

TOWARDS AUTONOMOUS SEMANTIC PLACE COGNITION IN MOBILE
ROBOTS

by

Semanur Kumaş

B.S., Electrical and Electronics Engineering, Karadeniz Technical University, 2018

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2025

ACKNOWLEDGEMENTS

First and foremost, I extend my heartfelt thanks to my advisor, Prof. H. Işıl Bozma, for her invaluable guidance, constructive feedback, and unwavering support during the research process. Her expertise and encouragement have been instrumental in shaping this work.

I am also profoundly grateful to the members of my thesis committee, Prof. Yağmur Denizhan and Assist. Prof. Özgür Erkent, for their critical input and valuable contributions, which improved the quality of this thesis.

I sincerely thank my colleagues Serhat İşcan, Arif Yılmaz, S. Batuhan Vatan, Yunus As, Yavuzhan Erdem, and H. Ziya Zorluoğlu at Intelligent Systems Laboratory for their collaboration, insightful discussions, and the sense of camaraderie that made this journey both enriching and unforgettable. I am particularly grateful for their invaluable support and for fostering such a welcoming and encouraging environment.

I would also like to express my special thanks to A. Harun Eker for his endless support, constructive criticism, and invaluable brainstorming sessions.

On a personal note, I owe an immense debt of gratitude to my parents, Ayşe Kumaş and Dursun Kumaş, my dear sister, Kübra Kumaş, and my friends for their unwavering love, patience, and encouragement. Their belief in me has been a constant source of motivation, especially during challenging times.

ABSTRACT

TOWARDS AUTONOMOUS SEMANTIC PLACE COGNITION IN MOBILE ROBOTS

This thesis is concerned with the problem of autonomous semantic place cognition in mobile robots. The importance of this problem is due to the growing demand of using robots for service tasks. This requires the robot to construct a semantic place map, retain this model in its memory and use it as necessary. A semantic map is defined by a labeled graph with nodes corresponding to objects and edges encoding their relations. This thesis addresses this problem through introducing a complete framework for building and using semantic place maps. In the proposed framework, differing from previous work, objects are represented as point cloud objects that are detected through tracking the candidate and thus class-independent point cloud objects over the incoming RGB-D stream. The tracking is done through considering features such as position, shape, color, keypoints and neighboring objects. As the robot is tracking the point cloud objects, it attempts to recognize the respective objects through object recognition. These objects are then used to incrementally build the semantic place map. These maps can then be used in the robot's subsequent visits of the associated place. Experimental evaluation indicates the recognition performance of the semantic place map to depend on both object recognition and routes traversed during first-time visits of places.

ÖZET

GEZGİN ROBOTLARIN OTONOM ANLAMSAL MEKAN BİLİŞİNE SAHİP OLABİLMESİ

Bu tez, gezgin robotlarda otonom anlamsal mekan algılama problemini ele almaktadır. Bu, robotun bir anlamsal mekan haritasını oluşturmamasını, bu modeli hafızasında tutmasını ve görev süresince gerektiğinde bu haritayı kullanmasını gerektirir. Bu problemin önemi, robotların hizmet görevlerinde kullanımına yönelik artan talepten kaynaklanmaktadır. Bu tez, bu sorunu kapsamlı bir yapı sunarak ele almaktadır. Önerilen yapı, nokta bulutu nesne adaylarının tespit edilmesini, robotun hareketleri boyunca bu nesnelerin izlenerek mevcut nesnelerin ve onların geometrik ilişkilerinin belirlenmesini, gerektiğinde bu nesnelerin tanınmasını veya öğrenilmesini, buna göre anlamsal mekan haritalarının oluşturulmasını, bu modellerin anlamsal mekan hafızasında saklanmasını ve aynı sahneye gelecekteki ziyaretlerde bu haritalara başvurulmasını içermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	viii
LIST OF TABLES	xi
LIST OF SYMBOLS	xiii
LIST OF ACRONYMS/ABBREVIATIONS	xvi
1. INTRODUCTION	1
1.1. Approach Summary	2
1.2. Contributions	3
1.3. Outline	3
2. CANDIDATE POINT CLOUD OBJECTS	4
2.1. Related Literature	5
2.2. Discovery of Candidate Point Cloud Objects	6
2.3. RGB Segmentation	6
2.4. Refining RGB Segments	7
2.5. Candidate Point Cloud Objects	8
3. DETECTING POINT CLOUD OBJECTS THROUGH TRACKING	10
3.1. Related Literature	10
3.2. Candidate Point Cloud Object States	12
3.2.1. Shape & Position	12
3.2.2. Color	14
3.2.3. RGB Features	14
3.2.4. Neighboring Objects' Relations	15
3.3. Pairwise Similarity of Point Cloud Object Candidates	17
3.4. Multi-Object Tracking	20
3.5. Object Cognition	22
3.6. Experimental Results	24

3.6.1. Datasets	24
3.6.1.1. Isaac Sim Dataset	24
3.6.1.2. Real-World Dataset	25
3.6.2. Performance Evaluation	26
3.6.3. Experiments on Isaac Sim Dataset	29
3.6.4. Experiments on Real-World Dataset	31
4. SEMANTIC PLACE MAP	36
4.1. Related Literature	36
4.2. Semantic Place Maps - Formulation	37
4.3. Semantic Place Maps - Incremental Construction	37
4.4. Semantic Place Maps - Recognition	38
4.5. Experimental Results	39
4.5.1. SPM Construction - First Visits	40
4.5.2. SPM-Based Place Recognition	48
4.5.3. SPM-Based Place Recognition - Subsequent Visits	50
5. CONCLUSION	62
REFERENCES	64
APPENDIX A: ELLIPSOID	70
A.1. Ellipsoid Fitting: Khachiyan's Algorithm	70
A.2. Ellipsoid Fitting: PCA	71
A.3. Ellipsoid Visualization	73
APPENDIX B: USER GUIDE	74

LIST OF FIGURES

Figure 2.1.	RGB segmentation.	7
Figure 2.2.	Candidate objects after refining.	8
Figure 2.3.	Refining candidate point cloud objects.	9
Figure 3.1.	Ellipsoid representation of a point cloud object.	13
Figure 3.2.	Object image and its Hue histogram.	14
Figure 3.3.	Local features state defined by keypoints.	15
Figure 3.4.	Finding neighbors.	17
Figure 3.5.	Sample frames in ISAAC dataset	25
Figure 3.6.	Sample frames from real-world dataset.	25
Figure 3.7.	Computation of tracking performance metrics.	28
Figure 3.8.	Segments of a sample Isaac Sim frame	29
Figure 4.1.	Flowchart of incremental graph construction.	38
Figure 4.2.	Isaac Sim SPM Experiments: Place 1.	41
Figure 4.3.	Place 1 - Incremental construction of G_1	42

Figure 4.4.	Isaac Sim SPM Experiments: Place 2.	43
Figure 4.5.	Place 2 - Incremental construction of G_2	44
Figure 4.6.	Isaac Sim SPM Experiments: Place 3.	45
Figure 4.7.	Place 3 - Incremental construction of G_3	45
Figure 4.8.	Isaac Sim SPM Experiments: Place 4.	46
Figure 4.9.	Place 4 - Incremental construction of G_4	47
Figure 4.10.	Revisiting place 1 with a route similar to that of SPM G_1^*	51
Figure 4.11.	Place 1 revisit - Incremental construction of SPM G along a route similar to that of G_1^*	52
Figure 4.12.	Revisiting place 1 with a different route.	53
Figure 4.13.	Place 1 revisit - Incremental construction of SPM G	54
Figure 4.14.	Revisiting place 2 with different a route.	55
Figure 4.15.	Place 2 revisit - Incremental construction of SPM G	56
Figure 4.16.	Revisiting place 3 with different a route.	57
Figure 4.17.	Place 3 revisit - Incremental construction of SPM G	58
Figure 4.18.	Revisiting place 4 with different a route.	59

Figure 4.19. Place 4 revisit - Incremental construction of SPM G	60
--	----



LIST OF TABLES

Table 3.1.	Parameter values used for tracking.	28
Table 3.2.	Tracking algorithm performance in Isaac Sim dataset with FastSAM segments.	30
Table 3.3.	Tracking performance in Isaac Sim dataset with Isaac Sim segments.	30
Table 3.4.	Tracking performance with different algorithms and scanned back to a certain frame.	32
Table 3.5.	Tracking performance with different algorithms and scanned back to the beginning of WCOM.	32
Table 4.1.	Parameter values for scene recognition.	40
Table 4.2.	Recognition performance of SPMs.	48
Table 4.3.	Evolving G_1 similarity.	49
Table 4.4.	Evolving G_2 similarity.	49
Table 4.5.	Evolving G_3 similarity.	49
Table 4.6.	Evolving G_4 similarity.	50
Table 4.7.	Evolving SPM G similarity in the second visit of place 1.	51
Table 4.8.	Evolving SPM G similarity in revisiting place 1.	55

Table 4.9.	Evolving SPM G similarity in revisiting place 2.	57
Table 4.10.	Evolving SPM G similarity in place 3.	59
Table 4.11.	Evolving SPM G similarity in place 4.	61
Table 4.12.	Recognition performance of revisit SPMs with different routes. . .	61



LIST OF SYMBOLS

A_O	Shape matrix of candidate point cloud object
c_1, c_2	RGB camera optic center
C_O	Centroid of candidate point cloud object
$d(u)$	Depth value associated with pixel u
$\mathcal{D}(G, G')$	A path for transforming G into G'
$E_O = (A_O, C_O)$	Ellipsoid of candidate point cloud object
f	Focal length
F_{pk}	k^{th} frame in place p
G_O	Object neighbor graph
$G_p(k)$	Semantic place map of place p at frame k
H_O	Hue histogram
$\bar{H}_O$	Mean value of hue histogram H_O
k	Discrete time index
$K_{O,O'}(s)$	Ellipsoid intersection function
$\mathcal{K}_p$	Index set of frames in place p
$\mathcal{K}_Q$	Index set of the frames in which the object is detected.
$\mathcal{N}(O)$	Neighborhood set of object O
N_h	Number of histogram bins
N_O	Minimum track cardinality
N_p	Cardinality of $\mathcal{K}_p$
O	Point cloud object
$\mathcal{O}_p$	Point cloud objects of place p
$\mathcal{O}_{pk}$	Candidate point cloud objects of place p from frame k
$\mathcal{O}_p(k)$	Labeled point cloud objects in $G_p(k)$
$\mathcal{O}_{pk}^*$	Refined candidate point cloud objects of place p from frame k
p	Place index
$P(u)$	Map from pixel coordinates u to object coordinates

$\mathcal{P}$	Index set of places
Q	Object index
$\mathcal{R}_p(k)$	Proximity relations in $G_p(k)$
$\mathcal{S}_{pk}$	RGB segments of frame F_{pk}
$\mathcal{S}_{pk}^*$	Refined RGB segments of frame F_{pk}
T^Q	Track of point cloud object O
$\mathcal{T}_{pk}$	Set of point cloud object tracks in place p at time k
w_i	i^{th} feature weight
$\mathcal{W}(G, G')$	Set of possible paths for transforming G into G'
x_O	State of object candidate O
$\mathcal{X}$	State space of features
γ	Dissimilarity of states
γ_i	Dissimilarity of states defined by i^{th} features
Δk	Time window index
$\delta(O, O')$	Distance between O and O'
δ_p	Pixel size
$\eta(O, O')$	Label of edge between O and O'
λ_d	Node degree comparison cost
λ_e	Edge insertion/deletion cost
λ_{es}	Edge substitution cost
λ_l	Eigenvalues of the ellipsoid shape matrix
λ_n	Node insertion/deletion cost
λ_{ns}	Node substitution cost
$\xi(e)$	Cost of edit operation e
ρ_o	Radius of averagely fitting sphere
τ_d	Data association threshold
τ_g	SPM recognition threshold
τ_l	Greatest lower bound (percentage)
τ_m	RGB segments overlap factor

τ_n	Upper bound for proximity of two candidate point cloud objects
τ_O	Minimum track cardinality
τ_r	Object recognition threshold
τ_s	Sphere distance threshold
τ_u	Least upper bound (percentage)
$\phi(G, G')$	Graph edit distance between G and G'
ψ	Segment confidence score



LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
COCO	Common Objects in Context
FN	False Negative
FP	False Positive
GED	Graph Edit Distance
ID	Identification
LTOM	Long Term Objects' Memory
MOT	Multi Object Tracking
MVEE	Minimum Volume Encapsulating Ellipsoid
ORB	Orientated Fast and Robust Brief
PCA	Principal Components Analysis
PCL	Point Cloud Library
RGB	Red Greed Blue
RGB-D	Red Greed Blue Depth
SIFT	Scale Invariant Feature Transform
SOR	Statistical Outlier Removal
SORT	Simple Online and Realtime Tracking
SPM	Semantic Place Map
SURF	Speeded-Up Robust Features
TN	True Negative
TP	True Positive
WCOM	Working Candidate Objects Memory

1. INTRODUCTION

Semantic place cognition is a critical ability for mobile robots as it enables the robot to understand the contents of its surroundings and use this knowledge in tasks requiring spatial and semantic reasoning throughout its operation lifetime. It is a complex ability as it requires the robot to build semantic place maps, store them, and refer to them in future visits and or tasks as necessary. A semantic place map is an internal representation that incorporates the major constituent objects and their spatial relations within each place [1]. Such a map would be intuitively understandable by humans and thus would alleviate its interaction with its surroundings including humans. While there has been tremendous progress in the construction of semantic maps, it is still far from being solved - especially in cluttered settings such as households or workplaces where humans and robots interact the most. If autonomy is desired, the problem is even further exacerbated.

The discovery of all objects in a place is viewed as being the first step in building semantic place maps. The proposed approaches vary depending on the type of visual data (RGB only, depth only or RGB-D) and the number of frames (single or multiple) used in determining them. Most approaches are observed to be single RGB image based. In order to cover the scene, the robot needs to repeat the method for each different image and then integrate them. An alternative approach has been to use a continuous video input - instead of a single image. Its advantages have been shown to be two-fold: First, the reliability of object candidates is shown to improve when using multiple views [2, 3] or the temporal coherence of the visual stream [4, 5]. Second, as a video input can cover the scene comprehensively, the resulting semantic place map can be more comprehensive.

More recently, with the advance of three-dimensional (3D) sensing technology, objects are increasingly represented as point cloud objects. However, the majority of current 3D object recognition approaches follow the single frame detection paradigm

while few of them consider the incoming point cloud stream (video) as defined by a temporal sequence of point cloud frames. However, for example, using only a single RGB-D frame may suffer from ambiguities due to its nature or viewing geometry. On the other hand, the spatiotemporal coherency of data can help to resolve single-frame decision ambiguities [6–8].

1.1. Approach Summary

The thesis proposes a complete framework for building and using semantic place maps. Here, a ‘place’ refers to a specific area or region [9] - similar to human’s concept of a ‘place’. As such, it differs from many works in which each location is thus viewed as a distinct place - as it has a different appearance. In the proposed approach, a place is defined by the complete set of RGB and depth frames captured as the robot moves therein. This requires objects to be identified from individual scene frames as well as integrating the objects detected across all the frames into a single entity.

The proposed framework incorporates the following parts:

- (i) Discovery of candidate point cloud objects;
- (ii) Tracking them as the robot looks around and/or moves around in the place;
- (iii) Constructing a semantic place map of the place accordingly and referring to these maps in future visits of the same place.

This enables the robot to use this knowledge in a range of tasks requiring spatial and semantic understanding such as scene recognition verification or determining the most likely locations of specific objects.

1.2. Contributions

The contributions of this thesis can be summarized as follows:

- A novel method to determine the proximity relations among point cloud objects based on their modeling as an ellipsoid.
- A novel approach for the class-independent detection of point cloud objects in a place that is based on tracking candidate point cloud objects in the incoming RGB-D stream using a feature-based approach. Hence, as it is class-agnostic, the proposed approach can be used to map new places with possibly unknown objects.
- A novel approach for the construction of semantic place maps based on point cloud objects and their proximity relations and their usage in tasks requiring place recognition

1.3. Outline

This thesis is structured as follows:

- Chapter 2 introduces the proposed object model.
- Chapter 3 details a novel object tracking framework, including short-term memory models.
- Chapter 4 focuses on the development of a semantic place model and work on place recognition.
- Finally, Chapter 5 discusses the findings and contributions of this thesis.
- Additionally, Appendix A provides detailed information on ellipsoid fitting to point clouds.

2. CANDIDATE POINT CLOUD OBJECTS

Object candidates refer to class-independent entities that can be inferred from the incoming visual data without any prior knowledge pertaining to their appearances or categories [10, 11]. They are primarily defined by contiguous regions in the data with perceptually similar content. Thus, they are thought to constitute fundamental units for ensuing higher-level visual data analysis and reasoning. This approach has been in part motivated by findings in vision science that suggest object candidates are obtained from early segmentation processes [12]. In these communities, they are equivalently referred to as proto-objects. As such, object discovery has also been considered within the cognitive psychology communities and the computer vision and ‘proto-object perception’ and ‘object proposal generation’ respectively [13].

This is an important problem as these candidates serve as the basis for the robot to determine the objects in its surroundings which it can then use in tasks requiring spatial and semantic reasoning. Interestingly, even though it is simpler than object detection, object discovery is still a non-trivial problem - due to the vast appearance and size variations amongst the objects as well as confusing backgrounds such as common work or household settings. Note that the objects of interest within a scene may vary depending on the scope or purpose of the analysis. For instance, in an outdoor scene, smaller objects like stones or leaves, which may not significantly contribute to describing the scene, might be excluded, whereas, in an indoor scene, some objects that are common in most indoor scenes may not be of interest.

Interestingly, there is no general consensus in regards to what constitutes candidate objects. In most work, these are primarily defined by two-dimensional segments that encode perceptually similar data in the RGB data [14]. More recently, point cloud objects are increasingly being considered - as they can encode both shape and color information.

2.1. Related Literature

Most approaches to finding objects work with 2D RGB images and generate object candidates through image segmentation [11]. This is an extensively studied area in computer vision with approaches varying from classic algorithms to more recent learned network based approaches. Among the latter, FastSAM has become one of the most popular 2D image segmentation algorithms [15]. It is built upon the YOLOv8-seg model, which is designed to detect object candidates with training on the Common Objects in Context (COCO) dataset [16]. The proposed algorithms also differ depending on whether the segments are labeled or not in which the labels vary depending on whether they refer to class labels, instance labels, or part labels. For example, FastSam can be used as a semantic segmentation algorithm - as it can provide class labels as well. In this case, these algorithms become also object detectors.

More recently, segmentation has been considered using 3D data. As 3D data contain richer geometric, shape, and scale information with less background noise, 3D segmentation typically yields a more comprehensive understanding of a scene in comparison to conventional single-view 2D segmentation. Recently, deep-learning techniques have dominated many research areas, including computer vision and natural language processing. Motivated by the success of deep learning methods in 2D segmentation, deep learning for 3D segmentation has also attracted growing interest from the research community. For example, as the depth map in an RGB-D image contains geometric information about the real world, it enables distinguishing foreground objects from the background. Typical approaches consider a two-channel (color and depth) framework and introduce various modules to incorporate contextual and geometric information. However, there still remains many unsolved challenges. For example, local features are hard to exploit due to the irregularity of point cloud data. While this may be addressed through converting them to high-resolution voxels, it is not feasible in most real-time applications due to its computational complexity [17].

2.2. Discovery of Candidate Point Cloud Objects

Consider a place as indexed by $p \in \mathcal{P}$ with $\mathcal{P}$ denoting the place index set. Suppose the robot collects a set of RGB-D frames in this place as defined by the index set $\mathcal{K}_p$ with cardinality N_p . Let k^{th} RGB-D frame taken by the robot in the place p be denoted by F_{pk} .

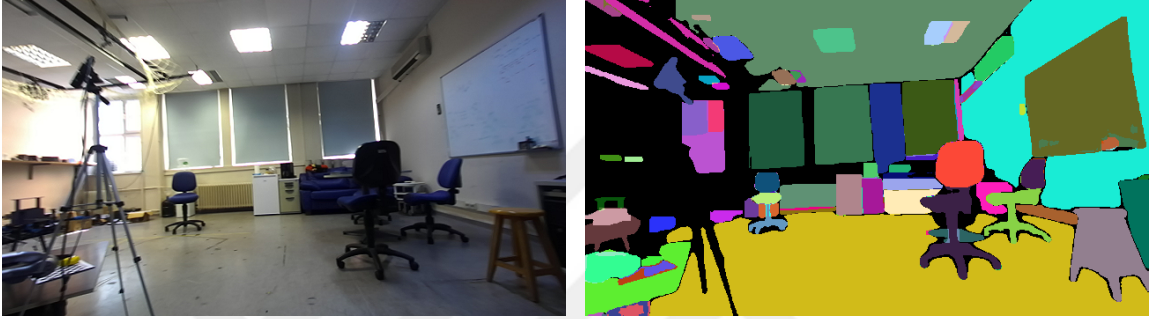
The proposed approach consists of the following stages:

- The initial step is to determine all the segments $\mathcal{S}_{pk}$ considering the frame F_{pk} .
- The second step is to process $\mathcal{S}_{pk}$ as to obtain a refined set of segments $\mathcal{S}_{pk}^*$ with excessive or erroneous detections removed.
- The final step is to construct candidate point cloud objects $\mathcal{O}_{pk}^*$ associated with frame F_{pk} using the candidate object masks obtained from the refined segments.

2.3. RGB Segmentation

The first step is to segment the incoming RGB-D image associated with frame F_{pk} into segments $\mathcal{S}_{pk}$. For this, a state-of-art RGB segmentation network such as FastSAM is used. Leveraging its pretrained model, FastSAM segments a given RGB frame into meaningful regions, which are considered object candidates in this work. Since it is based on a trained model on predefined object classes, some objects within a frame—and consequently the place may possibly not belong to any learned classes. These objects, despite not being associated with predefined classes, can still be crucial for describing the scene. FastSAM addresses this limitation by providing flexibility through its confidence score $\psi : \mathcal{S}_{pk} \rightarrow \mathbb{R}$ that encodes the similarity of a segment to its assigned class. By setting a low minimum confidence threshold, FastSAM eliminates the necessity for a segment to strongly belong to a specific class while extracting segments, i.e., identifying object candidates. It is designed to leverage objects with low similarity to predefined classes that do not actually belong to any predefined class, thereby enabling the detection of objects outside the learned categories. The resulting

segments can account nearly for all the objects in the RGB data. For example, given the RGB data shown in Figure 2.1(a), there is a whiteboard and an air conditioner. When observing the resulting object candidates in Figure 2.1(b), it can be seen that, although there are no "whiteboard" or "air conditioner" classes in the COCO dataset, they are successfully identified as segments.



(a) Input RGB frame.

(b) FastSam segments.

Figure 2.1. RGB segmentation example: (a) input RGB frame and (b) segments of input frame.

2.4. Refining RGB Segments

The next step involves refining the RGB segments $\mathcal{S}_{pk}$ as to obtain a refined set of segments $\mathcal{S}_{pk}^*$ as to remove excessive or erroneous detections that result from having FastSAM run with a lower confidence score threshold. For the former, it may be preferable to focus only on objects with specific occupancy properties such as selecting only reasonably sized object candidates rather than too large or too small sized ones. As an example of the latter case, both an object (i.e. a cabinet) and its parts (i.e. its doors) may be detected as separate segments. To address these two cases, the filtering process is designed in two distinct stages:

- Filtering according to the size of the segments: For all segments $S \in \mathcal{S}_{pk}$, the refined set of segments is defined as

$$\mathcal{S}_{pk}^* = \begin{cases} \mathcal{S}_{pk} - \{S\} & \text{if } |S| > |F_{pk}| * \tau_u \\ \mathcal{S}_{pk} - \{S\} & \text{if } |S| < |F_{pk}| * \tau_l \end{cases} \quad (2.1)$$

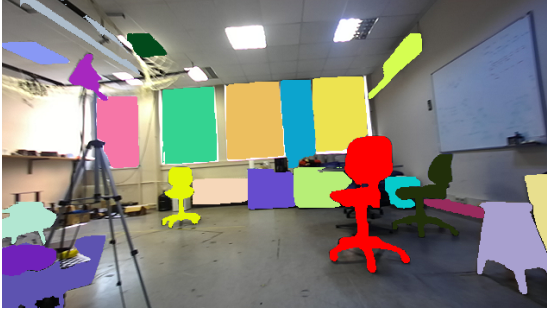
where τ_u and τ_l represent the least upper and greatest lower bounds, respectively, expressed as percentages.

- Filtering according to the amount of the intersection: For any pair $S, S' \in \mathcal{S}_{pk}$ such that $S \cap S' \neq \emptyset$, the refined set of segments is defined as

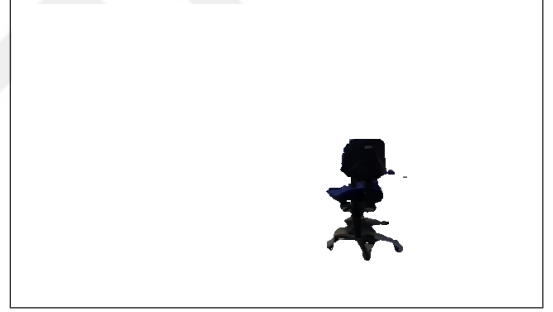
$$\mathcal{S}_{pk}^* = \begin{cases} \mathcal{S}_{pk} - \{S'\} & \text{if } |S| > |S'| \text{ and } |S \cap S'| \geq \tau_m |S'| \\ \mathcal{S}_{pk} - \{S\} & \text{if } |S'| > |S| \text{ and } |S \cap S'| \geq \tau_m |S| \end{cases} \quad (2.2)$$

where τ_m is the predetermined intersection factor.

The frame object candidates after the filtering process are given in Figure 2.2(a). It should be noted that in comparison to Figure 2.1(b), chairs are now detected as single entities, and its RGB image is given in Figure 2.2(b)



(a) Refined segments.



(b) Sample object candidate image.

Figure 2.2. (a)Candidate objects are obtained after refining segments and (b) refined sample object candidate image.

2.5. Candidate Point Cloud Objects

Candidate point cloud objects are determined based on the masks obtained from the refined segments $\mathcal{S}_{pk}^*$ and the depth data of frame F_{pk} . Let $\mathcal{O}_{pk}$ refer to the set of point cloud objects associated with frame k at place p . For each segment $S \in \mathcal{S}_{pk}^*$, the corresponding candidate point cloud object O is defined as

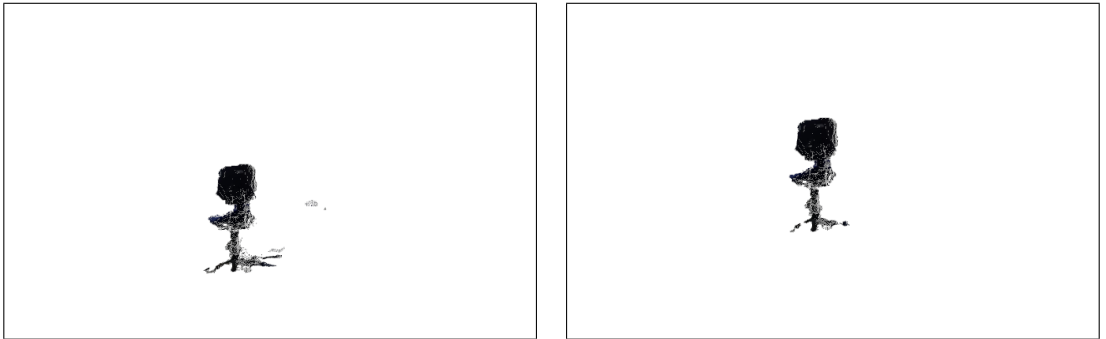
$$O = \{P(u) \mid u \in S\} \quad (2.3)$$

where $P : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ is the map from pixel coordinates to the point cloud object coordinates and defined as

$$P(u) = \begin{bmatrix} \frac{\delta_p(u_1 - c_1)X_3}{f} \\ \frac{\delta_p(u_2 - c_2)X_3}{f} \\ X_3 \end{bmatrix} \quad \text{where} \quad X_3 = \frac{d(u)}{\sqrt{\left(\frac{\delta_p(u_1 - c_1)}{f}\right)^2 + \left(\frac{\delta_p(u_2 - c_2)}{f}\right)^2 + 1}}. \quad (2.4)$$

Here $d(u)$ is the depth value associated with pixel $u = [u_1, u_2]^T$, the parameters (c_1, c_2) refer to the RGB camera optic center, f refers to its focal length, and δ_p is the pixel size. An example candidate point cloud is given in Figure 2.3(a).

Due to distortions introduced by both FastSAM segments and depth data, the resulting point cloud object may contain noisy data. These data are typically viewed as outliers that need to be removed. Among the various available algorithms, the Statistical Outlier Removal (SOR) [18] filter is utilized for this purpose using the Point Cloud Library (PCL) [19]. Assuming that these distances follow a Gaussian distribution characterized by a mean and standard deviation, outliers are detected by considering the mean distance between each point and its neighbors in the point cloud object. Finally, refined candidate point cloud objects $\mathcal{O}_{pk}^*$ are obtained via subjecting each object $O \in \mathcal{O}_{pk}$ to the SOR method. A sample case showing the effect of this refinement can be seen in Figure 2.3(b).



(a) Initial point cloud object.

(b) Refined point cloud object.

Figure 2.3. (a) Initial point cloud object with outliers and (b) refined candidate point cloud objects.

3. DETECTING POINT CLOUD OBJECTS THROUGH TRACKING

Point cloud objects are determined by considering the incoming point cloud stream (video) as defined by a temporal sequence of point cloud frames. This approach differs from most work in two aspects:

- First, the detected point cloud objects are class-agnostic entities. Hence, they differ from most work in which this is not the case. Hence, the proposed approach can be used to handle unknown objects.
- Second, the detection of each class-agnostic object is based on multiple frames. This differs from most work in which decisions are based on a single frame. Its advantages have been shown to be two-fold: First, the reliability of object candidates is shown to improve when using multiple views [3] or the temporal coherence of the visual stream [4]. Second, scene coverage will increase accordingly. Furthermore, the decision-making can then be triggered only for objects that have not yet been classified which would lead to the computational burden being reduced considerably.

The tracking of point cloud objects is achieved by matching the detected point cloud objects in each frame to those previously detected. This is based on the similarity of computed features. Object matching involves calculating the dissimilarity between object features, combining these values using a dissimilarity function, and comparing the result to a threshold to determine matches.

3.1. Related Literature

There has been extensive work on multi-object tracking (MOT). In algorithmic approaches, tracking is based on the similarity of objects in consecutive frames. The similarity is measured by matching object candidates of consecutive frames - con-

sidering object features such as their positions, shapes, and/or other image-derived properties. Matching is typically based on assignment algorithms from combinatorial optimization such as the Hungarian algorithm [20]. For example, Simple Online and Realtime Tracking (SORT) is a simple real-time tracking algorithm that uses Kalman filtering and the Hungarian algorithm to match objects in consecutive frames [21]. However, while using the Kalman filter may be good for the prediction of dynamic objects, it may not find the position of a static object if it leaves and re-enters the robot’s field of view.

More recent work has focused on the tracking-by-detection strategy. Here, a plethora of category detectors are used to form trajectories of the target objects. In order to resolve object detection ambiguities and to minimize detection failures, most process video sequences in a batch mode [22–24]. Hence, video frames from future time steps are also utilized to solve the data association problem which implies that these non-causal approaches are not suitable for real-time applications. For real-time tracking-by-detection strategy, the major challenge is the association of noisy object detections in the current video frame with previously tracked objects. One approach to address this problem has been to consider learning in MOT. The proposed learning schemes are primarily supervised approaches that work on RGB data and can be categorized into two groups depending on whether learning is offline or online [25]. In the former, learning is performed prior to the actual tracking such as learning the similarity function from ground truth trajectories. [26,27]. Consequently, they cannot handle the dynamic nature and history of the targets in data association which is required in order to resolve ambiguities such as re-assigning missed or occluded objects when they appear again. Alternatively, online approaches enable learning during tracking by using the data generated during tracking [28]. However, they need the data to be annotated which is typically not available. DeepSORT [29] was created to improve the performance of the SORT algorithm by integrating the appearances of the objects. FairMOT uses object detection and re-identification processes in a single network [30]. It also uses anchor-free object detection architecture. By doing this, they have obtained computation efficiency and high accuracy. In [31], a tracking algorithm has

been developed as to compensate for the low accuracy of the Siamese network-based tracking algorithm. They have employed ResNet network and utilized layer-wise and depth-wise aggregations to extract the object features.

Most of these works report experiment results on outdoor benchmark datasets. Hence, their performances in indoor scenes are not known. Also, most of them use learning for object extraction. However, to describe a place, all objects need to be considered not just learned ones since any object may help to recognize a place. For this purpose, [32] has developed a tracking algorithm that operates independently of the object extraction algorithm in an object feature-based manner. The tracking algorithm uses the centers of the objects, mean color, and object size, and looks at the Manhattan distance between the objects. [33] has developed a tracking algorithm based on the Kalman filter for position and shape feature dissimilarity of an object without using learning-based methods and the greedy data association algorithm. However, position and shape features may not be enough for the reason of Kalman filtering-based position disadvantage for static objects and shape may fall behind if two objects have the same shape.

3.2. Candidate Point Cloud Object States

Recall that $\mathcal{O}_{pk}^*$ refers to the set of point cloud objects associated with frame k at place p . Each object candidate $O \in \mathcal{O}_{pk}^*$ is associated with a state $x_O \in \mathcal{X}$ where $\mathcal{X}$ denotes the state space as defined by five different features: position and shape, color, RGB keypoints, and object neighborhood.

3.2.1. Shape & Position

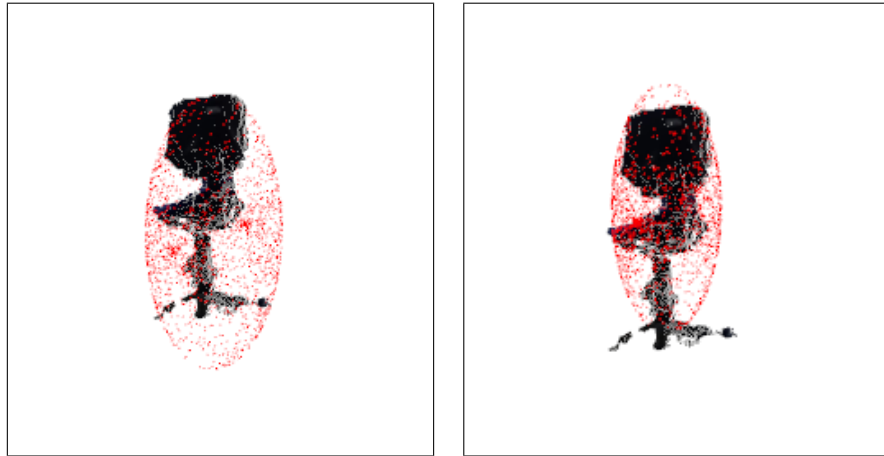
The first two components of the state x_O of an object candidate O encode its position and shape. The position is the key feature for object tracking. A significant portion of tracking algorithms only uses the position information of the objects for tracking. However, it may be misleading if an object candidate is not seen for a while

then it appears again. For example, suppose an object candidate o is seen at frame k , then it disappears for a while and then it appears at frame $k' \gg k$ after a while. When the object candidate is searched across those of previous frames, the difference between the position of the last appearance at frame k and the position of the appearance at frame k' will be very high and hence may be misleading for matching. Therefore, relying solely on position information is insufficient. Still, positional information of an object in consecutive frames provides valuable data.

The position and shape states are obtained from the minimum volume enclosing ellipsoid (MVEE) as defined by the normalized quadratic form,

$$(P - C_o)^T A_o (P - C_o) \leq 1. \quad (3.1)$$

Here, $C_o \in \mathbb{R}^3$ refers to the ellipsoid center and A_o is a symmetric matrix with positive eigenvalues that encode shape information. The position of a candidate object is taken directly as C_o , while the shape parameters are derived from the shape matrix A_o . C_o and A_o are found using two different methods: Khachiyan's Algorithm as detailed in Appendix A.1, and based on Principal Components Analysis (PCA) as explained in Appendix A.2. The resulting ellipsoids for a sample point cloud object are given in Figure 3.1(a) and in Figure 3.1(b) respectively.



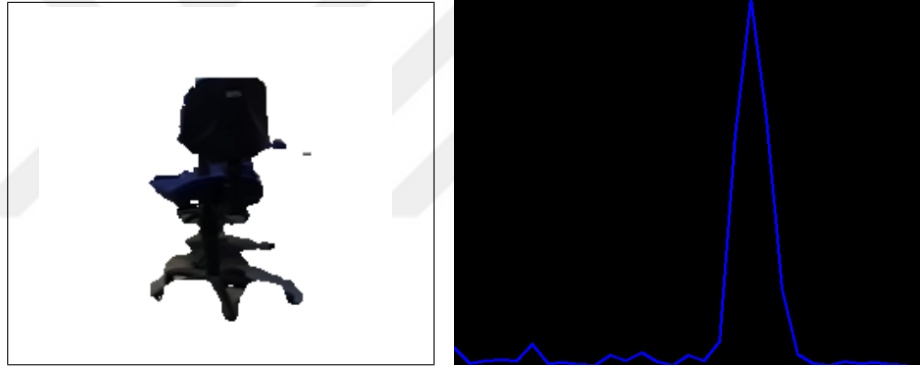
(a) Khachiyan algorithm.

(b) PCA algorithm.

Figure 3.1. Ellipsoid representation (a) obtained from Khachiyan and (b) obtained from PCA.

3.2.2. Color

The third component of the state x_O of an object candidate O encodes its color. While the color of an object is not always unique, it can be crucial for differentiation when two object candidates share the same shape. Here, the color data is obtained from the RGB data associated with the object candidate. In particular, after transforming the RGB color associated with the corresponding segment, the histogram of the hue channel is computed. Hence, the color state is defined by the resulting hue histogram. The hue histogram consists of 30 bins. An example object candidate image and its Hue histogram are shown in Figure 3.2(a) and in Figure 3.2(b) respectively.



(a) Candidate point cloud object. (b) Color state defined by hue histogram.

Figure 3.2. Color state of a point cloud object candidate (a) sample candidate point cloud object and (b) Hue histogram of sample candidate point cloud object.

3.2.3. RGB Features

The fourth component of the state x_O of an object candidate O encodes local features of the corresponding RGB segment. For this, keypoints, as defined by Scale Invariant Feature Transform (SIFT), Orientated Fast and Robust Brief (ORB), or Speeded-Up Robust Features (SURF), are considered. In particular, SURF keypoints are used. The keypoints are shown in Figure 3.3 for the given object in Figure 3.2(a).

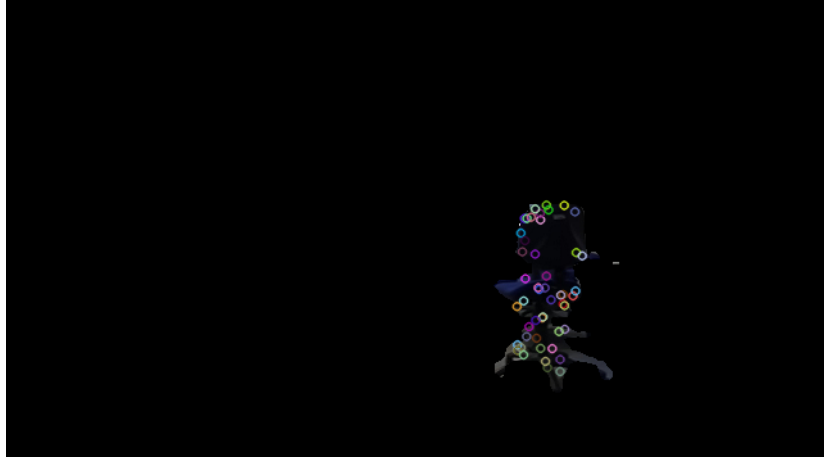


Figure 3.3. Local features state defined by keypoints.

3.2.4. Neighboring Objects' Relations

The fifth component of the state x_O of an object candidate O encodes information regarding the neighbors $\mathcal{N}(O) \subset \mathcal{O}_{pk}^*$ of an object candidate. Specifically, an object candidate $O' \in \mathcal{O}_{pk}^*$ is a neighbor of O iff they are in close proximity - namely $\delta(O, O') \leq \tau_n$ where τ_n is a predefined neighborhood distance threshold. Consequently, for each $O \in \mathcal{O}_{pk}^*$, its neighborhood set $N(O)$ can be formally defined as

$$\mathcal{N}(O) = \{O' \in \mathcal{O}_{pk}^* \mid \delta(O, O') \leq \tau_n\}. \quad (3.2)$$

Let C_O and $C_{O'}$ be the centers of the respective ellipsoids. While defining the proximity of two candidate point cloud objects based on the distance $\delta(C_O, C_{O'}) = \|C_O - C_{O'}\|$ between the centers of these ellipsoids is simple, this can be misleading as it does not consider their respective shapes or their positions with respect to the robot's field of view. An alternative approach is to compute the distance between the closest points of the objects. However, identifying the closest points may require extracting the contour points of the objects and performing point-by-point comparisons, which is computationally expensive and unsuitable for real-time systems. Two alternative approaches have been considered based on the ellipsoid representation of any pair of two objects $O, O' \in \mathcal{O}_{pk}^*$ as defined by $E_O = (A_O, C_O)$ and $E_{O'} = (A_{O'}, C_{O'})$ respectively. In the first approach, if two objects intersect, they can be considered as neighbors.

According to Proposition 2 of [34], the following result holds.

Proposition 3.1. $E_O \cap E_{O'} = \emptyset$ iff $\exists s \in (0, 1)$ s.t $K_{O,O'}^*(s) < 0$ where $K_{O,O'}^* = \min_{s \in (0,1)} K_{O,O'}(s)$.

The scalar valued function $K_{O,O'} : (0, 1) \rightarrow \mathbb{R}$ is defined as

$$K_{O,O'}(s) = 1 - (C_O - C_{O'})^T \left(\frac{1}{1-s} A_O^{-1} + \frac{1}{s} A_{O'}^{-1} \right)^{-1} (C_O - C_{O'}). \quad (3.3)$$

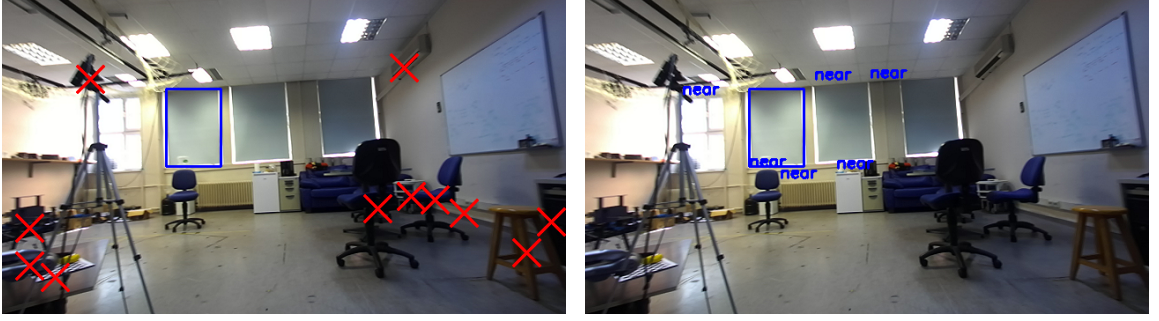
A second alternative is to compute the closest distance between the ellipsoids using the Moving Balls algorithm [35] and then checking whether the resulting distance is smaller than the proximity threshold τ_n or not. However, as the computational complexity of this algorithm is high, the number of pair of objects for which it will be applied is reduced. For this, a heuristic neighbors' elimination algorithm has been developed. In this algorithm, the ellipsoid model E_O of each candidate point cloud object is transformed into an averagely fitting sphere centered at C_O with radius ρ_o ,

$$\rho_o = \frac{1}{3} \sum_{l=1}^3 \frac{1}{\sqrt{\lambda_l}}. \quad (3.4)$$

where the values λ_l , $l = 1, 2, 3$ are the eigenvalues of the ellipsoid shape matrix A_O . Hence, the distance between any pair of candidate point cloud objects is coarsely computed by the closest distance between the respective two spheres,

$$\delta(O, O') = \|C_O - C_{O'}\| - (\rho_O + \rho_{O'}). \quad (3.5)$$

If $\delta(O, O') > \tau_s$, where τ_s is the sphere distance threshold, then it can be inferred that $O' \notin \mathcal{N}(O)$ since the spheres are not sufficiently close. Otherwise, the closest distance between the two enclosing ellipsoids is computed using the Moving Balls algorithm. The heuristic neighbors' elimination algorithm has led to two-fold speeding of finding neighbors. A sample case is shown in Fig. 3.4. For a candidate point cloud object represented by the bounding box, the point cloud objects eliminated by the heuristic elimination algorithm, marked with crosses, are shown in Figure 3.4(a). The distance of this object to the remaining objects is then computed by the Moving Balls algorithm. The resulting neighbors are as shown in Figure 3.4(b).



(a) Heuristic neighbors' elimination algorithm.

(b) Neighbors as identified by their labels

Figure 3.4. Finding neighbors: First, unlikely neighbors are found using a heuristic neighbors' elimination algorithm. Following, moving balls algorithm is applied to the remaining pairs of objects.

For the state, the encoding of the neighbors is represented by a labeled graph G_O . The nodes of this graph correspond to the candidate point cloud object O and its neighbors $\mathcal{N}(O)$. For each $O' \in \mathcal{N}(O)$, the edge label $\eta(O, O')$ is determined depending on its proximity to the object O . The edge label $\eta(O, O')$ can be determined as

$$\eta(O, O') = \begin{cases} \text{Touching} & \text{if } \delta(O, O') = 0 \\ \text{Near} & \text{if } 0 < \delta(O, O') \leq \tau_n. \end{cases} \quad (3.6)$$

These relationships are symmetric.

3.3. Pairwise Similarity of Point Cloud Object Candidates

Consider the sets $\mathcal{O}_{pk}^*$ and $\mathcal{O}_{pk'}^*$ of candidate point cloud objects associated with frame F_{pk} and $F_{pk'}$ respectively where $k' < k$. As a starting point, let $k' = k - 1$. Let $O \in \mathcal{O}_{pk}^*$ and $O' \in \mathcal{O}_{pk'}^*$ be two candidate point cloud objects from each set respectively. Let each be associated with ellipsoid as defined by $E_O = (A_O, C_O)$ and $E_{O'} = (A_{O'}, C_{O'})$. For convenience, we measure dissimilarity so that the smaller its value is, the more similar the two objects are. The dissimilarity of O and O' is defined by the function $\gamma : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$ consisting of five terms $\gamma_i : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$ - each of which measures the dissimilarity of one feature associated with the i^{th} state element,

$$\gamma(x_O, x_{O'}) = \sum_{i=1}^5 w_i \frac{2}{\pi} \arctan(\gamma_i(x_O, x_{O'})) \quad \text{where} \quad \sum_{i=1}^5 w_i = 1. \quad (3.7)$$

As the range of γ_i terms changes depending on the feature, a normalization function $\arctan()$ is applied to each term so that one term does not dominate the others.

The weights w_i , $i = 1, \dots, 5$ determine the contribution of the different features to the overall dissimilarity value γ . The summation of the weights is equal to one and the weights are adjusted according to the importance of the features as determined manually. For instance, although the color is a useful feature for matching objects, it can be misleading in varying lighting conditions. So its weight is selected as relatively small. On the other hand, the neighborhood and position of an object are unique features, so they can be assigned relatively higher weights. Assigning a large weight to the positional similarity can significantly improve tracking performance when an object appears in consecutive frames. However, it may negatively affect the tracking performance if there is a long time gap between the object's appearances. To mitigate the negative impact of the position feature, one approach sets the position dissimilarity to 1 if the remaining feature dissimilarities are sufficiently high - as this indicates that the two object candidates are significantly different based on non-position features.

The position dissimilarity $\gamma_1 : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$ is simply defined as the Euclidean distance between centers of respective ellipsoids E_O and $E_{O'}$,

$$\gamma_1(x_O, x_{O'}) = \|C_O - C_{O'}\|. \quad (3.8)$$

The shape dissimilarity $\gamma_2 : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$ is defined based on a modified version of a metric for comparing covariance matrices [36]. In the modification, the covariance matrices are replaced by shape matrices A_O and $A_{O'}$ - as they are similarly symmetric and positive-definite matrices,

$$\gamma_2(x_O, x_{O'}) = \sqrt{\text{tr}(\ln^2(\sqrt{A_O^{-1}} A_{O'} \sqrt{A_O^{-1}}))}. \quad (3.9)$$

The color dissimilarity $\gamma_3 : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$ is defined based on comparing the two hue N_h -dimensional histograms H_O and $H_{O'}$ associated with each object as

$$\gamma_3(x_O, x_{O'}) = \sqrt{1 - \frac{1}{\sqrt{\bar{H}_O \bar{H}_{O'}} N_h^2} \sum_{i=1}^{N_h} \sqrt{H_O(i) H_{O'}(i)}} \quad (3.10)$$

where $H_O(i)$ and $H_{O'}(i)$ are the values at the i -th bin of histograms H_O and $H_{O'}$. The values $\bar{H}_O$ and $\bar{H}_{O'}$ refer to the mean values of the histograms.

The term $\gamma_4 : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$ measures the dissimilarity of their respective key-point. For this, Fast Library for Approximate Nearest Neighbors (FLANN) is used [37]. FLANN uses Euclidean distance to compute the descriptor distance and identifies the closest match for each descriptor of O in O' . The value γ_4 is calculated as the average of these descriptor distances.

The term $\gamma_5 : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{\geq 0}$ measures the neighborhood dissimilarity. For each neighbor $\hat{O} \in \mathcal{N}(O)$ and $\tilde{O} \in \mathcal{N}(O')$, the dissimilarity of their states with respect to position, shape, color, and keypoints is computed as shown in Algorithm 1.

Algorithm 1 Neighborhood Dissimilarity

```

1: Input: Neighbor sets  $\mathcal{N}(O)$  and  $\mathcal{N}(O')$ 
2: Output: Minimum neighbor dissimilarity  $\beta$ 
3: Initialization:  $IndexPairs \leftarrow \mathcal{N}(O) \times \mathcal{N}(O')$ 
4: for  $n = 1 : |IndexPairs|$  do
5:    $(\hat{O}, \tilde{O}) = IndexPairs(n)$ 
6:   if  $\eta(O, \hat{O}) = \eta(O', \tilde{O})$  then
7:      $\nu(x_{\hat{O}}, x_{\tilde{O}}) \leftarrow \sum_{i=1}^4 \frac{2}{\pi} \arctan(\gamma_i(x_{\hat{O}}, x_{\tilde{O}}))$ 
8:   end if
9: end for
10:  $sortedPairs \leftarrow IndexPairs$  sorted by  $\nu$ 
11:  $\beta = 0$ 
12: for  $\hat{O} \in \mathcal{N}(O)$  do
13:    $\beta \leftarrow \beta + \frac{1}{|\mathcal{N}(O)|} \min_{\tilde{O} \in \mathcal{N}(O')} \nu(x_{\hat{O}}, x_{\tilde{O}})$ 
14: end for

```

3.4. Multi-Object Tracking

Multi-object tracking is comprised of two coupled parts: Data association and maintaining object tracks. Through data association, each object in the refined set of candidate point cloud objects $O \in \mathcal{O}_{pk}^*$ is matched with those detected in the previous frames. For data association, two of the most popular algorithms are Hungarian and Greedy algorithms [38]. In this work, each method has been tried separately. The resulting matched pairs (O, O') are accepted only if their dissimilarity value is sufficiently low - namely whether $\gamma(x_O, x_{O'}) < \tau_d$ where τ_d is a predefined threshold. Data association is extended to consider object candidates observed in a time window Δk . In this case, dissimilarity is possibly searched - considering set $\mathcal{O}_{pk'}^*$ candidate point cloud objects for $k' = k - 1, \dots, k - \Delta k$. At each stage of data association, unmatched object candidates from both $\mathcal{O}_{pk}^*$ and $\mathcal{O}_{pk'}^*$ are taken into consideration. This step ensures that the dissimilarity is calculated exclusively between the remaining unmatched object candidates. For example, let $\mathcal{O}_{pk}^{*M} \subset \mathcal{O}_{pk}^*$ denote the matched object candidates of $\mathcal{O}_{pk}^*$ and $\mathcal{O}_{pk}^{*U} \subset \mathcal{O}_{pk}^*$ denote unmatched object candidates - considering the previous frame $k - 1$. Initially, the dissimilarity values object candidates in $\mathcal{O}_{pk}^*$ and those of $\mathcal{O}_{p(k-1)}^*$ are computed. In case there are unmatched object candidates - namely $\mathcal{O}_{pk}^{*U} \neq \emptyset$, then these object candidates are matched with unmatched object candidates from $k - 2^{th}$ frame - namely $\mathcal{O}_{p(k-1)}^{*U} \subset \mathcal{O}_{p(k-2)}^*$. The search and association processes are applied until either unmatched object candidates associated with previous Δk frames are considered or all object candidates in $\mathcal{O}_{pk}^*$ are matched. The processing flow is given in the Algorithm 2.

Algorithm 2 Track search

```

1: Input:  $\gamma$  : Dissimilarity function
2:  $\tau_d$ : Threshold for matching
3: Initialization:
4:  $t \leftarrow 1$  and  $k' \leftarrow k - t$ 
5:  $\mathcal{O}_{pk'}^{*M} \leftarrow \emptyset$  : Matched object from  $\mathcal{O}_{pk'}^*$ 
6:  $T \leftarrow \emptyset$  : Temporary matched object from  $\mathcal{O}_{pk'}^*$ 
7:  $\mathcal{O}_{pk'}^{*U} \leftarrow \mathcal{O}_{pk'}^*$  : Unmatched object from  $\mathcal{O}_{pk'}^*$ 
8:  $\mathcal{O}_{pk}^{*M} \leftarrow \emptyset$  : Matched object from  $\mathcal{O}_{pk}^*$ 
9:  $\mathcal{O}_{pk}^{*U} \leftarrow \mathcal{O}_{pk}^*$  : Unmatched object from  $\mathcal{O}_{pk}^*$ 
10: while  $k' \geq k - \Delta k$  do
11:    $matchedPairs \leftarrow$  data association algorithm between  $\mathcal{O}_{pk}^{*U}$  and  $\mathcal{O}_{pk'}^{*U}$ 
12:   for  $n = 1 : |matchedPairs|$  do
13:      $(O, O') = matchedPairs(n)$ 
14:     if  $O' \notin \mathcal{O}_{pk'}^{*M}$  and  $O \notin \mathcal{O}_{pk}^{*M}$  then
15:       if  $\gamma(O, O') < \tau_d$  then
16:          $\mathcal{O}_{pk'}^{*M}.append(O')$ 
17:          $\mathcal{O}_{pk}^{*M}.append(O)$ 
18:       else
19:         break
20:       end if
21:     end if
22:   end for
23:    $\mathcal{O}_{pk}^{*U} \leftarrow \mathcal{O}_{pk}^* - \mathcal{O}_{pk}^{*M}$ 
24:    $T \leftarrow \mathcal{O}_{pk'}^{*M}$ 
25:   if  $\mathcal{O}_{pk}^{*U} > 0$  then
26:      $t \leftarrow t + 1$ 
27:      $\mathcal{O}_{pk'}^{*U} \leftarrow \mathcal{O}_{pk'}^* - T$ 
28:   else
29:     break
30:   end if
31: end while

```

Differing from most work, each point cloud object $O \in \mathcal{O}_{pk}^*$ is assigned a track $T^Q \in \mathcal{T}_{pk}$ and stored in the ‘Working Object Candidates Memory’ (WOCM). Thus, the sequence of matched candidate point cloud objects is retained through the object tracks. Each track aims to represent a distinct point cloud object as indexed by Q that is seen in the incoming RGB-D stream. It is defined by an ordered set of states $T^Q = \{x_O(k) \mid k \in \mathcal{K}_Q\}$ where $\mathcal{K}_Q$ denotes the index set of the frames in which the object is detected. If the index set is defined as $\mathcal{K}_Q = \{k_1, \dots, k_N\}$, k_1 indicates when the track starts and k_N indicates the last index in the track. Each entry in this set indicates that the object has been detected in the respective frame. As long as the robot tracks a candidate point cloud object Q , the set $\mathcal{K}_Q$ expands accordingly. Whenever a candidate point cloud object $O \in \mathcal{O}_{pk}^*$ cannot be matched in the time window, the index Q is incremented by one and a new track T^Q is started. This candidate is added to the top of the WCOM along with the unique object ID Q . The complete state information of each track is stored in the working candidate objects memory (WCOM) as defined by the set $\mathcal{T}_{pk}$ of tracks. In the new place, each new track is added to the WCOM. The tracks in the WCOM encode the knowledge acquired by the robot by looking around or moving in the place. Objects viewed currently by the robot are encoded by tracks with the last indices k_N matching the current frame index k . Objects that are no longer visible to the robot are recalled by the tracks with $k_N < k$. Note that a large difference $k - k_N$ implies that the candidate point cloud object has not been detected for a while. However, as long as the robot is moving in place, tracks remain in the WCOM. The set $\mathcal{T}_{pk}$ is reset whenever the robot moves to a new place p .

3.5. Object Cognition

The set $\mathcal{T}_{pk}$ refers to all the tracks being retained in the WCOM at time k . This is advantageous since the reasoning can then consider the track information rather than that of a single candidate point cloud object. In particular, tracks with cardinality $\|T^Q\| \geq N_O$ are then passed to the recognition module. The parameter N_O is the minimum cardinality threshold.

As object detection and object recognition are decoupled, it is possible to use a state-of-the-art method for object cognition. The object cognition process is viewed as consisting of two integral parts: First, there should be a long-term objects' memory (LTOM) which retains previously recognized objects' knowledge. Second, the recognition module should be capable of relating each point cloud object data in the WCOM to LTOM as to either determine whether it is a previously seen object or a new object. In the latter case, the new object needs to be learned by adding it to the existing LTOM.

Each track $T^Q \in \mathcal{T}_{pk}$ is associated with a status: 'labeled' or 'not labeled' depending on whether it is assigned an object label or not. All tracks having 'not labeled' status are processed by the recognition module depending on their cardinality. In particular, they are processed if their cardinality is a multiple of a minimum cardinality threshold N_O - namely if $\|T^Q\| = mN_O$ where $m = 1, 2, \dots$. Since the recognition process is computationally expensive, this strategy minimizes unnecessary invoking of the recognition module for a track.

For each track $T^Q \in \mathcal{T}_{pk}$ that is 'not labeled' but having cardinality $\|T^Q\| = mN_O$, ($m = 1, 2, \dots$), the robot attempts to recognize the respective object Q - considering all the associated candidate point cloud objects. If $T^Q = \{x_O(k) \mid k \in \mathcal{K}_Q\}$ where $\mathcal{K}_Q$ denotes the index set of the frames in which the candidate point cloud object was matched to Q . In particular, the recognition decision is based on the majority label. Namely, the label with the maximum number of votes is considered. If the resulting percentage exceeds the recognition threshold τ_r , the track's status changes to 'labeled'. The point cloud object set $\mathcal{O}_p$ associated with this place is then updated as

$$\mathcal{O}_p(k) = \mathcal{O}_p(k-1) \cup \{Q\}. \quad (3.11)$$

Once a track becomes labeled, subsequent additions to the track do not affect or change its label.

3.6. Experimental Results

This section details the construction of two datasets, the performance evaluation metrics, and the experimental results of point cloud object tracking. The study evaluates the impact of ellipsoid fitting using the Khachiyan and PCA methods and analyzes the effect of data association algorithms by comparing the Hungarian and greedy algorithms. Tracking evaluation is conducted based on metrics such as the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN), as well as the sizes of the segments $|S|$ associated with respective object candidates for each TP, FP, TN, and FN.

3.6.1. Datasets

Two separate benchmark data sets have been built - as there are no such publicly available indoor RGB-D datasets with sequential frames and objects' annotation.

3.6.1.1. Isaac Sim Dataset. Isaac Sim dataset has been generated using the NVIDIA Isaac Sim application [39] that is built on NVIDIA Omniverse. Its advantages are providing photorealistic virtual environments and pixel-level annotations. The Office building environment is one of the available simulation environments. It is comprised of multiple rooms - each of which can be considered a distinct place. Each place is simulated like a real office building room. While there are numerous places in the Office building environment, the current dataset includes four distinct places. In particular, the dataset consists of RGB-D data generated with a simulated ZEDX RGB-D stereo camera that moves in each place. The camera model incorporates a Gaussian noise model on depth information to better represent real-world data. The ZEDX Isaac Sim simulation camera was moved around four places in the Office building environment, and frames were recorded.

The tracking evaluation experiments are done considering a subset of 130 frames that have been acquired from a single office. Sample frames are shown in Figure 3.5.



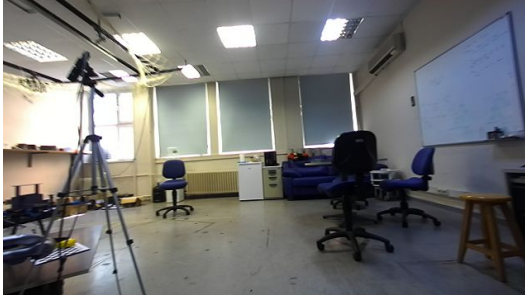
(a) Frame 1.

(b) Frame 30.

(c) Frame 60.

Figure 3.5. Sample frames in ISAAC dataset.

3.6.1.2. Real-World Dataset. A real-world dataset has been constructed via scanning the Intelligent Systems Laboratory with a hand-held ZED2 camera first from right to left and then from left to right. It consists of 29 frames. Some sample frames are as seen in Figure 3.6. For ground-truth information, the data has been annotated manually using the masks generated by FastSAM.



(a) Real-world frame 1.



(b) Real-world frame 16.



(c) Real-world frame 22.



(d) Real-world frame 28.

Figure 3.6. Sample frames from real-world dataset.

3.6.2. Performance Evaluation

Most related work on evaluating tracking performance use metrics that also assess detection performance since detection and tracking are done simultaneously. In the proposed approach, this is not applicable as detection is done separately. Rather, the definitions of true positive, false positive, true negative, and false negative are revised as to incorporate only tracking performance:

- True Positives(TP): A candidate point cloud object is associated with a track correctly.
- False Positives(FP): A candidate point cloud object is matched with a track incorrectly.
- False Negatives(FN): While the candidate point cloud object should be matched with a track, this is not the case and a new track is initiated.
- True Negatives(TN): The candidate point cloud object is seen for the first time. It is not associated with a track and thus a new track is initiated.

The given metrics are calculated in two ways:

- Object-based: Overall TP, FP, TN, and FN are determined by counting their occurrences in object candidates.
- Pixel-based: The overall TP, FP, TN, and FN are computed by summing the pixel counts $|S|$ of the object candidates image assigned to each respective TP, FP, TN, and FN.

Pixel-based metrics provide insights into the size of the tracked object candidates. For instance, if count-based metrics are relatively high but pixel-based metrics are not, it indicates that the mistracked object candidates are those with sizes significantly larger than others.

Consider a sequence of m frames given in Figure 3.7. Let F_{pk} be an incoming frame, $F_{p(k-1)}$ be the last frame in WOCM, and $F_{p(k-\Delta k)}$ be the oldest frame to search backward. The numbers on the symbolic object candidates denote the IDs of object candidates. In the figure, each candidate object within F_t is labeled with the corresponding metric components TP, FP, FN, and TN. If a match has been made, the accuracy of the match is learned by checking the IDs of the matched objects. If the two matched object candidates have the same ID then, the object candidate in F_{pk} is considered as TP. If they have different IDs, then they are considered as FP. For instance, object 6 in F_{pk} was matched with object 6 in $F_{p(k-1)}$. Since they have the same IDs, object 6 in F_{pk} was considered as TP. On the other hand, object 8 in F_{pk} was matched with object 7 in $F_{p(k-1)}$. Since they have different IDs, object 8 in F_{pk} was considered as FP. Also, if an object candidate in F_{pk} is not in the WOCM namely, it is a new object candidate, but it is still matched with an object candidate in the WOCM, then it is considered as FP. For instance, object 15 in F_{pk} was matched with object 4 in $F_{p(k-\Delta k)}$ although it is not actually in the WOCM. Therefore, object 15 was considered as FP. If an object candidate in F_{pk} is actually in the WOCM, however, it is not matched with anyone and added as a new object candidate, then it is considered as FN. For instance, object 9 in F_{pk} did not match any object candidate in WOCM, thus it was considered as FN. If an object candidate in F_{pk} is a new object candidate and it is not matched with any object candidate in WOCM, namely is added as a new object candidate, then it is considered as TN. For instance, object 14 in F_{pk} did not match any object candidate in WOCM, thus was considered as TN.

The parameters used in both experiments are presented in Table 3.1. The parameter values were determined based on empirical results. The differences in parameters between the real and simulated datasets primarily arise from variations in camera resolution. The weights are set by evaluating their contribution to the discriminative performance as discussed in Section 3.3.

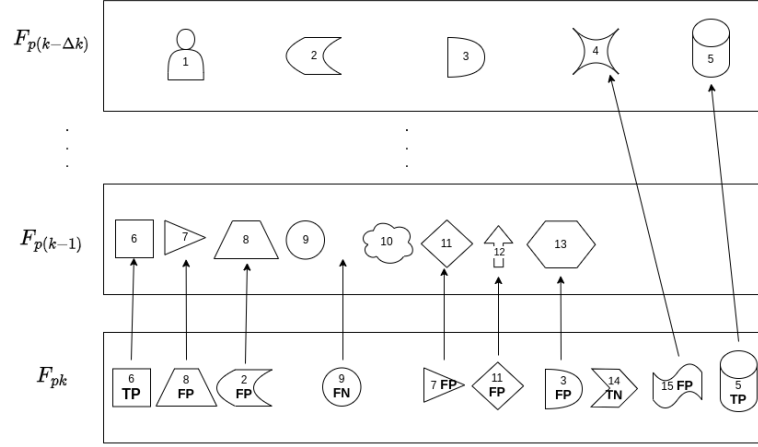


Figure 3.7. Computation of tracking performance metrics.

Table 3.1. Parameter values used for tracking.

Dataset	τ_u	τ_l	τ_m	τ_n	τ_s	τ_d	w_1	w_2	w_3	w_4	w_5
Real-World	10.85	0.22	0.31	1.5	3	0.7	0.25	0.3	0.1	0.05	0.3
Isaac Sim	3.55	0.023	0.31	2.15	4.3	0.7	0.25	0.3	0.1	0.05	0.3

Tracking performance is evaluated on both real-world and Isaac Sim datasets with varying ellipsoid fitting and data association configurations. Khachiyan and PCA-based methods are used for ellipsoid fitting. For data association, Hungarian and greedy algorithms are considered.

Execution time consists of three different parts: tracking time, feature extraction time, and total time. Tracking time denotes the time consumption by the tracking module and the feature extraction time denotes the time consumption by the feature extractor module, which includes ellipsoid fitting, as well as color, point cloud, shape, local descriptor, and neighbor extractions. The execution time of both the tracking and feature extraction modules represents the overall execution time, and it was calculated by summing their execution times each time they were called throughout the entire dataset. The total time was obtained by summing the time consumed by the tracking and feature extraction modules.

3.6.3. Experiments on Isaac Sim Dataset

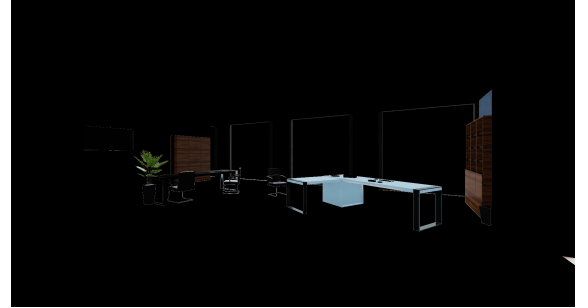
The tracking performance is evaluated on the Isaac Sim dataset. Object candidates are generated using FastSAM as explained previously. For ablation studies, we also consider the segments provided by Isaac-Sim. Hence, the dependency of the tracking algorithm on the segmentation performance is removed. For a sample frame from the Isaac Sim dataset as seen in Figure 3.8(a), the object candidates obtained by FastSAM and Isaac Sim are shown in Figure 3.8(b) and Figure 3.8(c) respectively.



(a) A sample Isaac Sim frame.



(b) Segments determined by FastSAM.



(c) Segments obtained from Isaac Sim.

Figure 3.8. Segments of a sample Isaac Sim frame.

Tracking performance with FastSam segments is given in Table 3.2. The Hungarian algorithm is observed to yield significantly better tracking performance. It is worth noting that a high recall value in this context might result in mistakenly considering a new object candidate as an existing one. The results of using Isaac Sim segments is given in Table 3.3.

Table 3.2. Tracking algorithm performance in Isaac Sim dataset with FastSAM segments.

Dataset	Data Association	Ellipsoid Fitting	Performance Metrics								Execution Time (sec.)		
			Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1	Tracking	Feature Extractor	Total
			Object Count				Pixel Count						
Isaac Sim	Hungarian	Khachian	98.989	98.1768	97.2615	98.5812	85.3357	76.9557	79.5756	80.9294	41.195	829.461	870.656
	Hungarian	PCA	98.5388	98.8095	97.4382	98.674	85.2585	89.7529	86.3426	87.448	35.644	177.172	212.816
	Greedy	Khachian	97.0723	98.6059	95.8481	97.8331	78.3255	77.8835	76.6075	78.1038	40.410	812.124	852.534
	Greedy	PCA	96	98.9691	95.1413	97.4619	76.0232	90.0324	79.6625	82.4369	34.645	173.180	207.825

Table 3.3. Tracking performance in Isaac Sim dataset with Isaac Sim segments.

Dataset	Data Association	Ellipsoid Fitting	Performance Metrics						Execution Time (sec.)				
			Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1	Tracking	Feature Extractor	Total
			Object Count			Pixel Count							
Isaac Sim	Hungarian	Khachian	99.5562	99.6299	99.2017	99.593	89.1926	77.4268	80.5471	82.8943	137.045	2016.735	2153.780
	Hungarian	PCA	99.6669	99.63	99.3106	99.6485	97.0061	78.1247	85.267	86.5476	154.577	126.4449	1419.026
	Greedy	Khachian	99.0789	99.7774	98.8752	99.4269	80.8913	82.8306	78.3094	81.8495	128.968	1915.074	2044.042
	Greedy	PCA	99.1529	99.7776	98.9478	99.4643	77.5947	82.8738	76.1721	80.1474	155.806	129.1082	1446.888

Table 3.3 demonstrates that the overall results are better than those achieved with tracking using FastSAM. This is because the segments extracted with FastSAM may include parts that do not actually belong to the intended segments. For instance, in Figure 3.8(b), the blue table and a chair contain some ground parts. Such inaccuracies can affect the color information obtained from these objects or lead to shape inconsistencies if the unrelated parts are not consistent across all frames. Furthermore, Table 3.3 indicates that the combination of Hungarian + PCA-based ellipsoid fitting once again yields the best results across all metrics, except for recall. However, the recall value does not show any significant difference.

Table 3.2 shows that Greedy required less tracking time than Hungarian. Similarly, Table 3.3 indicates that Greedy outperforms the others with the Khachiyan-based ellipsoid fitting. Greedy may be faster due to its lack of global optimization if object candidate dissimilarities are quite different. For feature extraction, as shown in both Table 3.2 and Table 3.3, Khachiyan-based fitting takes longer than PCA-based, as it encapsulates all points to find the minimum ellipsoid.

Table 3.3 shows that tracking and feature extraction take more time across all combinations compared to the combinations in Table 3.2. This increase is possibly due to the use of Isaac Sim segments, which introduced additional object candidates not recognized by fastSAM. Consequently, the higher number of object candidates increases the time required for both tracking and feature extraction.

3.6.4. Experiments on Real-World Dataset

To optimize the tracking process, WOCM is scanned within a predefined time window. This constraint primarily impacts the matching of object candidates that appear outside this window. However, given that the annotated real-world dataset consists of a limited number of frames, tracking object candidates back to the beginning of WOCM remains feasible. Accordingly, Table 3.4 shows results for scanning within a time window, while Table 3.5 presents results for scanning extended to the beginning.

Table 3.4. Tracking performance with different algorithms and scanned back to a certain frame.

Dataset	Data Association	Ellipsoid Fitting	Performance Metrics						Execution Time (sec.)				
			Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1	Tracking	Feature Extractor	Total
			Object-Based			Pixel-Based							
Real-World	Hungarian	Khachian	96.9838	93.9326	92.0949	95.4338	85.9266	58.6394	67.7776	69.7077	100.919	90.234	191.153
	Hungarian	PCA	97.2665	95.5257	93.6759	96.3883	89.1305	68.8566	76.0604	77.6927	89.948	22.629	112.577
	Greedy	Khachian	95.3917	93.8776	90.7115	94.6286	69.1284	57.1754	61.1426	62.5863	102.327	92.944	195.271
	Greedy	PCA	94.5701	95.4338	91.3043	95	74.1509	67.4431	69.0096	70.6381	91.089	23.036	114.125

Table 3.5. Tracking performance with different algorithms and scanned back to the beginning of WCOM.

Dataset	Data Association	Ellipsoid Fitting	Performance Metrics								Execution Time (sec.)		
			Precision	Recall	Accuracy	F1	Precision	Recall	Accuracy	F1	Tracking	Feature Extractor	Total
			Object-Based				Pixel-Based						
Real-World	Hungarian	Khachian	96.4286	97.2973	94.4664	96.861	85.6681	75.9912	79.0085	80.54	122.108	94.734	216.842
	Hungarian	PCA	96.0526	98.6486	95.2569	97.3333	77.8425	87.0317	80.6836	82.181	102.863	22.868	125.731
	Greedy	Khachian	94.9002	97.27276	93.083	96.0718	68.9689	75.4956	70.724	72.0848	118.301	92.094	210.395
	Greedy	PCA	93.4641	98.6207	92.8854	95.9732	73.1748	86.3888	77.8034	79.2347	103.649	22.307	125.956

Based on Table 3.4, Hungarian matching algorithm gives the best results for all metrics when we compare it to the greedy matching algorithm particularly when PCA is used for the ellipsoid fitting. According to the metrics, the precision metric demonstrates correct object candidate associations. Namely, the tracking algorithm frequently identifies the true object candidate in memory. However, the recall metric suggests that the tracking sometimes considers an object already present in memory as a new object candidate. As shown in Table 3.5, this behavior may stem from the tracking algorithm’s limited search range in memory, which is restricted to a certain number of frames. Consequently, if an object appears at the beginning of the sequence and reappears after several frames, it may not be associated with its previous instance and is instead added as a new object candidate. The Greedy algorithm also performs best when PCA is used; however, its approach introduces a significant limitation. Consider a scenario where O_i and O_j are two object candidates in the incoming frame, and $o' \in T^Q$ is an object candidate that is in the track of object Q . Consider the dissimilarity values $\gamma(x_{O_i}, x_{O'})$ and $\gamma(x_{O_j}, x_{O'})$. With the greedy algorithm, if O_i is matched to O' , it is not possible for O_j to match with O' even if its dissimilarity value is the lowest among all the existing tracked objects. This limitation of the Greedy algorithm reduces the correctness of the matches. Hence, the Hungarian algorithm consistently outperforms the Greedy algorithm in terms of accuracy and overall performance.

From Table 3.4, we observe that PCA-based ellipsoid provides better results compared to the Minimum Volume Enclosing Ellipsoid (MVEE) based on Khachiyan’s algorithm. This may be due to Khachiyan’s method of constructing an ellipsoid that encompasses all the points within the point cloud of an object candidate. One drawback of this approach is that, even after point cloud filtering, any remaining outliers in the cloud are also included in the MVEE. As a result, the representation of an object candidate with an ellipsoid becomes less accurate, which significantly affects the shape dissimilarity. Additionally, this inaccurate representation impacts the determination of an object candidate’s neighbors. A larger ellipsoid, resulting from the inclusion of outliers, increases the number of neighbors, thereby complicating the process of identifying

similarities between the neighbors.

When analyzing the tracking time in Table 3.4, we observe that the Hungarian algorithm is slightly faster than the Greedy algorithm. This may be because, instead of focusing on finding the absolute minimum distance, the Hungarian algorithm minimizes the total dissimilarity. This approach allows for matches like o_2 and o' without requiring extensive memory scans. As a result, the tracking algorithm spends less time searching the memory, reducing the overall tracking time. In contrast, when comparing these results with those in Table 3.5, we notice a significant increase in tracking time. This is primarily due to the larger number of frames scanned in this scenario, which understandably increases the computational load. The feature extractor time is mostly related to the ellipsoid fitting.

To assess the computational time of the Feature Extractor, we vary the ellipsoid fitting algorithm while keeping the other components, color, point cloud, local descriptor, and neighbor extractions, of the feature extractor module unchanged. This allows us to evaluate and compare the time consumption of the MVEE based on Khachiyan's algorithm and the ellipsoid derived from PCA. Examining the feature extractor time in Table 3.4, it is evident that the MVEE based on Khachiyan's algorithm consumes significantly more time than the ellipsoid derived from PCA. This is attributed to the computationally intensive nature of the process required to determine the minimum encapsulating ellipsoid, which involves iterative calculations to ensure that all points within a given point cloud are enclosed. In contrast, the ellipsoid based on PCA involves fewer processes and does not prioritize including all points, resulting in reduced computation time. The feature extractor time in Table 3.5 is almost identical to that in Table 3.4. This consistency arises because each object's features are extracted for an object candidate only once and are stored in the memory frame. Even when the tracking algorithm searches the memory up to a certain frame, it does not impact the feature extraction time.

As anticipated, the overall performance on the real-world dataset is lower compared to the Isaac Sim dataset. This difference can be attributed to the synthetic nature of the Isaac Sim data, which introduces minimal noise. Additionally, object candidates in the synthetic dataset have more distinct shapes and precise colors. Moreover, the synthetic environment is less complex, further contributing to improved tracking performance.

In summary, best performance results were achieved using PCA-based ellipsoid fitting and Hungarian algorithm in both dataset. Therefore, the remainder of the study was conducted using PCA-based ellipsoid fitting as the ellipsoid fitting approach and the Hungarian algorithm as the data association method and

4. SEMANTIC PLACE MAP

The semantic place map (SPM) aims to encode knowledge regarding the entities present in a place. This is an internal representation that incorporates the major constituent objects and their spatial relations within each place. The advantages of such a map are two-fold: First, it can help in the robot’s spatial cognitive reasoning. Secondly, such a model is beneficial for robot-human interaction. For example, the semantic place map can be used to enhance the imaginative capabilities of a robot for completing the unseen or occluded portions of objects [40].

In this chapter, we propose a novel approach to the construction of semantic place models. Differing from previous work, a SPM is defined by a graph based on the point cloud objects and their relations that evolves incrementally as the robot is looking around and moving in the place.

4.1. Related Literature

The development of semantic place maps is being increasingly studied with the advancement of social robotics in which robots interact with humans. In this aspect, a semantic place model enables the robot to represent its surroundings so that it can interact with humans. The most general approach is to use a graph representation in which nodes correspond to objects and edges are defined based on their relationships. In some work, the graphs are defined at the pixel level. The semantic place model is limited to a few classes [40,41]. For example, in the latter work, although a significant number of objects are extracted from the RGB image, only four distinct labels are utilized. Alternatively, nodes correspond to identified objects within the place, and edges are constructed based on distances and angles between these objects [42]. As object recognition in this approach relies exclusively on SIFT features, it is sensitive to variations in the local features. Furthermore, the robot does not make use of the incoming video stream to identify the objects. SIFT features are enhanced with shape

information that is derived from depth data [43, 44]. In [44], the place is likewise represented as a graph, with edges defined by spatial and contextual relationships. However, the graph lacks labels for the edges. More recent work has focused on using RGB-D data to construct semantic maps. Some work considers semantic point clouds based on the labels obtained from the semantic segmentation algorithm [45].

4.2. Semantic Place Maps - Formulation

The semantic place map of a place p is defined by a time-varying graph $G_p(k) = (\mathcal{O}_p(k), \mathcal{R}_p(k))$ where the objects are defined by labeled point cloud objects $\mathcal{O}_p(k)$ and the edges are defined by their proximity relations $\mathcal{R}_p(k)$. As the robot is looking around or moving in a place p , its SPM $G_p(k)$ is incrementally constructed. In order to simplify the notation, the time argument will be omitted whenever possible. Object labels refer to the unique class ID for the object and edge labels are defined as either ‘touching’ or ‘near’ as explained in Chapter 3. The robot assigns labels to the object candidates via an object recognition module. Hence, object recognition is integral to SPM. If the robot is to be deployed reliably in previously unseen places, it is crucial for objects’ related cognitive processing to be unsupervised. However, this is still an ongoing research area. As such, the objects’ related cognitive processing is done using a state-of-art approach YOLOv11 [46]. This is an approach that works with RGB segments of the point cloud objects. Since its learning is supervised, it is not able to handle previously unseen objects. Hence, the SPMs that are constructed can include only objects that can be recognized by YOLOv11.

4.3. Semantic Place Maps - Incremental Construction

Suppose the robot enters a place p . As the robot is looking around or moving in a place p , its SPM G_p is incrementally constructed. The flow of processing associated with the construction is as shown in Figure 4.1. For each track T^Q whose status changes from ‘not labeled’ to ‘labeled’, the SPM G_p is updated as: First, nodes are updated. Following, edges are updated considering the edges associated with the newly added

node Q . The incremental construction continues as long as the robot is looking around or moving in place. It is finalized once the robot leaves the place. The finalized SPM is then retained in the memory in conjunction with the respective place.

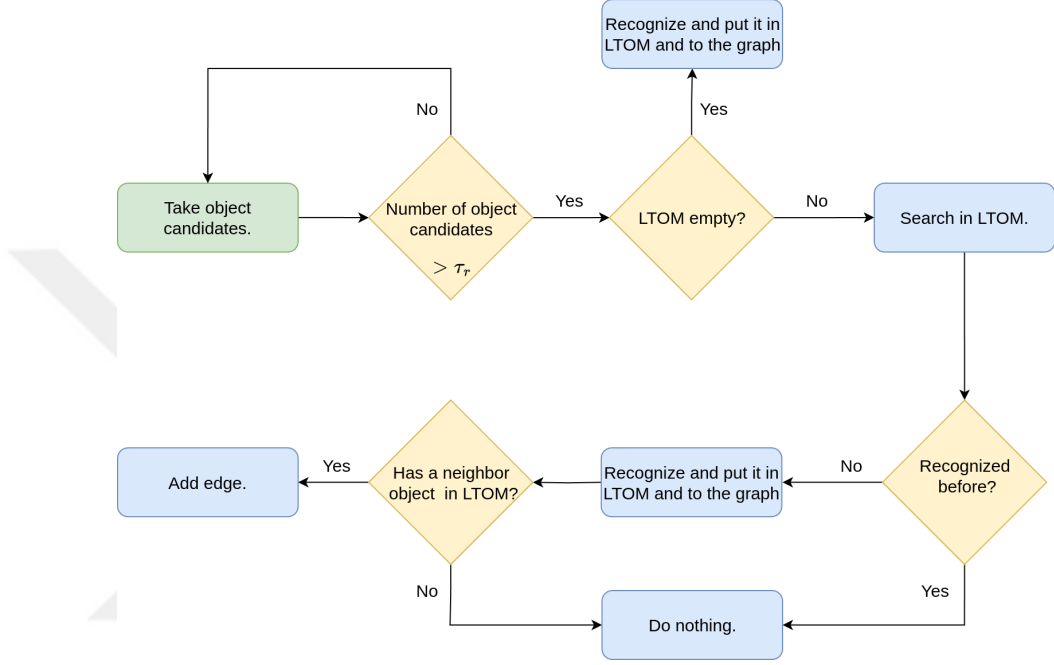


Figure 4.1. Flowchart of incremental graph construction.

4.4. Semantic Place Maps - Recognition

Recognition based on SPMs is done via finding the similarity of the current SMP with those previously learned. For similarity measure, Graph Edit Distance (GED) is used [47]. Here, SMPs are compared pairwise and GED is defined based on the operations required to transform one SMP to the other. The following types of editing operations are possible:

- Node addition/deletion: To introduce/remove a single object to a graph.
- Node substitution: To change the label of an existing object.
- Edge addition/deletion: To introduce/remove a single edge to a graph.
- Edge substitution: To change the label of an existing edge.

Let $\mathcal{W}(G, G')$ be the set of possible paths for transforming G into G' . Each path $\mathcal{D}(G, G') \in \mathcal{W}(G, G')$ is an ordered set of operations. Each edit operation $e \in \mathcal{D}(G, G')$ is associated with a cost $\xi(e)$ depending its type and the types are defined as

$$\xi(e) = \begin{cases} \lambda_n & \text{If type = node addition/deletion} \\ \lambda_e & \text{If type = edge addition/deletion} \\ \lambda_{ns} & \text{If type = node substitution} \\ \lambda_{es} & \text{If type = edge substitution} \end{cases} \quad (4.1)$$

The GED $\phi(G, G')$ is defined as the minimum of sum of costs among the possible ordered set $\mathcal{D}(G, G')$ of transformations,

$$\phi(G, G') = \min_{\mathcal{D}(G, G') \in \mathcal{W}(G, G')} \sum_{e \in \mathcal{D}(G, G')} \xi(e). \quad (4.2)$$

Various exact algorithms have been developed to solve this problem. We used an implementation from NetworkX [48] based on the algorithm [49]. This method transforms the problem into a minimum-cost path-finding problem to provide a depth-first search-based solution.

Finally, a place p associated with SPM G_p is considered to be recognized as place $p^* \in \mathcal{P}$ if $\phi(G_p, G_{p^*}) \leq \tau_g$ where τ_g is a predetermined SPM recognition threshold.

4.5. Experimental Results

Experiments are done in four different places obtained from the Isaac Sim Office simulation environment. The office environment is a default simulation environment in Isaac Sim, comprising multiple rooms that are considered as distinct places. For the experiment, four of them were used. A detailed explanation of the dataset construction process is provided in Chapter 3. Each place is visited multiple times. After the first visit, an SPM of the place is constructed and associated with this place. In the following visits, the robot navigates through the place in a manner similar to the first visit as well as following different routes. Some of the routes have starting positions and viewpoints that are opposite to those of the first visit. In order to study SPM

recognition performance, each place is visited again from a different approach path. First, they visited with the same viewpoint and followed the same camera path. Later, the same environments were revisited with opposite viewpoints and paths.

Parameter values including GED cost parameters used in the experiments are listed in Table 4.1. While the cost values were primarily determined empirically, higher values were assigned to edge operations due to their discriminative nature.

Table 4.1. Parameter values for scene recognition.

Parameter	N_O	λ_e	λ_n	λ_{es}	λ_{ns}	λ_d	τ_g
Value	10	3	1	1	0.3	0.5	1

4.5.1. SPM Construction - First Visits

The robot constructs the respective SPM during the first visit of each place. For place $p^* = 1$ as seen in Figure 4.2 the construction of the SMP G_1 evolves as seen in Figure 4.3. It should be noted that each newly added node is indicated in green while the edges are labeled as T (touching) and N (near). It is observed that the SPM of the place is constructed as objects are being recognized - with the edge relationships between them being added accordingly as new objects were identified and added to the graph. In Figure 4.3(e), new edge relationships were added with the inclusion of a new node. Recognizing the place during the early stages, when edge connections are limited, is challenging. However, as new nodes are added, the increasing number of edges can significantly enhance the recognition of the environment.

In Figure 4.3(a), two nodes, node 546 and node 111, were initially added to the graph. The neighborhood relationship between these nodes is classified as type N, where N denotes 'near.' Node 546 corresponds to object 3, while node 111 corresponds to object 5 in Figure 4.2(a). Upon closer examination of Figure 4.2(a), it becomes evident that these two objects are positioned in close proximity to each other, thereby

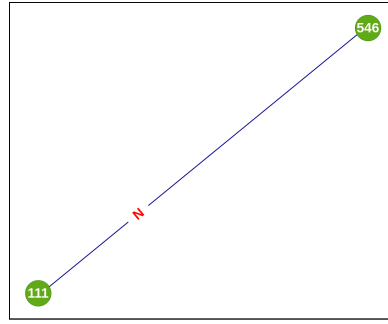
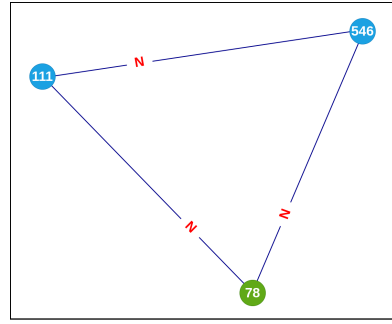
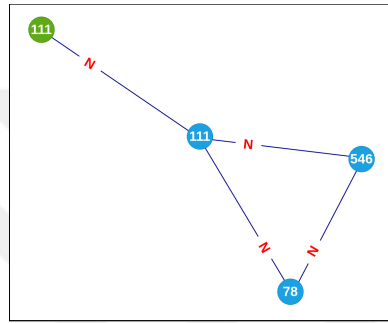
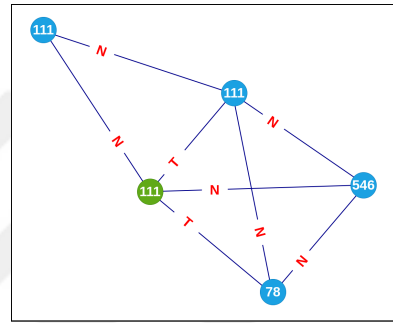
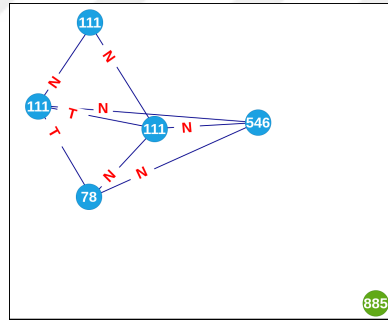
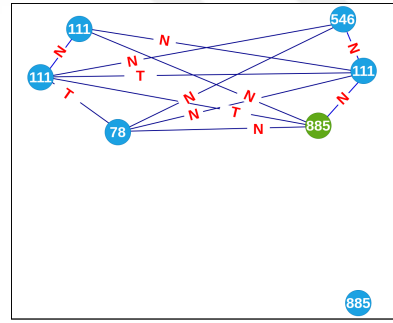
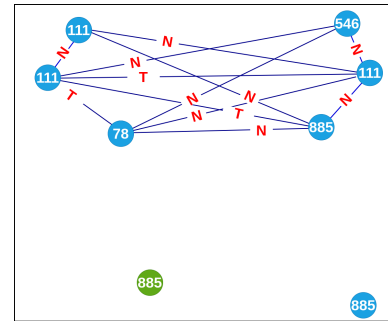
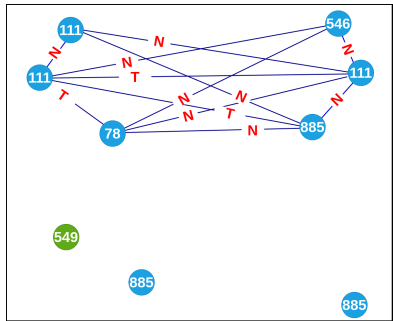
confirming the 'near' relationship assigned between their corresponding nodes. In Figure 4.3(d), a new node 111 was added to the graph. In the place figure, node 111 corresponds to object 8 and its neighbors are node 546 (object 3), 111(object 5), 111 (object 7), and 78 (object 6). In figure 4.3(d), we observe that the relationship type between node 546 and node 111 is identified as 'touching.' This specific type of relationship is also evident in Figure 4.2(a), where a similar interaction is depicted between object 3 and object 5.



(a) Place 1 - Onset. (b) Place 1 - After 30 frames. (c) Place 1 - After 50 frames.

Figure 4.2. Isaac Sim SPM Experiments: (a) place 1 - onset, (b) place 1 after 30 frames, and (c) place 1 after 50 frames.

G_1 is given in Figure 4.3(h) corresponds to the finalized SPM of place 1 - denoted as G_1^* . From Figure 4.3(h), it becomes apparent that object IDs are not unique, as the same label can be assigned to multiple objects within the scene. For instance, label 111 is assigned to three different objects. Among these, one object is connected by three edges, while the other two are connected by five edges. By examining the number of edge connections, it is possible to distinguish the first object with three edges from the others. However, the two remaining objects cannot be differentiated solely based on their edge count, as both have the same number of edges. To distinguish between them, it is necessary to analyze the labels of their edges more closely. A detailed examination reveals a subtle difference: one of the objects has edge labels N, N, T, T, T, while the other has edge labels N, N, N, N, T. This slight variation in edge labeling provides a reliable way to differentiate between the two objects. Additionally Figure 4.3(h), there are two nodes, 78 and 885, that have the same number of edges, and their edge labels are identical. In this case, the objects can only be distinguished by examining their labels.

(a) $G_1(1)$.(b) $G_1(2)$.(c) $G_1(3)$.(d) $G_1(4)$.(e) $G_1(5)$.(f) $G_1(6)$.(g) $G_1(7)$.(h) SPM G_1^* .Figure 4.3. Place 1 - Incremental construction of G_1 .

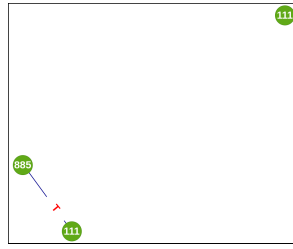
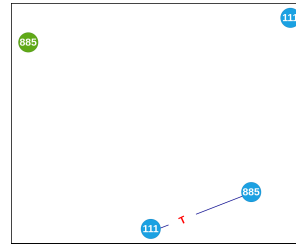
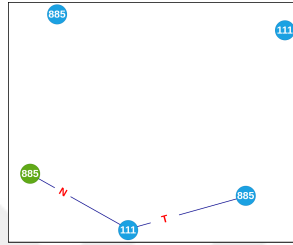
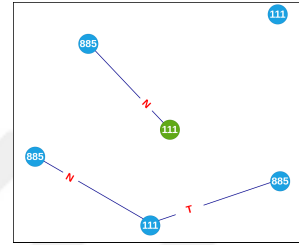
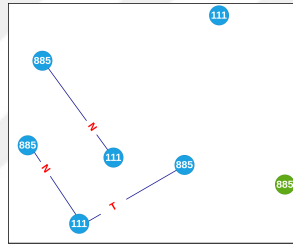
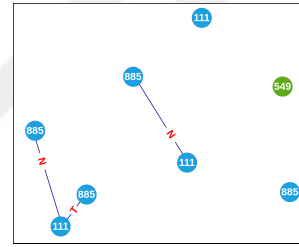
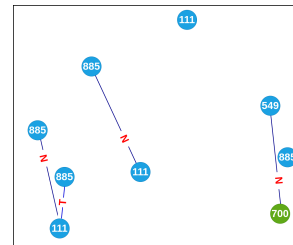
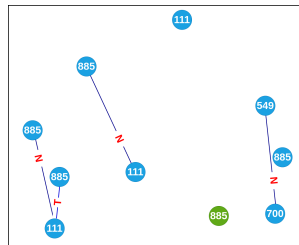
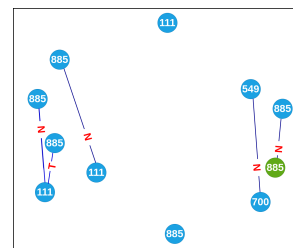
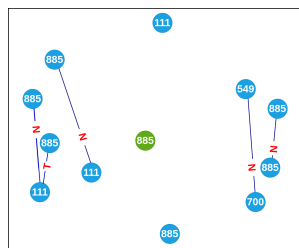
The evolution of SPM G_2 of place 2 as seen in Figure 4.4 occurs as seen in Figure 4.5. The initial graph of G_2 given in Figure 4.5(a) has three nodes: node 885, node 111, and node 111. These correspond to object 29, object 32, and object 11 in Figure 4.4(a) respectively. The edge label was determined as 'touching' in the graph and this relation can be observed in Figure 4.4(a). Another neighbor of node 885 is node 885 which is object 31 in the place figure. Although object 28 is another neighbor of node 885, since its size is very large, it was not accepted as an object candidate. It is observed that, despite the presence of different objects, the labels are predominantly of two types. This indicates that the recognition module is not working properly.

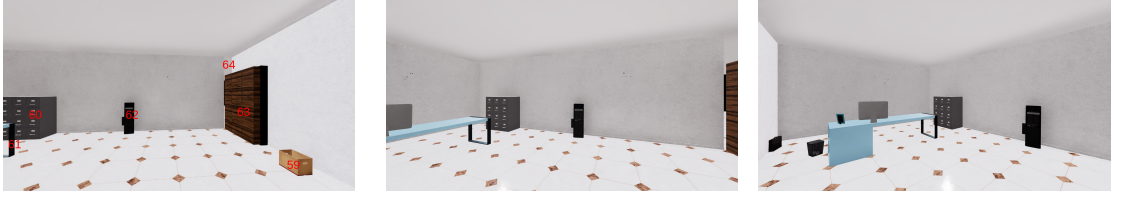
In the next place 3, sample route scenes are shown in Figure 4.6(a)-Figure 4.6(c). Incremental construction of G_3 evolves as seen in Figure 4.7. The first graph of place 3 given in Figure 4.7(a) has two nodes, node 918 and node 769, and the edge label between them is 'near'. These two nodes correspond to object 60 and object 61 in Figure 4.6 and we can observe that these two objects are indeed close to each other. Although there are some other objects in the figure that have a neighboring relationship, in the final graph given in Figure 4.7(e) there is just one edge. It is observed that the number of edges in the resulting SPM is less compared to the number of relationships depicted in Figures 4.6(a), 4.6(b), and 4.6(c). This discrepancy may be due to the object candidate recognition process. Specifically, the number of edges may be low because some object candidates that could establish these edges were not recognized and, therefore, not added to the graph.



(a) Place 2 - Onset. (b) Place 2 - After 30 frames. (c) Place 2 - After 67 frames.

Figure 4.4. Isaac Sim SPM Experiments: (a) place 2 - onset, (b) place 2 after 30 frames, and (c) place 2 after 67 frames.

(a) $G_2(1)$.(b) $G_2(2)$.(c) $G_2(3)$.(d) $G_2(4)$.(e) $G_2(5)$.(f) $G_2(6)$.(g) $G_2(7)$.(h) $G_2(8)$.(i) $G_2(9)$.(j) SPM G_2^* .Figure 4.5. Place 2 - Incremental construction of G_2 .



(a) Place 3 - Onset

(b) Place 3 - After 70 frames

(c) Place 3 - After 123 frames.

Figure 4.6. Isaac Sim SPM Experiments: (a) place 3 - onset, (b) place 3 after 70 frames, and (c) place 3 after 123 frames.

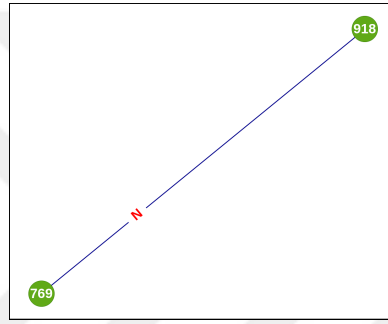
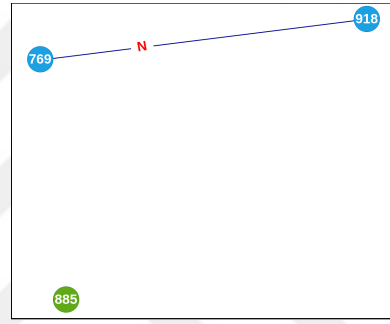
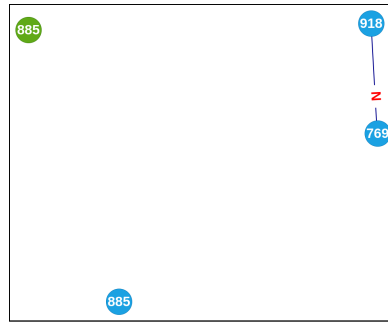
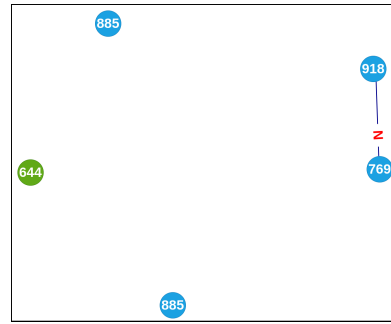
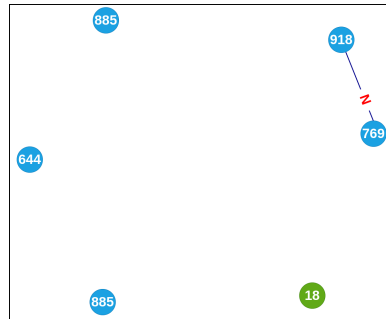
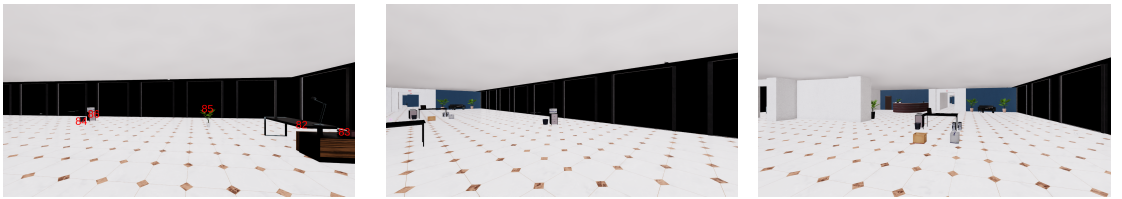
(a) $G_3(1)$.(b) $G_3(2)$.(c) $G_3(3)$.(d) $G_3(4)$.(e) SPM G_3^* .

Figure 4.7. Place 3 - Incremental construction of G_3 .

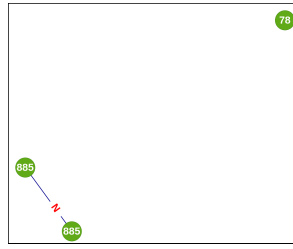
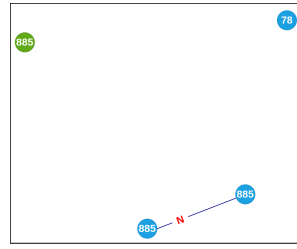
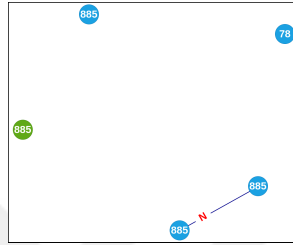
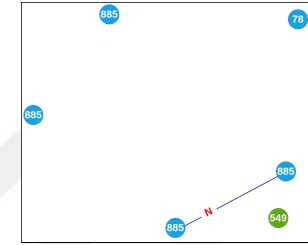
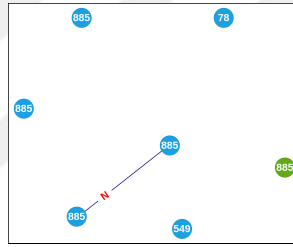
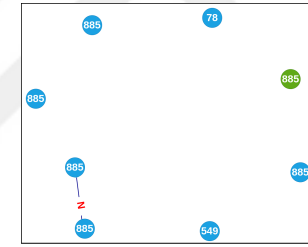
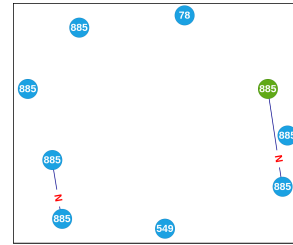
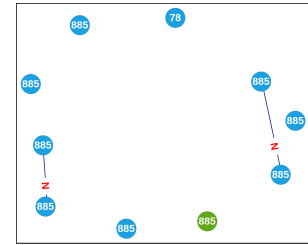
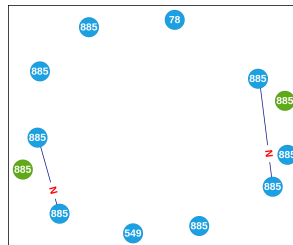
Finally, during the initial visit to place 4, the camera follows a specific path, which is depicted in Figure 4.8(a) through Figure 4.8(c). The initial graph, as shown in Figure 4.9(a), begins with the addition of nodes 78, 885, and 885. Among these, an edge labeled 'near' connects the two nodes with label 885, indicating their spatial relationship within the place. In Figure 4.8(a), the corresponding objects are identified as object 84, object 86, and object 85, which are mapped to nodes 885, 885, and 78, respectively. A closer examination of Figure 4.8(a) reveals that object 84 and object 86 share a 'near' neighboring relationship. This indicates that the neighboring relationships between these objects are accurately reflected in the graph, preserving their neighborhood connections. Similarly, in Figure 4.8(a), object 82 and object 83 also exhibit a 'near' neighboring relationship. However, in Figure 4.9, no additional edges are observed until $G_4(7)$ in Figure 4.9(g). This suggests that no relationship between object 82 and object 83 was established, as these objects were seen at the beginning while $G_4(7)$ was constructed later when object 82 and object 83 were no longer visible. Also, despite the presence of numerous objects that are 'near' to each other, the final graph G_4^* , shown in Figure 4.9(i), contains only two observed edges.

Furthermore, as seen in Figure 4.9, label 885 appears frequently and dominates the object labels. This dominance potentially diminishes the role of unique node labels in contributing to reliable place recognition, as multiple objects share the same label, complicating their differentiation.



(a) Place 4 - Onset. (b) Place 4 - After 145 frames. (c) Place 4 - After 240 frames.

Figure 4.8. Isaac Sim SPM Experiments: (a) place 4 - onset, (b) place 4 after 145 frames, and (c) place 4 after 240 frames.

(a) $G_4(1)$.(b) $G_4(2)$.(c) $G_4(3)$.(d) $G_4(4)$.(e) $G_4(5)$.(f) $G_4(6)$.(g) $G_4(7)$.(h) $G_4(8)$.(i) SPM G_4^* .Figure 4.9. Place 4 - Incremental construction of G_4 .

4.5.2. SPM-Based Place Recognition

These experiments aim to evaluate the recognition performance of the SPMs. The evaluation is based on pairwise GED values. The results are given in Table 4.2. As expected, self-similarity is zero.

Table 4.2. Recognition performance of SPMs.

SPM	G_1^*	G_2^*	G_3^*	G_4^*
G_1^*	0.0	40.1	38.7	43.9
G_2^*	40.1	0.0	19.7	19.1
G_3^*	38.7	19.7	0.0	17.2
G_4^*	37.2	12.7	10.2	0.0

We also consider the similarity of an incrementally constructed SPM. This study aims to determine whether a partially built SPM can be used to recognize a place as it evolves. For the ablation study, the similarity of the evolving SPM to its final form is also computed.

For place 1, the results are given in Table 4.3. It is observed that since the final graph G_1^* has many objects and relations, the dissimilarity between the earlier steps of the graph G_1 is very high when we compare it with the others. However, the dissimilarity increases over time. It is observed that after the fourth iteration, there is a significant discrepancy between similarity to ground truth similarity and similarity to others.

For place 2, the results are given in Table 4.4. These results are similar to those in Table 4.3. In the initial stages of the construction process, the dissimilarity with G_2 is lower than that with G_1 as G_2 contains fewer edges. It is observed that the dissimilarity with G_2 decreases over time and reaches zero dissimilarity.

Table 4.3. Evolving G_1 similarity. Note $G_1^* = G_1(8)$.

SPM	$G_1(1)$	$G_1(2)$	$G_1(3)$	$G_1(4)$	$G_1(5)$	$G_1(6)$	$G_1(7)$	$G_1(8)$
G_1^*	43.6	37.5	31.0	18.0	17.0	2.0	1.0	0.0
G_2^*	25.3	19.6	24.9	31.4	30.4	41.6	40.6	40.1
G_3^*	7.6	7.9	13.7	26.0	25.0	37.7	38.2	38.7
G_4^*	23.3	23.4	23.7	35.7	34.7	45.9	44.9	43.9

Table 4.4. Evolving G_2 similarity. Note $G_2^* = G_2(10)$.

Graphs	$G_2(1)$	$G_2(2)$	$G_2(3)$	$G_2(4)$	$G_2(5)$	$G_2(6)$	$G_2(7)$	$G_2(8)$	$G_2(9)$	$G_2(10)$
G_1^*	40.3	39.3	36.1	33.4	32.4	31.4	34.7	35.1	39.6	40.1
G_2^*	21.5	20.5	16.0	12.0	11.0	10.3	6.0	5.0	1.0	0.0
G_3^*	4.9	3.9	5.9	8.7	9.2	9.7	14.2	14.7	19.2	19.7
G_4^*	20.3	19.3	20.8	17.6	16.6	15.6	18.4	17.4	20.1	19.1

For place 3, the results are given in Table 4.5. Table 4.5 shows that since place 2 already has fewer notes and edges than the others, its dissimilarity with itself starts from a much smaller value and reaches zero over time and since the number of nodes and edges of place 1 and place 2 are much higher than place 3, it is seen that their dissimilarity is quite high. Table 4.6 presents results for place 4. According to Table 4.6, when the robot returns to place 4, it seems to recognize place 4 over time.

Table 4.5. Evolving G_3 similarity. Note $G_3^* = G_3(5)$.

Graphs	$G_3(1)$	$G_3(2)$	$G_3(3)$	$G_3(4)$	$G_3(5)$
G_1^*	41.6	40.6	39.6	38.9	38.7
G_2^*	22.6	21.6	20.6	19.9	19.7
G_3^*	4.0	3.0	2.0	1.0	0.0
G_4^*	20.6	19.6	18.6	17.9	17.2

Table 4.6. Evolving G_4 similarity. Note $G_4^* = G_4(9)$.

Graphs	$G_4(1)$	$G_4(2)$	$G_4(3)$	$G_4(4)$	$G_4(5)$	$G_4(6)$	$G_4(7)$	$G_4(8)$	$G_4(9)$
G_1^*	40.6	39.6	38.6	37.8	37.6	37.4	34.2	35.2	37.2
G_2^*	21.3	20.3	19.3	18.8	18.3	17.8	13.6	13.1	12.7
G_3^*	3.9	2.9	1.9	1.2	2.19	3.2	7.2	8.2	10.2
G_4^*	12.0	11.0	10.0	9.0	8.0	7.0	3.0	2.0	0.0

4.5.3. SPM-Based Place Recognition - Subsequent Visits

The next set of experiments evaluate the recognition performance of a newly constructed SMP in future visits. Future visits can be categorized into two groups depending on whether the navigation route in the place is similar or very different from that of the first visit. Two routes are considered to be similar if their starting locations are nearby and route locations with similar sequence order are also nearby.

First, we consider similar routes. This will occur in scenarios like entering a room through a door. The traversed routes will tend to be similar.

In revisiting places, first routes similar to the first visits are considered. For example, for the first place, consider a route with scenes as shown in Figure 4.10(d)-Figure 4.10(f). Both the starting point and route path are similar to those of SMP G_1^* .

The construction of SPM $G(k)$ evolves as shown in Figure 4.11. The final SMP G is that given by Figure 4.11(i). It is observed that its evolution resembles that G_1^* of the first visit to the same place. The number of edges is identical, the number of nodes is comparable, and some pairs share the same edge labels. This is attributed to the fact that the route traversed in the second visit is similar to that of the first visit.

The recognition performance of the incremental SPM $G(k)$ is given in Table 4.7. As expected, as it is being built, its dissimilarity with SPMs of other places increase,

while its dissimilarity with SPM G_1^* decreases over time.

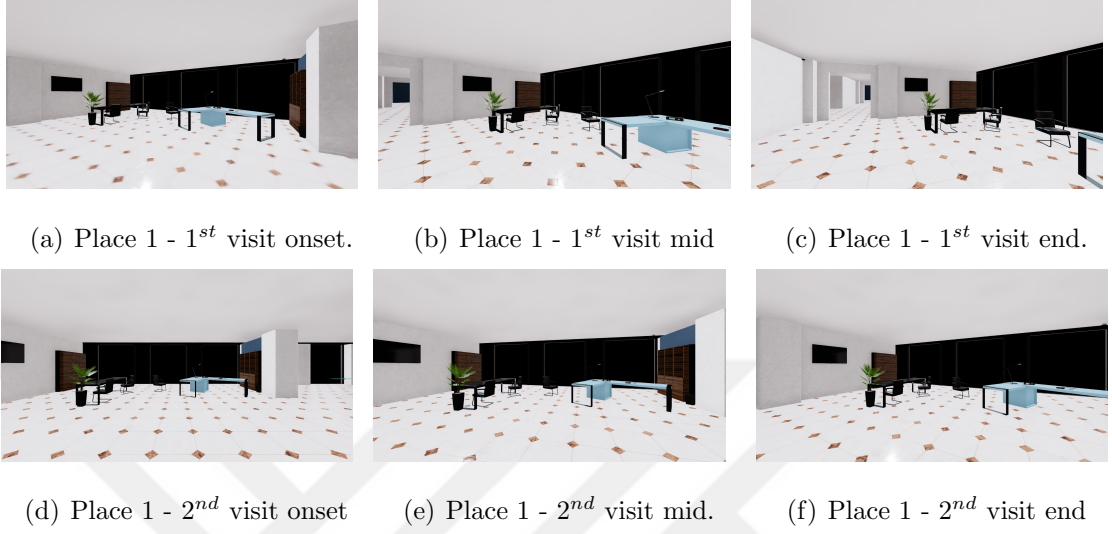


Figure 4.10. Revisiting place 1 with a route similar to that of SPM G_1^* : (a) place 1, first visit onset, (b) place 1, first visit mid (c) end of the first visit, (d) Place 1, second visit onset, (e) place 1, second visit mid and (f) end of the second visit.

Table 4.7. Evolving SPM G similarity in the second visit of place 1.

Graphs	$G(1)$	$G(2)$	$G(3)$	$G(4)$	$G(5)$	$G(6)$	$G(7)$	$G(8)$	$G(9)$
G_1^*	41.3	40.6	37.9	37.2	30.9	27.2	15.3	14.6	15.1
G_2^*	22.3	21.6	17.9	16.9	16.7	20.8	32.0	31.3	30.6
G_3^*	4.6	3.9	6.9	6.2	12.8	22.8	35.8	36.3	36.8
G_4^*	13.6	12.9	9.2	8.5	15.3	23.8	35.6	34.9	34.2

Following, we consider cases in which subsequent visits follow different routes. In this case, the second visit route in each place has a different starting point and viewpoint. such, the respective paths differ.

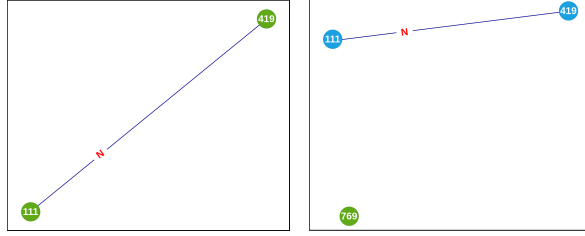
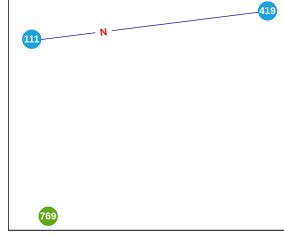
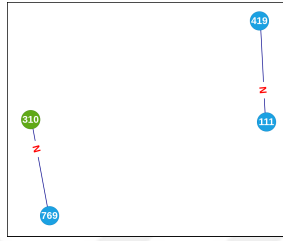
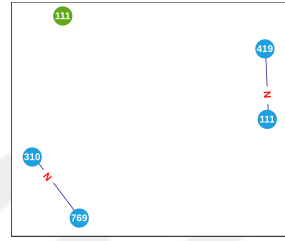
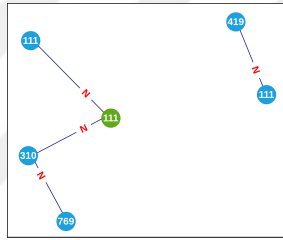
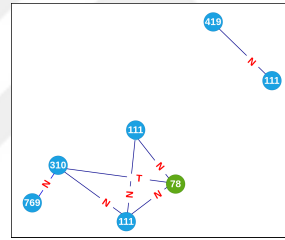
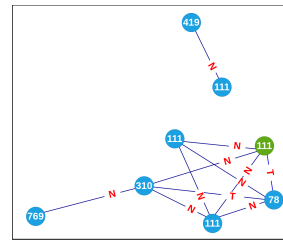
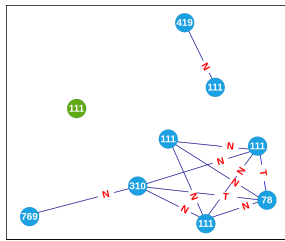
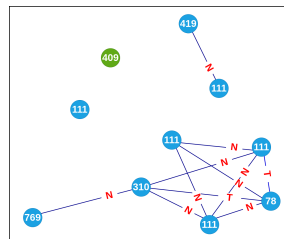
(a) $G(1)$.(b) $G(2)$.(c) $G(3)$.(d) $G(4)$.(e) $G(5)$.(f) $G(6)$.(g) $G(7)$.(h) $G(8)$.(i) $G(9)$.

Figure 4.11. Place 1 revisit - Incremental construction of SPM G along a route similar to that of G_1^* .

In revisiting place 1, sample scenes along the route are seen in Figure 4.12(d)-Figure 4.12(f). The evolution of the respective graph $G(k)$ is seen in Figure 4.13. When examining Figure 4.13, since the order in which objects appear changes, the sequence in which the graphs are constructed also changes. It is evident that the order of node and edge additions has shifted as expected. However, the final graph in Figure 4.13(h) does not align with expectations. When compared with G_1^* , the number of edges for these places differs significantly. This discrepancy may be due to the fact that only recognized object candidates are added to the graph. Consequently, some objects with potential neighborhood relationships to others might not have been recognized.

The similarity of evolving SPM G in revisiting place 1 is given in Table 4.8. It is observed that initially, the dissimilarity of SPM $G(k)$ with a graph of G_3^* was low due to the smaller number of nodes and edges in the final graph of place 3. However, over time, the dissimilarity increased. In contrast, when $G(k)$ is compared to G_1^* 's, the dissimilarity is high even though it decreases over time. In this case, it is found to be most similar to SPM G_2^* .

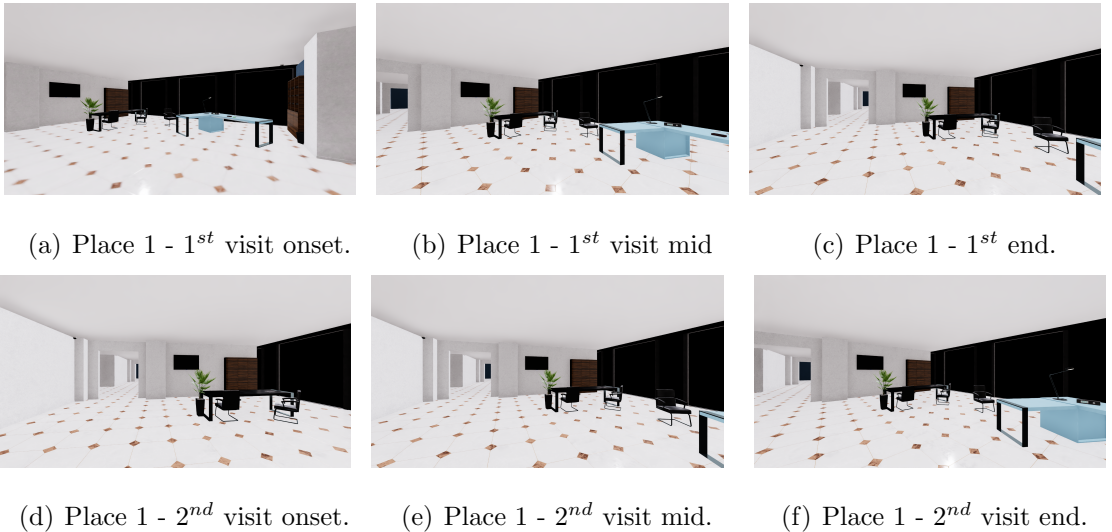


Figure 4.12. Revisiting place 1 with a different route: (a) place 1, first visit onset, (b) place 1, first visit mid, (c) end of first visit, (d) Place 1, second visit onset, (e) place 1, second visit mid and (f) end of the second visit.

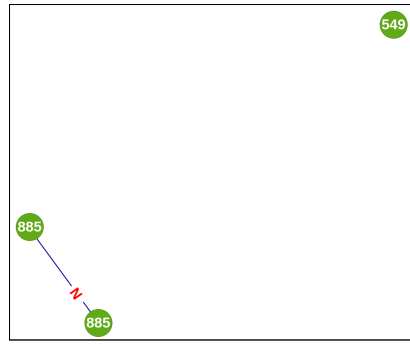
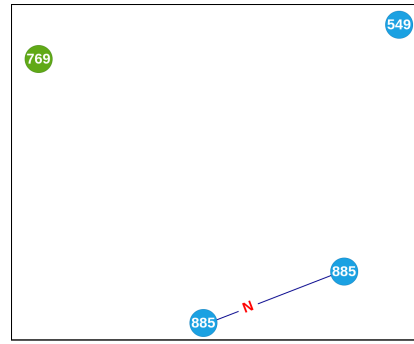
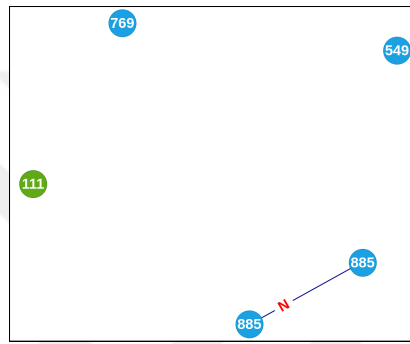
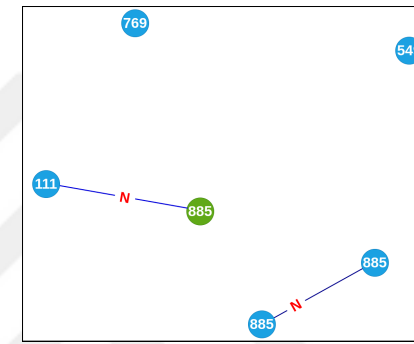
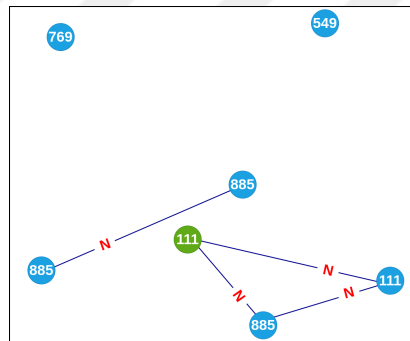
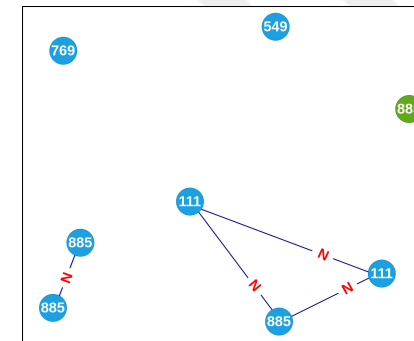
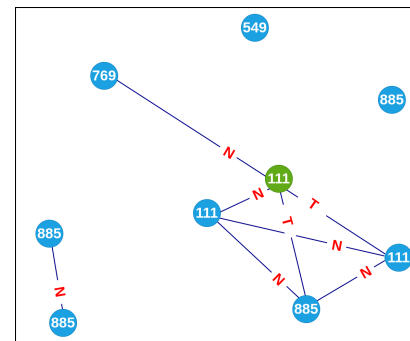
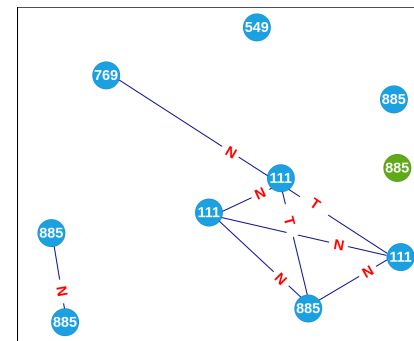
(a) $G(1)$.(b) $G(2)$.(c) $G(3)$.(d) $G(4)$.(e) $G(5)$.(f) $G(6)$.(g) $G(7)$.(h) SPM G .Figure 4.13. Place 1 revisit - Incremental construction of SPM G .

Table 4.8. Evolving SPM G similarity in revisiting place 1.

SPM	$G(1)$	$G(2)$	$G(3)$	$G(4)$	$G(5)$	$G(6)$	$G(7)$	$G(8)$
G_1^*	40.3	39.6	38.9	35.9	29.4	28.4	21.6	22.1
G_2^*	21.3	20.6	19.6	15.6	16.9	15.9	20.9	19.9
G_3^*	3.9	3.2	2.5	5.2	12.5	12.7	26.2	26.7
G_4^*	12.0	11.3	10.6	6.6	13.4	12.4	24.2	23.2

The revisit route of place 2 is as seen in Figure 4.14(d)-4.14(f). The respective SPM G evolves as seen in Figure 4.15. Table 4.9 presents the recognition performance of the evolving $G(k)$ as it is being built during a second visit of place 2.

According to Table 4.9, initially, G was more similar to G_3^* , but as the graph became more complete, the dissimilarity between G and G_3^* increased, while the dissimilarity between G and G_2^* decreased. The dissimilarity between SPM G and G_2^* was lower compared to the others by the end of the process, it was noted that they did not have sufficient similarity for recognition.

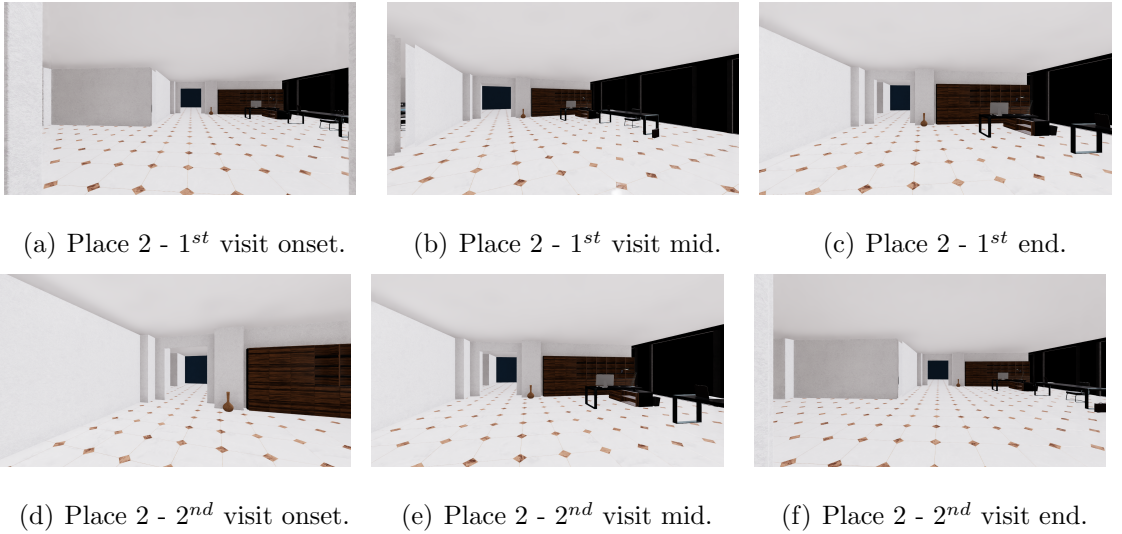


Figure 4.14. Revisiting place 2 with different a route: (a) place 2, first visit onset, (b) place 2, first visit mid, (c) end of first visit, (d) place 2, second visit onset, (e) place 2, second visit mid and (f) end of the second visit.

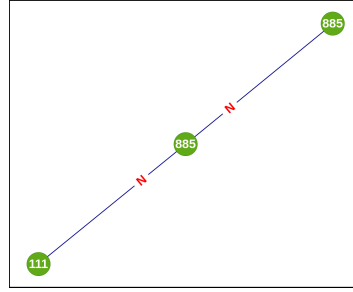
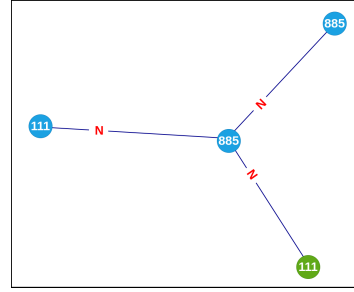
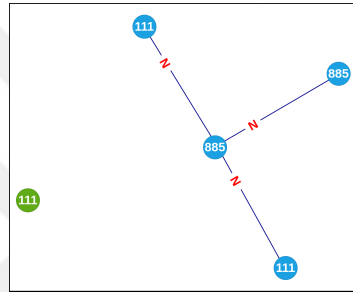
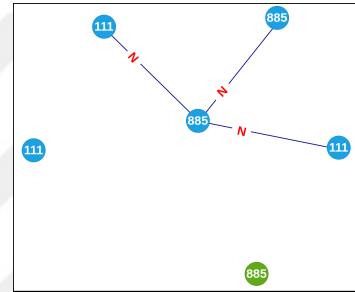
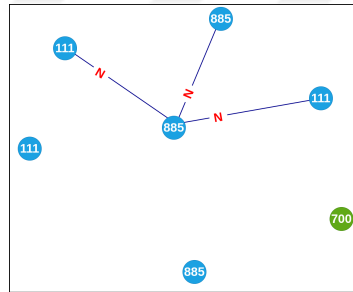
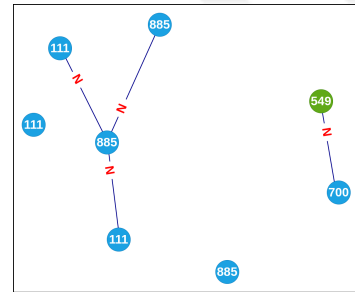
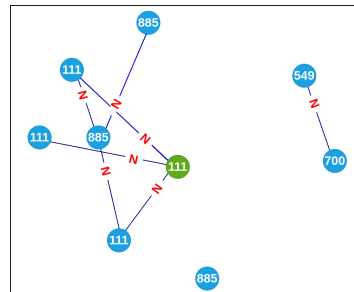
(a) $G(1)$.(b) $G(2)$.(c) $G(3)$.(d) $G(4)$.(e) $G(5)$.(f) $G(6)$.(g) Final SPM G of place 2.Figure 4.15. Place 2 revisit - Incremental construction of SPM G .

Table 4.9. Evolving SPM G similarity in revisiting place 2.

SPM	$G(1)$	$G(2)$	$G(3)$	$G(4)$	$G(5)$	$G(6)$	$G(7)$
G_1^*	37.8	33.8	33.09	32.1	31.4	28.7	25.2
G_2^*	19.6	22.1	21.1	20.1	19.4	15.1	18.6
G_3^*	7.6	10.4	9.7	8.7	9.2	13.2	23.7
G_4^*	15.8	18.1	17.4	16.4	15.7	13.0	22.29

Place 3 is revisited as seen in Figures 4.16(d)-4.16(f). The evolution of the respective SPM $G(k)$ is as shown in Figure 4.17. Its final graphs $G_6(7)$ is denoted as G_7 and given in Figures 4.17(h). In Figure 4.17(h), a completely different final graph is observed. When compared to G_3^* it is evident that new nodes were introduced into the graph. As a result of these additions, new edges were also formed.

Recognition performance based on the incremental SPG $G(k)$ when revisiting place 3 is given in Table 4.10. Since the final SPM G is significantly different from G_3^* , it is observed that their dissimilarity is very high.

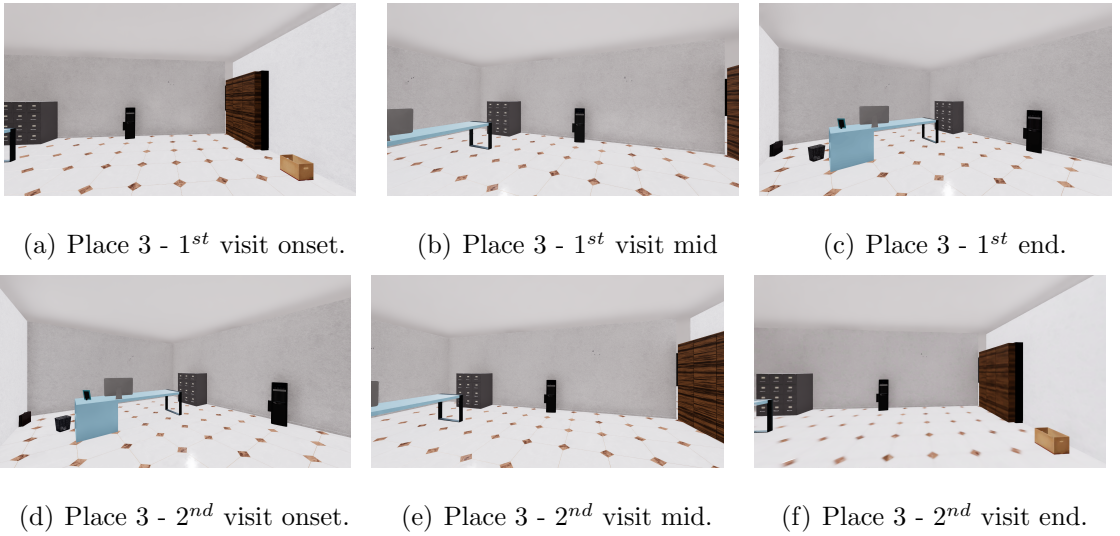


Figure 4.16. Revisiting place 3 with different a route: (a) place 3, first visit onset, (b) place 3, first visit mid, (c) end of first visit, (d) place 3, second visit onset, (e) place 3, second visit mid and (f) end of the second visit.

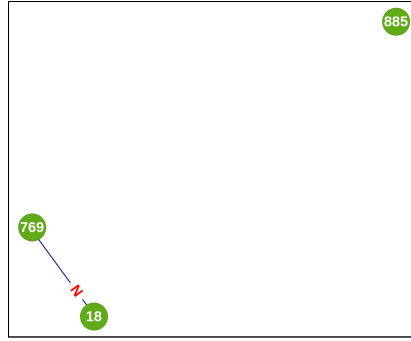
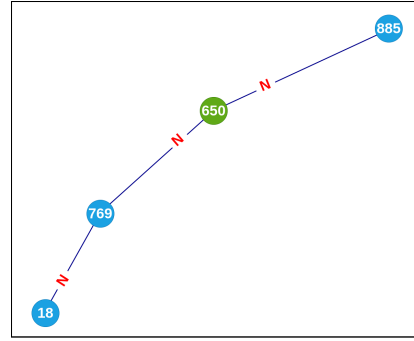
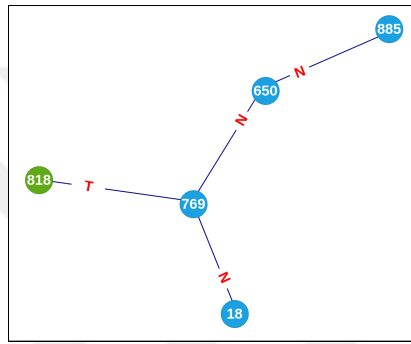
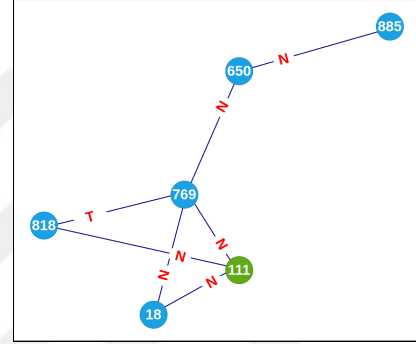
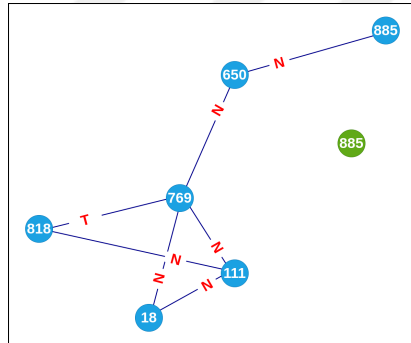
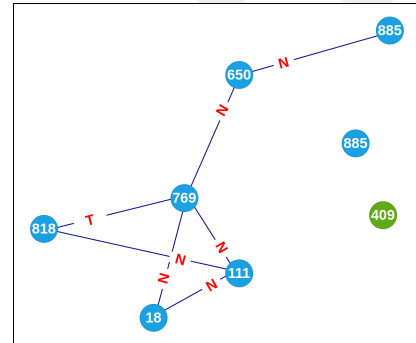
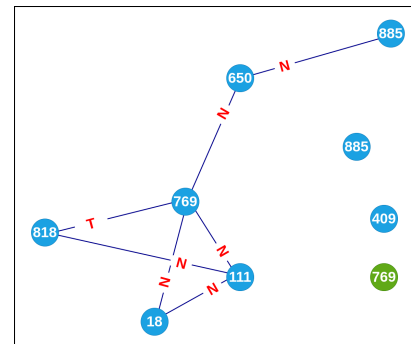
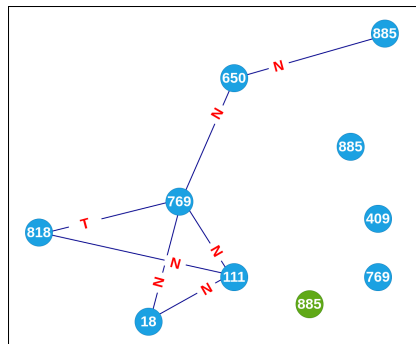
(a) $G(1)$.(b) $G(2)$.(c) $G(3)$.(d) $G(4)$.(e) $G(5)$.(f) $G(6)$.(g) $G(8)$.(h) Final SPM G of place 3.Figure 4.17. Place 3 revisit - Incremental construction of SPM G .

Table 4.10. Evolving SPM G similarity in place 3.

Graphs	$G(1)$	$G(2)$	$G(3)$	$G(4)$	$G(5)$	$G(6)$	$G(7)$	$G(8)$
G_1^*	40.6	34.9	31.7	21.5	20.5	19.8	19.1	19.8
G_2^*	21.6	21.4	18.2	27.2	26.2	25.5	24.8	24.3
G_3^*	3.3	10.1	12.89	25.0	24.0	22.6	22.6	23.7
G_4^*	12.6	12.89	15.7	21.9	22.1	23.3	23.2	21.59

Place 4 is revisited as seen in Figures 4.18(d)-4.18(f). The evolution of the respective SPM G is seen in 4.19. Similar to other revisited places, G is entirely different from G_4^* due to the non-robust nature of the graph for place 4 across different visits. However, unlike G_4^* , the neighboring relationships seen in in Figures 4.18(d)-4.18(f) are preserved.

Recognition performance based on the incremental SPG $G(k)$ when revisiting place 4 is given in Table 4.11. Recognition performance is also poor since the similarity of SPM G with G_4^* is observed to decrease over time.

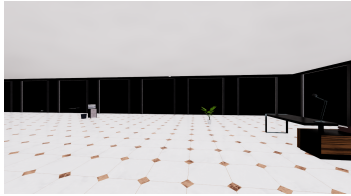
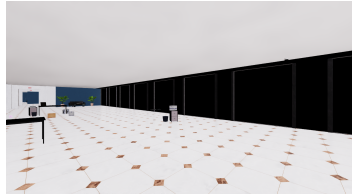
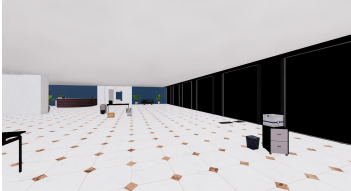
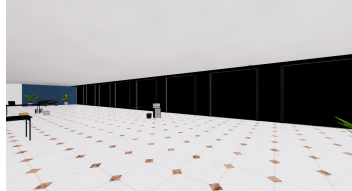
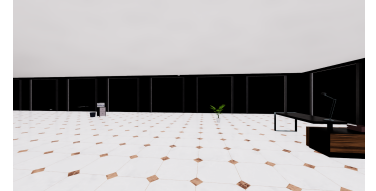
(a) Place 4 - 1st visit onset(b) Place 4 - 1st visit mid(c) Place 4 - 1st end.(d) Place 4 - 2nd visit onset..(e) Place 4 - 2nd visit mid.(f) Place 4 - 2nd visit end.

Figure 4.18. Revisiting place 4 with different a route: (a) place 4, first visit onset, (b) place 4, first visit mid, (c) end of first visit, (d) place 4, second visit onset, (e) place 4, second visit mid and (f) end of the second visit.

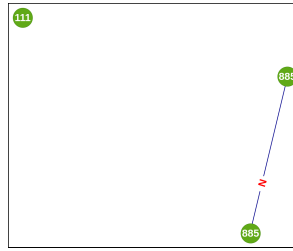
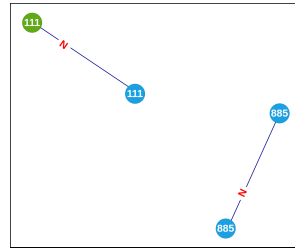
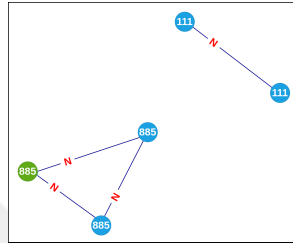
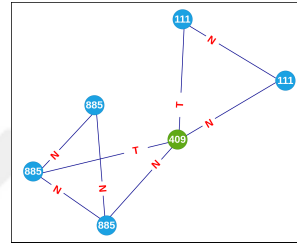
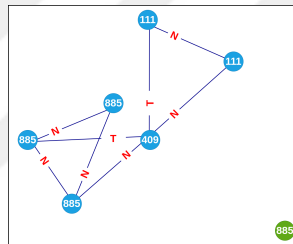
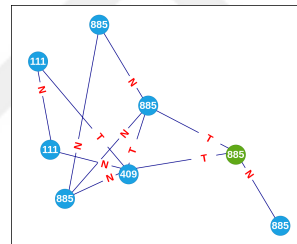
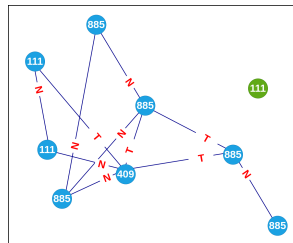
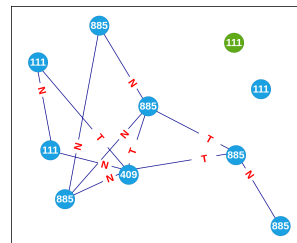
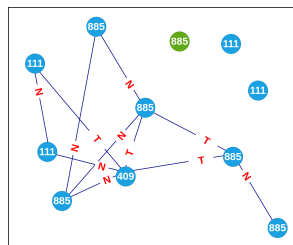
(a) $G(1)$.(b) $G(2)$.(c) $G(3)$.(d) $G(4)$.(e) $G(5)$.(f) $G(6)$.(g) $G(7)$.(h) $G(8)$.(i) Final SPM G Figure 4.19. Place 4 revisit - Incremental construction of SPM G .

Table 4.11. Evolving SPM G similarity in place 4.

SPM	$G(1)$	$G(2)$	$G(3)$	$G(4)$	$G(5)$	$G(6)$	$G(7)$	$G(8)$	$G(9)$
$\mathcal{G}_1^*$	40.59	37.3	31.1	19.0	18.0	26.0	25.3	26.1	26.6
$\mathcal{G}_2^*$	21.0	17.3	18.6	30.1	29.1	31.9	30.9	30.2	29.5
$\mathcal{G}_3^*$	3.9	6.6	12.39	25.2	25.7	35.69	36.19	36.69	37.19
$\mathcal{G}_4^*$	12.3	8.6	15.1	27.9	26.9	35.4	34.7	34.0	33.0

Recognition performance of the SPMs constructed in these visits are evaluated considering those of the first visits. The results are given in Table 4.12. Recall that in this case, the route in each place starts from a completely opposite location and proceeds in a completely opposite direction. In these cases, SMP recognition performance is observed to be relatively poor.

Table 4.12. Recognition performance of revisit SPMs with different routes.

SPM		Second Visit			
		G_1	G_2	G_3	G_4
First visit	G_1^*	22.1	25.2	19.8	26.6
	G_2^*	19.9	18.6	24.3	29.5
	G_3^*	26.7	23.7	23.7	37.19
	G_4^*	23.2	22.29	21.59	33.0

5. CONCLUSION

This thesis presents a comprehensive framework for building semantic place maps. A semantic place map is an internal representation that incorporates the major constituent objects and their spatial relations within each place. Differing from most related work, in this thesis, it is defined by a labeled graph whose nodes correspond to point cloud objects and whose edges encode proximity - touching or near - relations among these objects.

In the first part, candidate point cloud objects are discovered in each frame of the incoming RGB-D stream. This is achieved through processing RGB images using the YOLO network and then constructing the corresponding point cloud objects. A series of refinement processes are applied to in order to alleviate segmentation problems and noisy data.

In the second part of the thesis, point cloud objects are detected via tracking the candidate point cloud objects over the RGB-D stream. Tracking is based on their positions, shapes, colors, RGB keypoints, and neighboring objects. The first two properties are derived from their respective minimum volume enclosing ellipsoids. This representation is also used to identify nearby objects. For the evaluation of tracking, two indoor video datasets have been developed - due to the lack of such annotated datasets. In addition, custom metrics have been designed to evaluate tracking performance - since most metrics in the literature require an integrated approach to detection and tracking. According to these metrics, the tracking algorithm performed very well on real-world data and achieved significantly good results on the synthetic data generated by Isaac Sim.

The last part presents the formulation and usage of semantic place map SPM. In this approach, the tracking status of all candidate point cloud objects is recorded in a working candidate objects memory (WCOM) as separate tracks. Once a track has

accumulated sufficient knowledge, it is subjected to a recognition process. Recognized objects are then added to the incrementally constructed SPM along with their relations which are labeled either as ‘touching’ or ‘near’. The constructed semantic place maps are then associated with their respective places - as they encode the semantic content. Upon subsequent visits of place, the constructed semantic place map can be related to the semantic place map that was previously constructed. This is done based on a graph edit distance metric. Through an experimental evaluation, both the construction of the SPMs and their recognition performance have been studied. The results indicate two factors to be affecting recognition performance. The first pertains to the correct recognition of the point cloud objects. Therefore, improving object recognition performance can significantly enhance place recognition accuracy. The second factor pertains to the place traversal route. In particular, SPM-based recognition performance deteriorates as traversal routes differ from those of the first visits associated with the learned SPMs. This may be in part attributed to poor object recognition and in part to the graph matching metric GED used.

For future work, in the tracking process, the weights of the dissimilarity function, which directly impact tracking performance, need to be made adaptive. This would allow the function to assign greater importance to the most discriminative features on a frame-by-frame basis. For place recognition, the object candidate learning and recognition processes require further improvement to enhance semantic place recognition performance.

REFERENCES

1. Rensink, R. A., “The Dynamic Representation of Scenes”, *Visual Cognition*, Vol. 7, No. 1-3, pp. 17–42, 2000.
2. Meger, D., P.-E. Forssén, K. Lai, S. Helmer, S. McCann, T. Southey, M. Baumann, J. J. Little and D. G. Lowe, “Curious george: An Attentive Semantic Robot”, *Robotics and Autonomous Systems*, Vol. 56, No. 6, pp. 503–511, 2008.
3. Pillai, S. and J. Leonard, “Monocular SLAM Supported Object Recognition”, *arXiv preprint arXiv:1506.01732*, 2015.
4. Horbert, E., G. M. García, S. Frintrop and B. Leibe, “Sequence-level Object Candidates Based on Saliency for Generic Object Recognition on Mobile Systems”, *2015 IEEE International Conference on Robotics and Automation*, Seattle, WA, USA, pp. 127–134, 2015.
5. Held, D., D. Guillory, B. Rebsamen, S. Thrun and S. Savarese, “A Probabilistic Framework for Real-time 3D Segmentation using Spatial, Temporal, and Semantic Cues”, *Robotics: Science and Systems*, Ann Arbor, Michigan, USA, 2016.
6. Teichman, A., J. Levinson and S. Thrun, “Towards 3D Object Recognition via Classification of Arbitrary Object Tracks”, *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 4034–4041, 2011.
7. Teichman, A. and S. Thrun, “Tracking-Based Semi-Supervised Learning”, *The International Journal of Robotics Research*, Vol. 31, No. 7, pp. 804–818, 2012.
8. Ošep, A., W. Mehner, P. Voigtlaender and B. Leibe, “Track, Then Decide: Category-Agnostic Vision-Based Multi-Object Tracking”, *IEEE International Conference on Robotics and Automation*, Brisbane, QLD, Australia, pp. 3494–3501, 2018.

9. Lowry, S., N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke and M. J. Milford, “Visual Place Recognition: A Survey”, *IEEE Transactions on Robotics*, Vol. 32, No. 1, pp. 1–19, 2016.
10. Tuytelaars, T., C. H. Lampert, M. B. Blaschko and W. Buntine, “Unsupervised Object Discovery: A Comparison”, *International Journal of Computer Vision*, Vol. 88, No. 2, pp. 284–302, 2010.
11. Hosang, J., R. Benenson and B. Schiele, “How Good are Detection Proposals, Really?”, *arXiv preprint arXiv:1406.6962*, 2014.
12. Driver, J., G. Davis, C. Russell, M. Turatto and E. Freeman, “Segmentation, Attention, and Phenomenal Visual Objects”, *Cognition*, Vol. 80, No. 1–2, pp. 61 – 95, 2001.
13. Martín García, G., T. Werner and S. Frintrop, “Attentional Scene-Exploration and Object Discovery in Image and RGB-D Data”, *KI - Künstliche Intelligenz*, Vol. 29, No. 1, pp. 75–81, 2015.
14. Marfil, R., A. J. Palomino and A. Bandera, “Combining Segmentation and Attention: A New Foveal Attention Model”, *Frontiers in Computational Neuroscience*, Vol. 8, p. 96, 2014.
15. Zhao, X., W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang and J. Wang, “Fast Segment Anything”, *arXiv preprint arXiv:2306.12156*, 2023.
16. Lin, T.-Y., M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick and P. Dollár, “Microsoft COCO: Common Objects in Context”, *arXiv preprint arXiv:1405.0312*, 2015.
17. He, Y., H. Yu, X. Liu, Z. Yang, W. Sun, S. Anwar and A. Mian, “Deep Learning Based 3D Segmentation: A Survey”, *arXiv preprint arXiv:2103.05423*, 2024.

18. Rusu, R. B., Z. C. Marton, N. Blodow, M. Dolha and M. Beetz, “Towards 3D Point Cloud Based Object Maps for Household Environments”, *Robotics and Autonomous Systems*, Vol. 56, No. 11, pp. 927–941, 2008.
19. Rusu, R. B. and S. Cousins, “3D is Here: Point Cloud Library (PCL)”, *IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 1–4, 2011.
20. Kuhn, H. W., “The Hungarian Method for the Assignment Problem”, *Naval Research Logistics Quarterly*, Vol. 2, No. 1–2, pp. 83–97, 1955.
21. Bewley, A., Z. Ge, L. Ott, F. Ramos and B. Upcroft, “Simple Online and Realtime Tracking”, *2016 IEEE International Conference on Image Processing*, Phoenix, AZ, USA, pp. 3464–3468, 2016.
22. Berclaz, J., F. Fleuret, E. Turetken and P. Fua, “Multiple Object Tracking Using K-Shortest Paths Optimization”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 33, No. 9, pp. 1806–1819, 2011.
23. Butt, A. A. and R. T. Collins, “Multi-target Tracking by Lagrangian Relaxation to Min-cost Network Flow”, *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, pp. 1846–1853, 2013.
24. Milan, A., S. Roth and K. Schindler, “Continuous Energy Minimization for Multitarget Tracking”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 36, No. 1, pp. 58–72, 2014.
25. Xiang, Y., A. Alahi and S. Savarese, “Learning to Track: Online Multi-object Tracking by Decision Making”, *2015 IEEE International Conference on Computer Vision*, Santiago, Chile, pp. 4705–4713, 2015.
26. Li, Y., C. Huang and R. Nevatia, “Learning to Associate: HybridBoosted Multi-Target Tracker for Crowded Scene”, *2009 IEEE Conference on Computer Vision*

- and Pattern Recognition*, Miami, FL, USA, pp. 2953–2960, 2009.
27. Xu, Y., L. Qin, G. Li and Q. Huang, “Online Discriminative Structured Output SVM Learning for Multi-Target Tracking”, *IEEE Signal Processing Letters*, Vol. 21, No. 2, pp. 190–194, 2014.
 28. Song, X., J. Cui, H. Zha and H. Zhao, “Vision-Based Multiple Interacting Targets Tracking via On-Line Supervised Learning”, *Computer Vision – ECCV 2008*, Marseille, France, pp. 642–655, 2008.
 29. Wojke, N., A. Bewley and D. Paulus, “Simple Online and Realtime Tracking with a Deep Association Metric”, *2017 IEEE International Conference on Image Processing*, Beijing, China, pp. 3645–3649, 2017.
 30. Zhang, Y., C. Wang, X. Wang, W. Zeng and W. Liu, “FairMOT: On the Fairness of Detection and Re-identification in Multiple Object Tracking”, *International Journal of Computer Vision*, Vol. 129, No. 11, p. 3069–3087, 2021.
 31. Li, B., W. Wu, Q. Wang, F. Zhang, J. Xing and J. Yan, “SiamRPN++: Evolution of Siamese Visual Tracking With Very Deep Networks”, *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, pp. 4277–4286, 2019.
 32. Demir, M., *Semantic Scene Analysis Through Visual Exploration and Learning*, Ph.D. Thesis, Bogazici University, 2014.
 33. Durukan, M., *Depth-Based Scene Mapping Through Spatio-Temporal Knowledge Integration*, Master’s Thesis, Bogazici University, 2021.
 34. Gilitschenski, I. and U. D. Hanebeck, “A Robust Computational Test for Overlap of Two Arbitrary-Dimensional Ellipsoids in Fault-Detection of Kalman Filters”, *2012 15th International Conference on Information Fusion*, Singapore, Singapore, pp. 396–401, 2012.

35. Lin, A. and S.-P. Han, “On the Distance between Two Ellipsoids”, *SIAM Journal on Optimization*, Vol. 13, No. 1, pp. 298–308, 2002.
36. Förstner, W. and B. Moonen, *A Metric for Covariance Matrices*, pp. 299–309, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.
37. Huamán, A., “OpenCV: Feature Matching with FLANN”, https://docs.opencv.org/4.x/d5/d6f/tutorial_feature_flann_matcher.html, accessed in December 2024.
38. Chiu, H.-K., A. Prioletti, J. Li and J. Bohg, “Probabilistic 3D Multi-Object Tracking for Autonomous Driving”, *arXiv preprint arXiv:2001.05673*, 2020.
39. Corporation, N., “NVIDIA Isaac Sim”, <https://developer.nvidia.com/isaac/sim>, accessed in November 2024.
40. Shen, Z., L. Kästner and J. Lambrecht, “Spatial Imagination with Semantic Cognition for Mobile Robots”, *2021 IEEE International Conference on Robotics and Automation*, Xi’an, China, pp. 1234–1240, 2021.
41. Cadena, C., L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J. J. Leonard, “Semantic Parsing for Priming Object Detection in Indoors RGB-D Scenes”, *arXiv preprint arXiv:1402.4177*, 2014.
42. Vasudevan, S., S. Gächter, V. H. Nguyen and R. Siegwart, “Cognitive Maps for Mobile Robots—An Object-Based Approach”, *Robotics and Autonomous Systems*, Vol. 55, No. 5, pp. 359–371, 2007.
43. Ranganathan, A. and F. Dellaert, “Semantic Modeling of Places using Objects”, *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, pp. 1–8, 2007.
44. Anand, A., H. S. Koppula, T. Joachims and A. Saxena, “Contextually Guided Semantic Labeling and Search for 3D Point Clouds”, *arXiv preprint arXiv:1111.5358*,

2011.

45. Yang, G., X. Wang, D. Zhu and Y. Li, “Real-time Dense 3D Semantic Mapping using RGB-D Camera”, *2023 42nd Chinese Control Conference*, Tianjin, China, pp. 4419–4424, 2023.
46. Jocher, G. and J. Qiu, “Ultralytics YOLO11”, <https://github.com/ultralytics/ultralytics>, accessed in November 2024.
47. Gao, X., B. Xiao, D. Tao and X. Li, “A Survey of Graph Edit Distance”, *Pattern Analysis and Applications*, Vol. 13, No. 1, pp. 113–129, 2010.
48. Hagberg, A. A., D. A. Schult and P. J. Swart, “Exploring Network Structure, Dynamics, and Function using NetworkX”, *Proceedings of the 7th Python in Science Conference*, Pasadena, CA, USA, pp. 11–15, 2008.
49. Abu-Aisheh, Z., R. Raveaux, J.-Y. Ramel and P. Martineau, “An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems”, *4th International Conference on Pattern Recognition Applications and Methods 2015*, Lisbon, Portugal, pp. 271–278, 2015.

APPENDIX A: ELLIPSOID

A.1. Ellipsoid Fitting: Khachiyan's Algorithm

Given a set of points $P' = \{p'_1, p'_2, \dots, p'_n\} \subset \mathbb{R}^3$, where each point $p'_i = (x_i, y_i, z_i)^\top \in \mathbb{R}^3$, the goal is to find the Minimum Volume Enclosing Ellipsoid (MVEE).

The matrix $P' \in \mathbb{R}^{3 \times n}$ containing the point coordinates is defined as

$$P' = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \end{bmatrix}. \quad (\text{A.1})$$

An extension of P to include a row of ones, creating $Q \in \mathbb{R}^{4 \times n}$,

$$Q = \begin{bmatrix} P' \\ \mathbf{1}_n^\top \end{bmatrix} \quad (\text{A.2})$$

where $\mathbf{1}_n$ is a row vector of ones. The weight vector $u \in \mathbb{R}^n$ is defined as

$$u = \frac{1}{n} \mathbf{1}_n \quad (\text{A.3})$$

where $\mathbf{1}_n$ is a column vector of ones. The matrix $\mathbf{X} \in \mathbb{R}^{4 \times 4}$ is also defined as the following

$$X = Q \cdot \text{diag}(u) \cdot Q^\top \quad (\text{A.4})$$

and the regularization of X is done by

$$X \leftarrow X + \epsilon I \quad (\text{A.5})$$

where $\epsilon > 0$ is a small constant and I is identity matrix. The diagonal vector $M \in \mathbb{R}^n$ is obtained by

$$M_i = (Q^\top X^{-1} Q)_{ii}, \quad i = 1, 2, \dots, n. \quad (\text{A.6})$$

The maximum value of M and its index is obtained as follows

$$m_{\max} = \max(M_i), \quad i_{\max} = \arg \max(M_i). \quad (\text{A.7})$$

For updating weight vector u ,

$$u \leftarrow (1 - \alpha)u, \quad u_{i_{\max}} \leftarrow u_{i_{\max}} + \alpha \quad (\text{A.8})$$

where α is the step size and calculated as follows

$$\alpha = \frac{m_{\max} - 4}{4 \cdot (m_{\max} - 1)}. \quad (\text{A.9})$$

And final ellipsoid matrix A is determined as

$$A = (P' \cdot \text{diag}(u) \cdot P'^{\top} - CC^{\top})^{-1} / 3 \quad (\text{A.10})$$

where the center of the ellipsoid C is defined as

$$C = P' \cdot u. \quad (\text{A.11})$$

Add regularization to A

$$A \leftarrow A + \epsilon I. \quad (\text{A.12})$$

A.2. Ellipsoid Fitting: PCA

Given a P' matrix of an object candidate, PCA enables the dimension reduction and computes the principle components of the given P , which correspond to the eigenvectors of the covariance matrix. The eigenvectors can also be considered as principle axes and semi-axes lengths can be computed. Given a set of points $P' = \{p'_1, p'_2, \dots, p'_n\} \subset \mathbb{R}^3$, where each point $p'_i = (x_i, y_i, z_i)^{\top} \in \mathbb{R}^3$, the goal is to fit an ellipsoid .

The matrix $P' \in \mathbb{R}^{3 \times n}$ containing the point coordinates is defined as

$$P' = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ z_1 & z_2 & \dots & z_n \end{bmatrix}. \quad (\text{A.13})$$

The center of the ellipsoid C is $C \in \mathbb{R}^3$ and computed as

$$C = \frac{1}{n} \sum_{i=1}^n p'_i. \quad (\text{A.14})$$

The centered data is computed as

$$P'_c = P' - \mathbf{1}_n C^{\top} \quad (\text{A.15})$$

where $\mathbf{1}_n$ is an $n \times 1$ vector of ones. The covariance matrix of the centered data is computed as

$$\Sigma = \frac{1}{n} P'_c (P'_c)^\top \in \mathbb{R}^{d \times d}. \quad (\text{A.16})$$

Compute corresponding eigenvectors $v_1, v_2, \dots, v_d$ of the covariance matrix Σ

$$\Sigma \mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, 2, \dots, d. \quad (\text{A.17})$$

The eigenvectors v_i represent the principal components (directions of maximum variance).

The next step is to determine the semi-axes lengths of an ellipsoid. For each principal component $\mathbf{v}_i$, project the points onto the axis

$$p_i^{\text{proj}} = v_i^\top P'_c \quad (\text{A.18})$$

where $p_i^{\text{proj}} \in \mathbb{R}^n$ contains the 1D projections of all points onto the principal component v_i . Additionally,

$$\mu_{\max} = \frac{1}{s} \sum_{i=n-s+1}^n p_i^{\text{proj}} \quad (\text{A.19})$$

$$\mu_{\min} = \frac{1}{s} \sum_{i=1}^s p_i^{\text{proj}} \quad (\text{A.20})$$

where $s = 0.05n$. The semi-axes length is computed as

$$L_i = \frac{\mu_{\max} - \mu_{\min}}{2}. \quad (\text{A.21})$$

Note that, to represent an ellipsoid a center vector C and a symmetric, positive-definite matrix A whose eigenvalues can represent the semi-axes length. From A , the semi-axis length can be computed as

$$\frac{1}{\sqrt{\lambda_i}}. \quad (\text{A.22})$$

Therefore, given semi-axes lengths and eigenvectors A matrix can be obtained: Create a diagonal matrix D where the diagonal entries are related to the semi-axis lengths,

$$D = \begin{bmatrix} \frac{1}{L_1^2} & 0 & 0 \\ 0 & \frac{1}{L_2^2} & 0 \\ 0 & 0 & \frac{1}{L_3^2} \end{bmatrix} \quad (\text{A.23})$$

where $V \in \mathbb{R}^{3 \times 3}$ is the matrix whose columns are the eigenvectors of the covariance

matrix, i.e.,

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} \quad (\text{A.24})$$

$$A = VDV^\top. \quad (\text{A.25})$$

A.3. Ellipsoid Visualization

Consider the given ellipsoid form. As A is a symmetric, positive-definite matrix, it can be written as

$$S^\top S = A \quad (\text{A.26})$$

where $S = \sqrt{A}$. The quadratic matrix for an ellipsoid representation can be rewritten as

$$(P - C)^\top S^\top S (P - C) = 1. \quad (\text{A.27})$$

Now note that the set of points on the surface of a unit sphere satisfies $u^\top u = 1$. Hence, $S(P - C)$ is a point on the surface of a unit sphere,

$$S(P - C) = u. \quad (\text{A.28})$$

We can define a map from a point on the surface of a unit sphere to a point on the surface of the ellipsoid defined by the following equation,

$$P = S^{-1}u + C. \quad (\text{A.29})$$

To generate points on the surface of the ellipsoid, points on the surface of the unit sphere parameterized by θ and ϕ as the following

$$u_1 = \sin(\theta)\cos(\phi) \quad (\text{A.30})$$

$$u_2 = \sin(\theta)\sin(\phi) \quad (\text{A.31})$$

$$u_3 = \cos(\theta) \quad (\text{A.32})$$

where $\theta \in [0, \pi]$, and $\phi \in [0, 2\pi]$ are used.

APPENDIX B: USER GUIDE

Semantic place map (SPM) construction and recognition were designed as a Cmake project named *otsr* in C++. The project consists of the following processes.

- Segmentation of the given frames to obtain object candidates
- Refining object candidates
- Extracting states from the object candidates
- Tracking the object candidates
- Constructing semantic place map
- Recognizing the semantic place map

otsr includes *segmentframes.py*, *read_refined.cpp*, *otsr.cpp*, *ellipsoid.cpp*, *Hungarian.cpp*, *object_recognition.py*, *SPM_construction_recognition.py*, *otsr.h*, *ellipsoid.h*, *Hungarian.h*, and *CMakeLists.txt*

In the project, each RGB and depth frame pair is stored in a folder.

segmentframes.py is used for RGB frame segmentation with FastSAM. It takes the RGB frames from the folders, applies the FastSAM segmentation algorithm, and saves the object candidate masks to the related folder.

read_refined.cpp reads the masks from the folders and refines the object candidates.

otsr.cpp extracts the states of each object candidate, and tracks the object candidates. It also conducts the construction and recognition processes of the SPM.

ellipsoid.cpp is included by *otsr.cpp*. It computes the ellipsoid for an object candidate, visualizes the ellipsoid as a point cloud, and finds the minimum distance

between two given object candidate ellipsoids.

Hungarian.cpp is included by *otsr.cpp*. It employs the Hungarian algorithm for data association in tracking object candidates.

object_recognition.py is included by *otsr.cpp*. It applies the YOLO image classification model on the given object candidate images and returns an ID for each object candidate.

SPM_construction_recognition.py is included by *otsr.cpp*. It adds nodes and edges following object recognition and also employs SPM recognition.

otsr.h, *ellipsoid.h*, and *Hungarian.h* include the necessary methods for the corresponding operations in *otsr.cpp*, *ellipsoid.cpp*, and *Hungarian.cpp*, respectively.

To construct and recognize SPM, the steps given below need to be followed:

- Build the project: Create a directory named *build* within the project directory, navigate to the *build* directory via the terminal, and execute *cmake ...*. Then, return to the project directory by typing *cd ..* and compile the project using *make*.
- For segmentation of frames, run *segmentframes.py* file
- For refining of object candidates, run *read_refined.cpp*
- For extracting states, tracking of the object candidates; constructing, and recognizing of SPM, run *otsr.cpp* file.