1), for any coordinate (x,y,z), if $f<1$, the points would be inside of the SQ, and if $f=1$, the points are on the surface of the SQ, and if $f>1$, the points are outside of the SQ. However, the exponents in (1) increase the difficulty of calculating its derivation. Hence, as in [36], we convert the inside-outside function to an occupancy function as follows:

$$Occ(x,y,z) = Sig'(s(1-f^{e_1}(x,y,z))), \qquad (2)$$

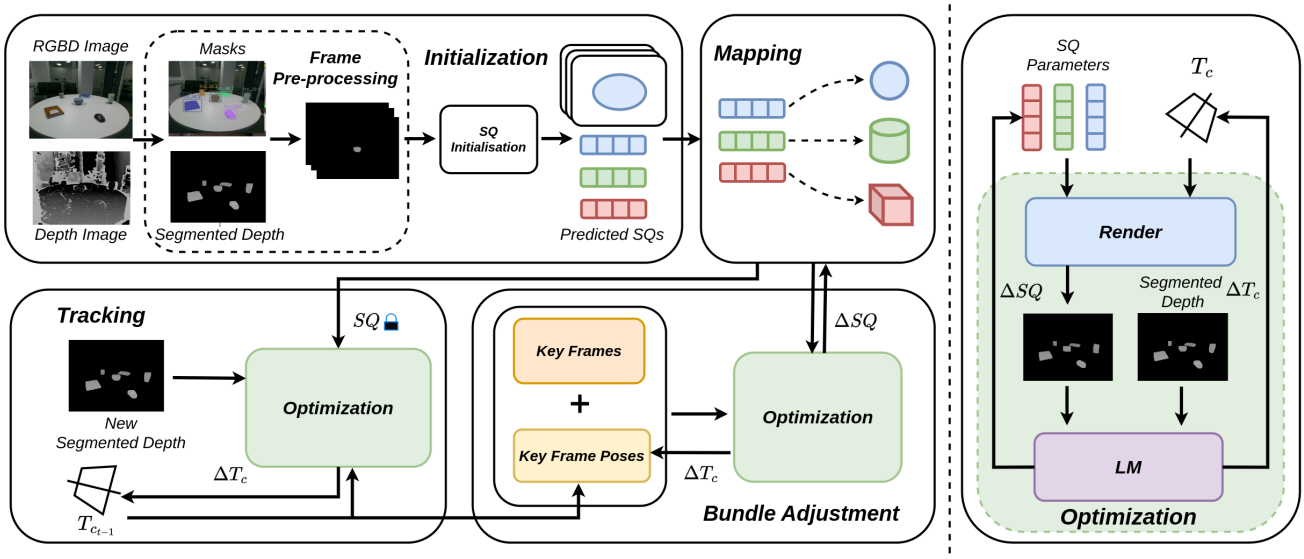

Fig. 3. Architecture of proposed object-based SLAM. The system contains several key parts, including initialisation, camera tracking, mapping and optimisation.

$$Sig'(x) = \frac{1+tanh(x)}{2}, \tag{3}$$

where s represents the sharpness, e_1 is the first shape parameter introduced previously and Sig' is a modified sigmoid function to avoid gradient vanishing during optimisation. In addition, we need to define 3D position and orientation of the SQ in the world coordinate system, which we denote by and normalised quaternion $\mathbf{q} = \{q_1, q_2, q_3, q_4\}$, respectively. Thus, an arbitrary SQ can then be represented by 12 parameters:

$$SQ = \{a, b, c, e_1, e_2, p_1, p_2, p_3, q_1, q_2, q_3, q_4\} \tag{4}$$

IV. OBJECT-LEVEL SLAM WITH SUPERQUADRICS

A. System Overview

The system architecture is illustrated in Fig. 3, comprising of initialisation, camera tracking, mapping and optimisation. Upon receiving a new frame of RGB and depth images, we first employ a pre-trained network to segment them into separate detected objects. During the initialisation stage, new objects are parameterised and stored in a map represented by SQs. In the camera tracking stage, the SQs from the map and the camera pose are utilised in a differentiable renderer to produce a predicted depth image. By minimizing the residual between the predicted depth image and the newly captured depth frame, the current camera pose can be estimated. Within the optimisation module, bundle adjustment is adopted to jointly optimise the keyframe camera poses and all the SQs.

B. Differentiable Rendering with Superquadrics

We use probabilistic rendering as in NodeSLAM [6] with several adjustments to incorporate SQs, notably the way in which raycasting and occupation value calculation are implemented. Firstly, the raycasting process in NodeSLAM sample points at a fixed distance, which may lead to some useless points before the ray reaches the object. To address this, we compute the intersection of the raycasting and the 3D bounding box of SQs and only sample M points within this intersection distance$[P_{near}, P_{far}]$. We use P_i to denote each sampled point. Also, since the occupancy function of a SQ is continuous, we can compute the occupancy probability directly and avoid the voxel grid approximation used in [6].

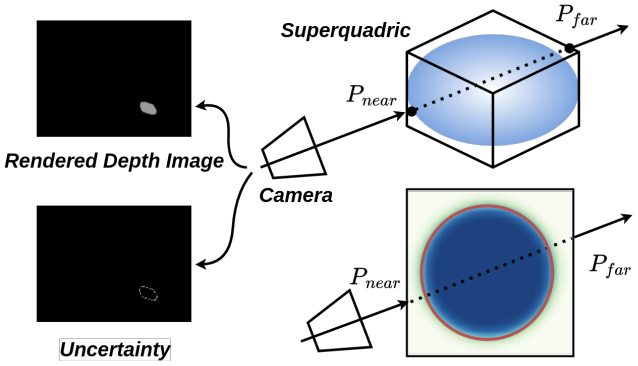

Fig. 4. Probabilistic Render with Superquadrics. Points will be sampled near the bounding box. For each point, the occupancy value will be calculated with (2).

Using a similar notation to that in [6], we define the termination probability ψ_i at depth $d(u,v) = P_i$ at each sampled point on the ray as:

$$\psi_i = prob(d[u,v] = P_i) = Occ_i \prod_{j=1}^{i-1}(1 - Occ_j). \tag{5}$$

where $Occ_i = Occ(P_i(x,y,z))$ and u and v are pixel coordinates in the estimated depth image. If the ray does not

intersect with the SQ, the escape probability is:

$$\psi_{M+1} = prob(d[u,v] > P_{far}) = \prod_{j=1}^{M}(1 - Occ_j), \quad (6)$$

where M is the number of sampled points. The estimated depth $d[u,v]$ is calculated by aggregating the above probability ψ_i and the corresponding distance P_i.

$$d[u,v] = \sum_{i=1}^{M+1} \psi_i P_i \quad (7)$$

Similarly, the depth uncertainty is:

$$d_{var}[u,v] = \sum_{i=1}^{M+1} \psi_i (P_i - d[u,v])^2 \quad (8)$$

Finally, with the probabilistic rendering R, for superquadric SQ, the rendered depth image d and its uncertainty d_{var} from the T_c camera viewpoint is:

$$d[u,v], d_{var}[u,v] = R(T_c, SQ)[u,v] \quad (9)$$

C. Initialisation

To establish an object-oriented SLAM system, the first step is to separate objects from images. Benefiting from the development of segmentation networks in recent years, Yolov8 [37] has been able to achieve high-precision object masking. It actively recognises and segments objects from images by their semantic information. But this also limits it to pre-trained object categories. To be more generalised, we also adopt MobileSAM [38] as an alternative object segmentation method and provide a user-interactive GUI for selecting objects of interest in the first frame. The user only needs to mark the object of interest using bounding boxes as a prompt in the first frame to get the object's segmentation masks. The preliminary tracking with MobileSLAM uses the 2D bounding boxes from the previous frame and expands the boxes as new prompts for the following frame segmentation.

With the masks of the object, it is sufficient to get the depth images of the objects, which can be transformed into point clouds of the objects. We then extract the minimal bounding box of the object's point cloud using the algorithm in Open3D [39] and use the box size, position, and rotation as the corresponding SQ initial parameters $\{a,b,c,p_1,p_2,p_3,q_1,q_2,q_3,q_4\}$. The initial values of the two shape-related parameters e_1 and e_2 are set to 0.1.

Further optimisation of the parameters of the initial SQ is performed using the least squares method. With the initial SQ, the initial camera pose (set as original), and the probabilistic render, we estimate a depth image d and the uncertainty d_{var} of the object represented. Combined with captured depth images D, by constructing the least squares function E in (10), we fix the camera parameters and only minimise this error with respect to the SQ parameters using the Levenberg-Marquardt method. Notably, in this step, we do not optimise all the parameters of the SQ at the same time but group them to separately optimise the shape $\{e_1,e_2\}$ first, and then optimise the size $\{a,b,c\}$ and pose $\{p_1,p_2,p_3,q_1,q_2,q_3,q_4\}$.

$$E = \sum_{u,v} \frac{(d[u,v] - D[u,v])^2}{d_{var}[u,v]} \quad (10)$$

D. Data Association and Tracking with Superquadrics

We have designated a set of attributes for each object to facilitate their identification. These attributes mainly contain tracking ID, class ID, mask, SQ's parameters, and depth image. The tracking ID is a key attribute to identify each object, and the class ID contains the semantic information of the object. The SQ's parameters imply the 3D reconstruction information and the depth image represents its latest point cloud information.

To achieve an accurate tracking ID, we primarily undertake the following steps:

- Tracking ID from Yolov8 tracking mode or the tracking ID from preliminary MobileSAM tracking results introduced previously.
- Masks matching: We match the masks of the current frame and the rendered masks from the SQs. If the IoU is above 0.2, they are identified as the same object.
- 3D bounding box matching: Both above approaches are limited to 2D. If both fail, a more reliable way is to drive the SQ initialisation again to obtain the 3D bounding box. By computing the IoU of the new 3D bounding box and the unmatched objects 3D bounding boxes, the highest one will be selected. If all IoUs are below the threshold, a new object will be added to the map.

Once the data association is finished, camera tracking will be applied by minimizing the residual of E by fixing the SQ's parameters and only optimizing the camera pose T_{c_t} from the last frame pose $T_{c_{t-1}}$ as (11).

$$\min_{T_c} \sum_{u,v} \frac{(d[u,v] - D[u,v])^2}{d_{var}[u,v]} \quad (11)$$

E. Bundle Adjustment with Superquadrics

Key Frames Selection There are three principles for selecting the keyframes:

- If the camera rotation is larger than n degrees or the camera position changed to larger than m centimeters compared to the last keyframe, it will be certificated as a new keyframe. n and m are set as 15 and 10 for practical experience.
- If a new object is added to the map, the current frame will be selected as a keyframe.
- If k frames have elapsed since the last keyframe, the current frame will also be inserted into the keyframe. k is 50 in our experiments.

Bundle Adjustment

Relying only on the tracking module generally leads to larger errors while the trajectory becomes longer. Meanwhile, the initial estimated SQ is not optimal due to the limited viewpoints. Therefore, Bundle Adjustment is used to optimise the camera pose and SQ parameters jointly. As shown in Fig. 5, each keyframe has corresponding SQs,

TABLE I

ABSOLUTE POSE ERROR OF FOUR SIMULATED SCENES AND THREE REAL DATASETS

APE(cm)	Simulation				Real Dataset		
	Scene1 (Nearly SQ shape)	**Scene2** (Mixed shape)	**Scene3** (Loop)	**Scene4** (Complex shape)	**Kitchen**	**Round Table**	**Dining Table**
Full Yolo/MobileSAM Mask	0.7398	2.3241	1.1804	5.4427	3.336	4.268	1.3309
No uncertainty Yolo/MobileSAM Mask	0.6822	4.6342	3.7548	7.4223	3.714	4.6025	1.3349
Full Perfect Mask	0.9684	1.8639	1.1466	5.1582	-	-	-
No uncertainty Perfect Mask	1.0436	4.5753	4.9989	6.1949	-	-	-

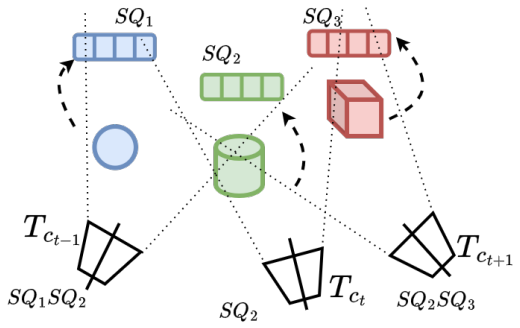

Fig. 5. Bundle Adjustment.

by minimising all the keyframes' residual together, as in (12), the optimised keyframe poses and corresponded SQ parameters are obtained.

$$\min_{T_c, T_{SQ}} \sum_{K} \sum_{u,v} \frac{(d[u,v] - D[u,v])^2}{d_{var}[u,v]} \tag{12}$$

where K represents the number of keyframe poses.

The local Bundle Adjustment is performed every time a new keyframe is added, to optimise the local map and the relative camera position. A sliding window method is applied to select keyframes. We fixed the camera poses for the first half of the window. We only optimised the camera poses for the later half and the visible SQs for all the keyframes in the window. Meanwhile, when the whole trajectory is complete, we also perform this global optimisation using all the keyframes and all the SQs but fixing the camera poses for the first frame to reduce the final bias.

V. EXPERIMENTS

Our system focuses on two missions: camera pose tracking and SQ reconstruction. Therefore, we designed two parts of experiments to evaluate the performance.

A. Camera Tracking

Our system is only based on objects to achieve the camera tracking. Therefore, intuitively, it is more reasonable to compare it with other works of objects orientated for camera positioning, e.g., NodeSLAM. However, it is hard for us to compare with it because their code and datasets are not open-source. Moreover, our system is based on minimizing the residual of the estimated depth and real depth; the ability of a SQ to resemble the object's shape will considerably affect the performance of the camera positioning. So, we choose to design similar scenes as NodeSLAM to evaluate the performance of our system.

We first employed the PyBullet[40] simulator to create four datasets, each based on a desktop scene and containing four to six objects. The four scenes consisted of three short trajectories (scene 1, 2, 4; trajectories are 4-6 meters long) and one trajectory containing a full loop (scene 3; trajectory is 11 meters long) to evaluate the loop's impact on the system. The object shapes vary from nearly perfect SQ shapes, such as books and tea boxes, to SQ-like shapes, such as apples, soaps, shampoo bottles, etc. We also experimented with more complex objects, such as glue bottles and cups. We used the virtual camera in the PyBullet to capture RGB and depth images and obtained camera ground truth trajectories from the PyBullet directly. Gaussian noises are added to the depth images and trajectories to guarantee that they simulate the real scene properly.

In addition, to evaluate the performance of the system in real-world scenarios, we also collected three datasets. Due to the difficulties of obtaining the ground truth trajectories, in these three scenarios, we use the trajectories from ORBSLAM3[11] as ground truth and compare our performance with it. It contains a kitchen scene with a straight trajectory, a long dining table scene with a round-trip trajectory, and a round table scene with a loop. Each dataset contains mugs, mice, books, bowls, keyboards, and other items. In particular, the kitchen scene includes microwave ovens and a larger tea box to verify the system's compatibility with different object scales.

The absolute pose error (APE) is adopted here to measure the camera tracking quality, and an ablation study is designed to analyse the effect of the accuracy of the mask (Yolo masks or MobileSAM masks compared with ground truth masks from PyBullet) and the impact of the uncertainty module on the system performance. The experimental results are shown in Table I. The nearly perfect SQ shapes result in the best tracking performance, with the APE error within 1 cm. With the scene becoming more complicated, the error

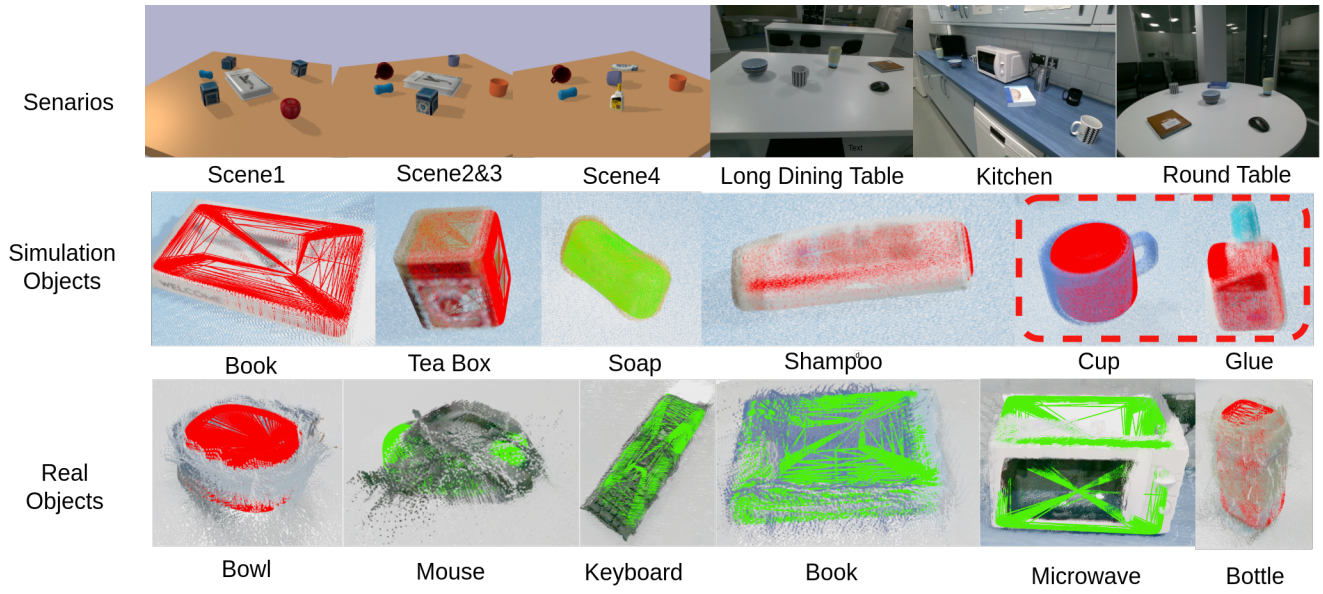

Fig. 6. Superquadric reconstruction results. The first row illustrates the scenario settings, including 3 simulation scenes and 3 real-world scenes. The second row contains the comparison of object point clouds(GT) and estimated SQs. The third row shows the reconstruction comparison of real data.

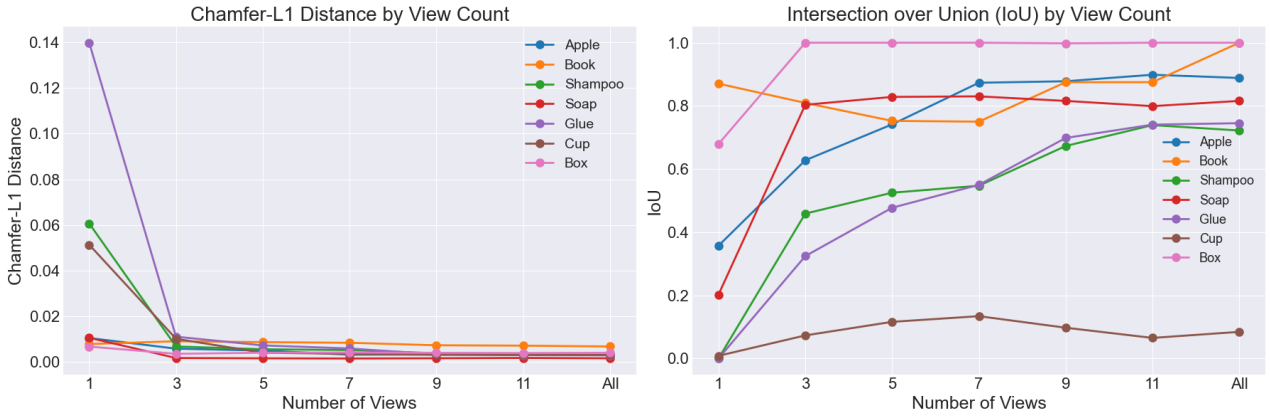

Fig. 7. Chamfer-L1 and 3D IoU comparison on apple, book, shampoo, soap, glue, cup, and box.

is relatively higher. Moreover, the better mask quality leads to less noise on the object's edge's depth, increasing the tracking accuracy. The uncertainty module influenced the optimisation with more precise results. It is worth noting that for scene 1, better mask accuracy and the application of the uncertainty module did not increase accuracy. This is because, for objects more similar to the shape of the SQ, the uncertainty module reduces the influence of the object edges information since the larger values of uncertainty are essentially concentrated at the edges of the SQ (as shown in Fig. 4). In cases where the shape of the object and the SQ are imperfectly similar, uncertainty reduces the effect of this imperfection. However, for the perfect shape of SQ, edge information is needed for further refined optimisation. It also was noteworthy that there was no significant discrepancy in the results of the realistic dataset compared to the simulation dataset (scene 2 and 3), which proved the reasonableness of the simulation dataset setup and also demonstrated that our system could also achieve reasonable quality for online camera tracking in certain scenarios of the real world.

B. Superquadric Reconstruction

This experiment was applied to both simulation and real environments, whose setup scene and reconstruction performance are summarised in Fig. 6. As displayed in the first row of images, the scenarios consist of indoor scenes with a tabletop and a variety of daily-used objects, including mugs, apples, books, and so on. The second and third rows of figures show the example of superquadric reconstruction in simulation and real scenarios, respectively. The generated reconstructed model and the real object can be accurately matched in terms of position and rotation. There is a slight error in size but it is within acceptable limits. It is also worth noting that for more complex objects, such as mugs with handles (highlighted by the dotted line), the reconstruction results show a certain segmentation effect. Specifically, the

TABLE II

COMPARISON OF CHAMFER-L1 DISTANCE AND IOU FOR DIFFERENT VIEWS AND OBJECTS INCLUDING RESULTS FROM ANOTHER PAPER

Object	View 1	View 3	View 9	View All	[12]
Chamfer-L1 Distance					
Apple	0.01054	0.00589	0.00332	0.00313	0.00067
Book	0.00790	0.00914	0.00737	**0.00468**	0.00586
Shampoo	0.06067	0.00684	0.00387	0.00314	0.00127
Soap	0.01065	0.00181	0.00169	0.00168	0.00052
Glue	0.13976	0.01113	0.00345	0.00298	0.00136
Cup	0.03517	0.00689	0.00349	0.00325	0.00227
Box	0.00680	0.00363	0.00407	0.00397	0.00073
IoU					
Apple	0.35714	0.62778	0.87793	0.88837	0.90972
Book	0.87030	0.80936	0.87500	**1.0**	**1.0**
Shampoo	0.0	0.45885	0.67299	**0.73910**	0.73585
Soap	0.20231	0.80341	0.81569	0.81571	0.85185
Glue	0.0	0.32436	0.69845	**0.74517**	0.68581
Cup	0.03124	0.08837	0.09528	**0.08229**	0.04736
Box	**0.67916**	**1.0**	**1.0**	**1.0**	**1.0**

mug body is reconstructed as a cylinder while the handle is separated from it, which leaves the potential to use multiple SQs to reconstruct complex-shaped objects in the future.

To quantitatively evaluate the performance of SQ reconstruction, Chamfer-L1 distance and 3D IoU were introduced as metrics. We also compare the results with [12], an algorithm for fitting SQs with the point cloud of the objects. Here, the input of [12] is the complete point cloud extracted from the object mesh model.

To ensure fairness and generalisability, several samples with different shapes and sizes were selected, including apple, book, shampoo, soap, glue, cup, and box, whose test results are summarised in Table. II and Fig. 7. At the very beginning of View 1, the initial value of Chamfer-L1 distance is at its maximum state, indicating a large error between the SQ reconstruction and the real object, and the corresponding IoU is also at a low level. With the view increasing, the Chamfer-L1 distance gradually decreases towards a lower level and IoU is improved to around 74% - 100%, which represents the SQ reconstruction error is significantly reduced until an acceptable level. As shown in Fig. 7, it is clear from the iterative trend that before View 3, the decrease in Chamfer-L1 distance and the increase in IoU are both rapid, while after View 3, the above trend slows down and the reconstruction quality reaches a more stable state. This proves that our system could achieve high-quality SQ reconstruction within limited views. Especially for the box and book, whose shape is nearly perfect SQ, the initial reconstruction is already 67%, and 87% and the final IoU reaches an incredible 100%. Compared with [12], our Chamfer-L1 results are relatively larger for almost all the classes, but the IoU results are better than [12]. This is mainly due to the principle optimisation idea differences: our optimisation is based on the occupancy value in 3D space benefitting the IoU results, while their optimisation is based on surface fitting, better with the Chamfer-L1 results. However, our system achieves fast initialisation with limited views of the objects and incrementally optimises SQs, while they directly require the full model of the objects to achieve the best results. On the other hand, our Chamfer-L1 is still at a remarkably low level, with final errors all under 4.7 mm.

Notably, both of the works got a high error with the cup class. This is due to the limitation of the SQ representation. For the cup shown in Fig. 6, whose internal body is empty, it will be wrongly covered by generated SQ reconstruction, leading to an IoU at a low level.

VI. CONCLUSIONS

In this paper, we develop a SLAM system that relies on lightweight SQ for map reconstruction and object-level tracking. We also design a bundle adjustment based on probabilistic rendering with SQs to optimise the camera and SQ maps simultaneously. The final results show that our system has a small error of less than 1 cm under ideal conditions. Even in a more complex real scenario, the error compared to ORBSLAM3 is about 3 cm. In addition, we also evaluate the reconstruction effect of SQ representation on real objects, and the final Chamfer-L1 error is only 1-4mm, and the 3D IoU can reach more than 74% for most objects shapes. However, since SQs are still abstract for complex shapes, our future work is to use multiple SQs to decompose such shapes and enhance the generalisation of the system.

REFERENCES

[1] A. P. Gee, D. Chekhlov, W. W. Mayol-Cuevas, and A. Calway, "Discovering planes and collapsing the state space in visual slam." in *BMVC*. Citeseer, 2007, pp. 1–10.

[2] J. M. Carranza and A. Calway, "Efficiently increasing map density in visual slam using planar features with adaptive measurement," in *British Machine Vision Conference*. British Machine Vision Association, 2009.

[3] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4605–4611.

[4] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 1352–1359.

[5] L. Nicholson, M. Milford, and N. Sünderhauf, "Quadricslam: Dual quadrics from object detections as landmarks in object-oriented slam," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 1–8, 2018.

[6] E. Sucar, K. Wada, and A. J. Davison, "Neural object descriptors for multi-view shape reconstruction," *CoRR*, vol. abs/2004.04485, 2020. [Online]. Available: https://arxiv.org/abs/2004.04485

[7] A. H. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer graphics and Applications*, vol. 1, no. 01, pp. 11–23, 1981.

[8] S. Alaniz, M. Mancini, and Z. Akata, "Iterative superquadric recomposition of 3d objects from multiple views," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 18 013–18 023.

[9] X. Han and L. Yang, "Sq-slam: Monocular semantic slam based on superquadric object representation," *Journal of Intelligent & Robotic Systems*, vol. 109, no. 2, p. 29, 2023.

[10] F. Tschopp, J. I. Nieto, R. Siegwart, and C. Cadena, "Superquadric object representation for optimization-based semantic SLAM," *CoRR*, vol. abs/2109.09627, 2021. [Online]. Available: https://arxiv.org/abs/2109.09627

[11] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[12] W. Liu, Y. Wu, S. Ruan, and G. S. Chirikjian, "Robust and accurate superquadric recovery: a probabilistic approach," *CoRR*, vol. abs/2111.14517, 2021. [Online]. Available: https://arxiv.org/abs/2111.14517

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[14] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[15] K. R. Beevers and W. H. Huang, "Slam with sparse sensing," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 2285–2290.

[16] P. Burger, B. Naujoks, and H.-J. Wuensche, "Map-aware slam with sparse map features," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 347–353.

[17] Y. Ling and S. Shen, "Building maps for autonomous navigation using sparse visual slam features," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1374–1381.

[18] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for rgb-d cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2100–2106.

[19] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. J. Davison, "Elasticfusion: Dense slam without a pose graph." in *Robotics: science and systems*, vol. 11. Rome, Italy, 2015, p. 3.

[20] Z. Yan, M. Ye, and L. Ren, "Dense visual slam with probabilistic surfel map," *IEEE transactions on visualization and computer graphics*, vol. 23, no. 11, pp. 2389–2398, 2017.

[21] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1697–1716, 2016.

[22] 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," *CoRR*, vol. abs/2003.08934, 2020. [Online]. Available: https://arxiv.org/abs/2003.08934

[23] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "isdf: Real-time neural signed distance fields for robot perception," in *Robotics: Science and Systems*, 2022.

[24] Z. Zhu, S. Peng, V. Larsson, W. Xu, H. Bao, Z. Cui, M. R. Oswald, and M. Pollefeys, "Nice-slam: Neural implicit scalable encoding for slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

[25] E. Sucar, S. Liu, J. Ortiz, and A. Davison, "iMAP: Implicit mapping and positioning in real-time," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[26] Z. Zhu, S. Peng, V. Larsson, Z. Cui, M. R. Oswald, A. Geiger, and M. Pollefeys, "Nicer-slam: Neural implicit scene encoding for rgb slam," *arXiv preprint arXiv:2302.03594*, 2023.

[27] A. Rosinol, J. J. Leonard, and L. Carlone, "Nerf-slam: Real-time dense monocular slam with neural radiance fields," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 3437–3444.

[28] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 925–938, 2019.

[29] Z. Liao, Y. Hu, J. Zhang, X. Qi, X. Zhang, and W. Wang, "So-slam: Semantic object slam with scale proportional and symmetrical texture constraints," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4008–4015, 2022.

[30] F. Tschopp, J. Nieto, R. Siegwart, and C. Cadena, "Superquadric object representation for optimization-based semantic slam," *arXiv preprint arXiv:2109.09627*, 2021.

[31] X. Han, H. Liu, Y. Ding, and L. Yang, "Ro-map: Real-time multi-object mapping with neural radiance fields," *IEEE Robotics and Automation Letters*, 2023.

[32] X. Kong, S. Liu, M. Taher, and A. J. Davison, "vmap: Vectorised object mapping for neural field slam," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 952–961.

[33] A. Leonardis, A. Jaklic, and F. Solina, "Superquadrics for segmenting and modeling range data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1289–1295, 1997.

[34] Y. Wu, W. Liu, S. Ruan, and G. S. Chirikjian, "Primitive-based shape abstraction via nonparametric bayesian inference," 2022. [Online]. Available: https://arxiv.org/abs/2203.14714

[35] D. Paschalidou, A. O. Ulusoy, and A. Geiger, "Superquadrics revisited: Learning 3d shape parsing beyond cuboids," 2019. [Online]. Available: https://arxiv.org/abs/1904.09970

[36] A. Jaklic, A. Leonardis, and F. Solina, *Segmentation and recovery of superquadrics*. Springer Science & Business Media, 2000, vol. 20.

[37] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics yolov8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[38] C. Zhang, D. Han, Y. Qiao, J. U. Kim, S.-H. Bae, S. Lee, and C. S. Hong, "Faster segment anything: Towards lightweight sam for mobile applications," *arXiv preprint arXiv:2306.14289*, 2023.

[39] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[40] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.