CONSTRUCTING SEMANTIC PLACE REPRESENTATIONS VIA OBJECT DISCOVERY AND VISUAL EXPLORATION

by

Çağatay Odabaşı

B.S., Electronics and Communication Engineering, İzmir Institute of Technology, 2014

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 Electronics and Communication Engineering

Boğaziçi University

2017

CONSTRUCTING SEMANTIC PLACE REPRESENTATIONS VIA OBJECT DISCOVERY AND VISUAL EXPLORATION

APPROVED BY:

Prof. H. Işıl Bozma ..... (Thesis Supervisor)

Prof. Yağmur Denizhan .....

Prof. Hakan Temeltaş .....

DATE OF APPROVAL: 20.03.2017

ACKNOWLEDGEMENTS

I firstly would like to thank to my supervisor Prof. H. Işıl Bozma for her guiding and support during my thesis. She provided me a top-class research environment in terms of both people and the facilities. As such, we presented our first work in a top-class workshop in just a few months after I started to Intelligent Systems Laboratory, Boğaziçi University.

Second, I also want to thank to my thesis committee, Prof. Yağmur Denizhan and Prof. Hakan Temeltaş for their constructive critics on this work.

I want to thank to Deniz Şenel for her support and ideas during this work, also want to thank to Mahmut Demir for his incredible work that I used in this thesis. Also, I want to thank to Dr. Hakan Karaoğuz, Esen Yel, Berkan Höke, Kadir Cumali, Kadir Türksoy, Halil Samed Çıldır, Aybars Safi for being incredible companions in our laboratory.

This work has been supported in part by TUBITAK EEAEAG-115E380.

Also, I want to thank to all of my friends for their support in my life. Especially, I want to express my special thanks to Aydın Haydar Işık, Gökhan Öztarhan, Selim Fırat, Alper Öztürk, Erçin Çağan Duran and Oğuzhan Oral for their support. I don't think that I would come to this point without them.

Finally, this thesis is dedicated to my family namely my mother, my father, my sister, my wife, my aunts, my uncle and his family and my grandparents. Without their dedication, I would definitely not be here.

ABSTRACT

CONSTRUCTING SEMANTIC PLACE REPRESENTATIONS VIA OBJECT DISCOVERY AND VISUAL EXPLORATION

The goal of this thesis is to develop a Semantic Place Model (SPM) using a mobile robot with a pan-tilt camera. Here, a semantic place model is defined to be an egocentric representation of objects and their spatial relations in the S^2-space. The construction of SPM consists of two parts. Firstly, the object discovery and visual exploration are used in a coupled manner. The visual exploration is driven by the object discovery via constructing a family of artificial potential function using the properties of object candidates. The object candidates are determined via spatio-temporally coherent segments based on a previous work on semantic place recognition in our laboratory. The benefit of this method is that it enables robot to generate its own video input which is crucial for autonomous mobile robots. In the second part, we propose a novel approach to constructing semantic place models. First, object candidates are assigned labels via employing a convolutional neural network based recognition system. The robot then uses the object labels along with their spatial relations to construct the semantic scene model as an attributed graph. It then merges the constructed scene model with its existing semantic place model in order to combine object information from different viewing directions and locations to get a complete representation of a place. In this way, the robot is able to cover a larger area. The resulting models are evaluated in a series of on-robot experiments.

ÖZET

NESNE BULGULAMA VE GÖRSEL KEŞİF İLE ANLAMSAL ORTAM GÖSTERİMİ OLUŞTURULMASI

Bu tezin amacı; pan-tilt kameraya sahip bir robot kullanarak içinde bulunulan ortamın anlamsal modelinin oluşturulmasıdır. Anlamsal modelin oluşturulması 2 farklı bölümden oluşmaktadır. İlk kısımda, nesne keşfi ve görsel keşif eş zamanlı ve birbirlerine bağlı bir şekilde yapılmaktadır. Kafa hareketi, nesne adaylarından oluşturulmuş bir dizi yapay gizilgüç işlevi tarafından yönlendirilmektedir. Nesne adaylarını bulmak için kullandığımız yöntemde laboratuvarımızda önceden yapılmış bilişsel ortam tanımlamaya yönelik bir çalışma temel alınmıştır. Bu çalışma zamansal ve uzaysal olarak uyumlu bölütleri bize vermektedir. Tezde sunduğumuz yöntemin en önemli kazanımı; robotun kendi görsel girdisini oluşturmasına olanak sağlamasıdır ki bu otonom robotlar için vazgeçilmez bir özelliktir. İkinci kısımda ise; bulunan nesne adayları kullanılarak anlamsal modeller oluşturulmaktadır. Anlamsal modeller oluşturulmak için bulunan nesne adayları nesne tanıma yöntemi tarafından anlamsal ve uzaysal olarak etiketlenir. Etiketlenen bu nesnelerin arasındaki bağlar kurularak anlamsal modeller oluşturulur. Daha sonra robot farklı bir pozisyona getirilerek yeni bir anlamsal model oluşuturulur ve bu oluşturulan yeni model, önceki modellerle daha kapsamlı bir model elde etmek için birleştirilir. Tezde burada yazılan savlarımız deneysel olarak ispatlanmaktadırlar.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . ix
LIST OF SYMBOLS . . . x
LIST OF ACRONYMS/ABBREVIATIONS . . . xii
1. INTRODUCTION . . . 1
1.1. Approach Summary . . . 2
1.2. Contributions . . . 3
1.3. Outline . . . 3
2. COUPLING OBJECT DISCOVERY AND VISUAL SCENE EXPLORATION 5
2.1. Overall Approach . . . 8
2.2. Visual Scene Exploration . . . 9
2.3. Experiments . . . 13
3. SEMANTIC PLACE MODELS . . . 18
3.1. General Approach . . . 18
3.2. Semantic Scene Model . . . 20
3.2.1. Temporary SSMs . . . 22
3.2.2. Constructing SSMs . . . 23
3.3. Semantic Place Model . . . 24
3.3.1. Matching SSM with SPM . . . 25
3.3.2. SSM-SPM Registration . . . 26
3.4. Experiments . . . 27
4. UNSUPERVISED OBJECT LEARNING . . . 37
4.1. Methodology . . . 38
4.1.1. Clustering . . . 40
4.1.2. Learning . . . 41

4.2. Experiments 41
5. CONCLUSION 45
REFERENCES 46
APPENDIX A: HARDWARE & SOFTWARE 54
A.1. Hardware 54
A.2. Hardware Setup 54
A.3. Required Software & Libraries 56
A.3.1. Software Design 57
A.3.2. Running the Software 58
APPENDIX B: OBJECT DISCOVERY 61

LIST OF FIGURES

Figure 2.1. Robot's visual exploration. 6

Figure 2.2. Object discovery and visual scene exploration. 9

Figure 2.3. Generating a camera movement from f^k to f^{k+1}. 12

Figure 2.4. Determining object candidates. 13

Figure 2.5. Visual exploration via a succession of 49 camera movements 14

Figure 2.6. Comparative paths in $\mathcal{F}$. 15

Figure 2.7. The explored scene is obtained via stitching of images 16

Figure 3.1. Constructing a semantic place model. 19

Figure 3.2. Semantic scene model. 21

Figure 3.3. SSM construction. 24

Figure 3.4. Merging SSM into SPM 27

Figure 3.5. SPM with varying N_r and τ_d 30

Figure 3.6. Matched objects across scenes 31

Figure 3.7. Updating the semantic place model via merging without SSM tracking. 32

Figure 3.8. SPM created in Place 1 33

Figure 3.9. Laboratory scene with generated SPM. 35

Figure 3.10. Lounge scene 36

Figure 4.1. Sample images of learned objects. 39

Figure 4.2. Stacked feature vectors of 672 object candidates. 42

Figure 4.3. Feature Extraction 43

Figure 4.4. Sample clusters in the appearance space. 43

Figure A.1. Electrical and electronic connections on the robot 55

Figure A.2. The mobile robot with a pan-tilt camera. 55

Figure A.3. Softwares and Hardwares 56

Figure A.4. 3D Drawing of the camera attachment. All dimensions are millimeter. 60

Figure B.1. Determining object candidates. 61

Figure B.2. Node existence matrix. 63

LIST OF TABLES

Table 2.1. Comparative evaluation. 17

Table 3.1. Properties of proposed semantic place models. 19

Table 3.2. Comparative recognition performance. - (Not available) 28

Table 3.3. Object recognition performance 29

Table 3.4. Object recognition performance in SSMs formed via tracking objects' coherency 34

Table 4.1. Comparative recognition precision. 44

Table A.1. Github repos written in the scope of this thesis 58

Table A.2. Parameter List 59

LIST OF SYMBOLS

$a(\mu_l^{-1}(f'))$	Area of object candidate
$c_m^p \in R^2$	Location
d_s	Average distance between the matches of two SPM
$\mathcal{F} \subset S^2$	Field of view of robot
$f = \begin{bmatrix} f_1 & f_2 \end{bmatrix}^T$	Viewing direction
$G(x_m)$	Semantic Scene Model
$G'(x_m, f^k)$	Temporary Semantic Scene Model
$G^p = (\Lambda^p, E^p, \mathcal{A}^p)$	Semantic Place Model
I^l	Incoming image
$k \in \mathcal{K}$	Index set of movements
$l \in \mathcal{F}$	Index set of viewing directions
M_p	Series of base points
N_c	Number of clusters emerged in unsupervised learning
N_m	Number of matches in creating Semantic Place Model
N_r	Spatial relations threshold
$N_\epsilon(f)$	Neighborhood of f
$\mathcal{O}(t)$	Set of viewing direction
p	Place
q	Descriptor vector
S^2	Unit sphere
$S_i^l \in \tilde{\mathcal{S}}^l$	The centroids of object candidates
t	Time instance
$\mathcal{V}(t)$	Set of inhibited object candidates
$w(f')$	Weighting function
$\mathcal{X} \subseteq R^2 \times S^1$	The base space
$x_m^p = \begin{bmatrix} c_m^{p\,T} & \alpha_m^p \end{bmatrix}^T$	Robot's position vector
$\alpha \in S^1$	Heading

ϵ	S^2 proximity threshold
λ	Labeling function
$\mu_l(S_i^l) \in \mathcal{F}$	The viewing directions of centroids of object candidates
ρ	Radius of object candidate
τ_c	Cluster cardinality threshold
τ_d	Object candidate S^2 proximity threshold
τ_h	Object clustering threshold
τ_j	Object candidate duration threshold
τ_m	Object matching cost threshold
$\varphi(f,t)$	Artificial Potential Function

LIST OF ACRONYMS/ABBREVIATIONS

1024-D	1024 dimensional
2D	Two Dimensional
3D	Three Dimensional
APF	Artificial Potential Function
CNN	Convolutional Neural Network
CPU	Central Processing Unit
GPU	Graphical Processing Unit
ISL	Intelligent Systems Laboratory
PD	Proportional-Derivative
ROI	Region of Interest
ROS	Robot Operating System
SIFT	Scale Invariant Feature Transform
SPM	Semantic Place Model
SSM	Semantic Scene Model

1. INTRODUCTION

Autonomous mobile robots must be able to detect the objects around them and explore the scene in parallel by using their cameras. Thus, they will be able to construct a semantic place model of their surroundings to do some certain tasks. Here, a semantic place model is defined to be an egocentric representation of objects and their spatial relations. The main advantage of a semantic place model is that it is a shared model between humans and robots, meaning that both sides can understand the model which is important for robots which share their working places with humans.

The construction of semantic place models is far from solved, since the real world environments are highly cluttered, thus a very high precision object recognition system is required. Also, this should be done autonomously. Furthermore, many tasks require it to be applicable in real-time. Hence, the exploration algorithm should be highly efficient and hence should not collect redundant information.

The construction of semantic place models is only possible by recognizing the objects around the robot based on their appearances. For instance, humans are capable of looking around in a scene and detecting the objects. According to [1], this action consists of three parts: detection, recognition and learning. Current approaches are able to detect and recognize the objects. Yet, learning is done in a supervised manner, since the training dataset is prepared by humans. Teaching all objects in a supervised manner is a challenging task, since the number of object models that need to be learned is very large (≈30,000) with appearances varying widely [2]; that's why, the robot should learn some new objects as it operates.

1.1. Approach Summary

The proposed approach consists of three related parts. We assume that the robot has a pan-tilt head, since a moving head can enable robot to look freely and gather more information beyond just one frame.

First, the robot discovers objects and visually explores its surroundings in a coupled manner. This is based on previous work in which spatio-temporally coherent segments are determined [2]. Then, these object candidates are used to generate a family of artificial potential functions to move the optical axis of robot through the most important parts of the place. Thus, different from related work, the robot can the scene both efficiently and autonomously.

Second, the semantic place model is constructed. This is egocentric representation of objects and their spatial relations in the S2-space. The semantic model is a model that can be shared with humans - as both the robots and humans can understand the constructed model. For this, the object candidates are labeled with their semantic meaning along with their spatial relations. Convolutional Neural Networks [3, 4] are used as our object recognition module. Each object candidate is labeled with its semantic label and spatial relations with other object candidates. This information is then used to construct the semantic scene model as an attributed graph. It then merges the constructed scene model with its existing semantic place model in order to incorporate object information from different locations. Finally, the resulting semantic model incorporates objects as seen from both different viewing directions and locations.

Finally, we propose a method for robots to add new objects to their object recognition dataset in an unsupervised manner, since the initial dataset may not suffice as the time passes. For this, the object recognition includes an "unknown" object class. When an object candidate is labeled as unknown, its feature vector gathered from high level filters of convolutional neural network is sent to the incremental clustering algorithm. When the number of object candidates inside a cluster exceeds a pre-defined

threshold, the robot decides to enhance its dataset with new object class.

1.2. Contributions

The contributions of this thesis can be summarized as follows:

- Coupling object discovery and visual exploration: A novel approach that couples the object discovery and visual exploration. The presented method lets the robot detect the object candidates by using spatio-temporally coherent segments and use those object candidates to drive its head to explore the scene visually. To our knowledge, this is the first work that couples the object discovery and visual exploration.
- Constructing semantic scene model and semantic place model: The construction of semantic place model is novel in this thesis, since most works focus on creating them from single locations; however, in our approach each semantic place model includes the knowledge of objects and their spatial relations in S^2 from a multitude of locations and viewing directions. Our method can generate different semantic scene models from different locations and combine them in a single semantic place model without any mapping requirements.
- Unsupervised object learning: We propose a method for mobile robots to extend their object recognition dataset as they run and get some images over time. By doing this, we use single convolutional neural network to get the label of object and extract the features in parallel, thus it saves computation time.

1.3. Outline

This thesis is organized as follows: In Chapter 2, coupling object discovery and visual scene exploration methodology is presented and its effectiveness is supported by experiments done on a real robot operated in realistic environments. In Chapter 3, the constructions of the semantic scene model and semantic place model are explained and they again supported by real-world experiments. In Chapter 4, a novel unsupervised

learning model for adding new object classes to training set is proposed. Then, in Chapter 5, the thesis is briefly concluded. In Appendix A, the hardwares and softwares which are used in this thesis are explained to the reader. Finally, in Appendix B, the object discovery algorithm is presented to complete the mathematical background.

2. COUPLING OBJECT DISCOVERY AND VISUAL SCENE EXPLORATION

The detection of the major objects in an environment is crucial for many robotic applications [5]. In the last years, with the help of new tools in computations such as GPU's, computer vision researchers have drastically advanced in the object detection area with very high precision and recall rates. However, their applications in robotics have remained to be limited, as the environments are cluttered and highly dynamic. The most guaranteed way to create an object detection system is via running a sliding window method followed by an object recognition module. As the sliding window method generates millions of windows. Unfortunately, this is not practical for robotics. As such, the recent trend is discovering the category independent object candidates without any prior knowledge about the objects' appearances and categories [6]. The advantage of such an approach is that these candidates can be used in anywhere with high computational capacity such as cloud systems instead of processing them on the robot. The computer vision systems approach to this problem as a proto-objects finding problem which are derived from low-level segmentation systems [5, 7, 8]. The object discovery has been considered in computer vision and cognitive psychology communities as object proposal generation and object perception problems respectively [9].

The general approach to this problem is to generate a high number of object proposals as to cover the all objects in the scene [6]. These approaches mostly consider single image rather than a video sequence. The trade-off between computational tractability and high detection quality is addressed via considering 'detection proposals', that serve to prune down the object candidates [10–12]. There are also saliency based approaches that aim to be selective to foreground entities [9]. However, these functions are set to give maximal results on foreground objects. Their advantage is that a low number of object candidates is determined; however, this makes them miss larger objects and backgrounds. This is a problem since, most applications in robotics need to be able to detect all objects in the scene in order to semantically understand

the scene.

Using video sequence is shown to be advantageous over using single image. First, unless the image is taken from too far from the scene, it is impossible to cover the scene sufficiently with single image; however, a video sequence with sufficient coverage can be satisfactory. Second, [13, 14] have shown that the object candidates become more reliable with the use of multiple views and [15,16] have shown with the use of temporal coherency information. The problem is that they assume that the video is generated by an external user which is not a case for a mobile robot. An autonomous mobile robot must generate its own video while visually exploring the scene. So, the robot should move its camera with a proper setup such as a pan-tilt system to get the maximal scene coverage. The closest well known method to this approach is presented in [13] which directs its camera to objects and uses a SLAM based iterated greedy search method to ensure the sufficient coverage; however, the mapping itself is not fully solved problem, so visual exploration must be valid without mapping.

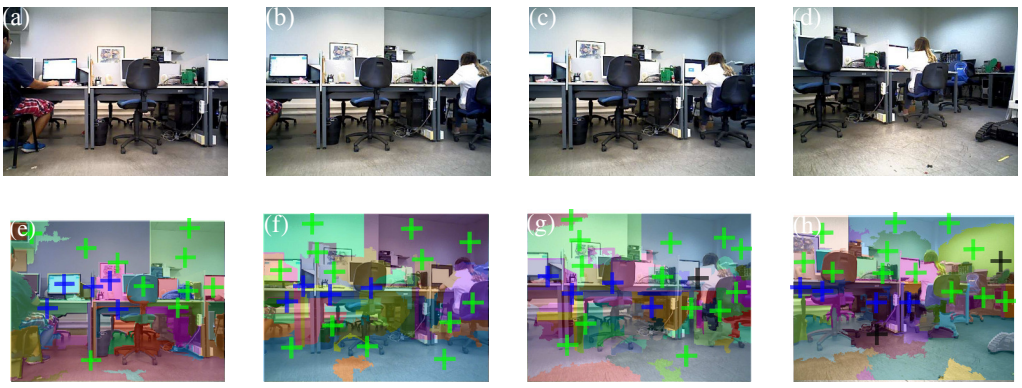


Figure 2.1. Robot's visual exploration through a succession of 4 movements. (a) I^k. (b) I^{k+1}. (c) I^{k+2}. (d) I^{k+4}. (e) $\tilde{\mathcal{S}}^k$. (f) $\tilde{\mathcal{S}}^{k+1}$. (g) $\tilde{\mathcal{S}}^{k+2}$. (h) $\tilde{\mathcal{S}}^{k+4}$.

Our methodology can be summarized as follows:

First, the spatio-temporally coherent segments namely object candidates are found based on [17]. To do this, the low level segments are tracked according to

their color, size and position. The coherent segments which are appeared more than a pre-defined threshold are determined as spatio-temporally coherent segments. The tracking is done by a simple graph matching algorithm as explained in Appendix B. This procedure generates low number of object candidates per frame while ensuring the discovering majority of objects.

Second, the object discovery and scene exploration are coupled together. The robot changes its gaze through a succession of movements by constructing an artificial potential function from object candidates. This is motivated by previous work that suggests a scene as containing several attended proto-objects [7, 8, 18, 19]. In this way, the robot can generate its own video as an input to its visual exploring algorithm without mapping. One more advantage of object discovery is that the patches cropped from the candidates are sent to object recognition module; so that, the image patch includes single dominant object instead of the whole image.

A sample sequence generated by a robot is presented in Figure 2.1. The movement in this sequence covers $(0°, 17°) \times (0°, 13°)$ part of its field-of-view. The top row shows images from a succession of 4 movements during a visual exploration. At the bottom row, the object candidates are marked with blue and green crosses at their centroids. The blue crosses indicate the object candidates which are on the scanpath of robot's line of sight. It is observed in Figure 2.1(e) that the robot has already made 5 movements. In the next movement, the robot's line of sight moves towards the object candidate that corresponds to a chair as seen in Figure 2.1(f). It is observed that a few of object candidates are no longer visible. The robot then moves its camera towards the computer as seen in Figure 2.1(g). Again, some of the previous object candidates vanish. Finally, it moves towards another computer on the right of the scene. Through these movements, the robot is observed to discover most of the major scene objects.

2.1. Overall Approach

In the proposed approach, object discovery and visual scene exploration are coupled and hence they effect each other. We assume that the robot has a pan-tilt camera mounted on a moving base. Suppose that at time t_l, the robot is looking in the viewing direction by $f^l \in \mathcal{F}$ where $l \in \mathcal{L}$ denotes its index, $\mathcal{L}$ is the set of time indices and $\mathcal{F} \subset S^2$ denotes its field-of-view. $f \in \mathcal{F}$ is a vector defined as $f = \begin{bmatrix} f_1 & f_2 \end{bmatrix}^T$ where $f_1 \in S^1$ and $f_2 \in S^1$ represent the pan and tilt angles of the camera respectively.

For object discovery, the robot processes the incoming video sequence $\{I^l\}_{l \in \mathcal{L}}$ in order to find the object candidates. Note that this video is being generated by the robot as it is visually exploring the scene. Object candidates are defined by the spatio-temporally coherent segments. The motivation is that segments which prevail over the incoming appearances are good candidates for summarizing the visual content of the scene [17]. This coherency includes the size, position, color and temporal coherency. Spatio-temporally coherent segments are determined via tracking the segments in the incoming appearances. The tracking is based on a simple graph matching and then identifying segments that are matched for a certain time frame of the video sequence. The details of object discovery are explained in Appendix B.

In parallel, the robot explores the scene visually via a succession of camera movements. Here, $k \in \mathcal{K}$ denotes the index set of movements. Each movement starts at one viewing direction $f^k \in \mathcal{F}$ and ends in another $f^{k+1} \in \mathcal{F}$ as shown Figure 2.2(b). This results in a series of viewing directions $\mathcal{F}^k = \{f^m\}_{m \leq k}$. The camera motion is determined by the closed-loop dynamics of a gradient vector field that is associated with a carefully constructed artificial potential function. The construction depends on the object candidates that are known. In order to ensure that the robot's head will look at the new parts of the scene, object candidates whose centroids are within the small neighborhood of the viewing direction f^m, $m \leq k$ are inhibited and hence not taken into consideration in the construction. This cycle is repeated in a continual manner.

The details of visual scene exploration are presented in Section 2.2.

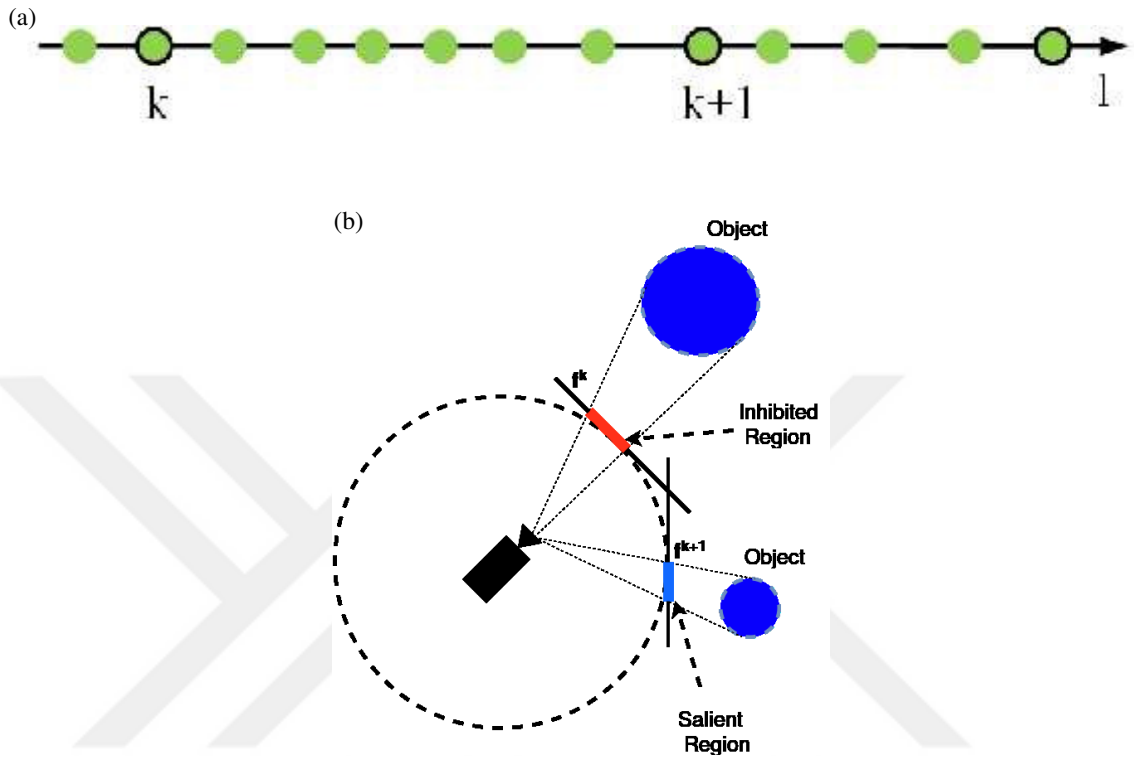


Figure 2.2. (a) Object discovery occurs continually as indicated by the green circles. In parallel, the robot explores the scene via a succession of movements - black-green circles. (b) Each movement from f^k to f^{k+1} depends on the object candidates.

2.2. Visual Scene Exploration

The visual exploration is the result of a series of a camera movements from f^k to f^{k+1} with $k \in \mathcal{K}$. Each movement is driven by an artificial potential function APF φ that encodes the object candidates and needs to be maximized. The movement dynamics is defined by the gradient of the APF as:

$$\dot{f}(t) = \nabla_f \varphi(f, t) \tag{2.1}$$

with initial condition $f(0)$. The velocity input is used to generate an incremental goal viewing direction. A PD controller is then employed to move the camera to the target coordinates. The movement continues until $\nabla_f \varphi(f^{k+1}, t_{k+1}) = 0$ at time $t_{k+1} > t_k$ with viewing direction f^{k+1}. Hence, the index is being updated when the resulting viewing direction reaches a critical point of APF.

There are several advantages of such system. First, [13, 14] states that taking multiple images from an object improves the recognition performance. Our method also allows robot to get multiple views from an object candidate. Also, the resulting motion is not random or exhaustive. It is driven by object candidates; therefore, it ensures that robot looks at some important parts of the scene which are mostly objects. Furthermore, since the robot sends only queries to object recognition module when a motion stops rather than sending the each frame, the computation cost is dramatically decreased. Note that getting a result from an object recognition module is highly expensive in terms of computation time. This also simplifies the resulting model.

The APF $\varphi : \mathcal{F} \times R \rightarrow [0, 1]$ defines the movement, it is carefully constructed to get more information at each movement. It encodes the object candidates. This object candidates driven motion lets robot shift its gaze across the objects. The centroids of the object candidates $S_i^l \in \tilde{\mathcal{S}}^l$ are mapped to a viewing direction $\mu_l(S_i^l) \in \mathcal{F}$. This results in a set of $\mathcal{O}(t)$ viewing directions depending on $t_l \leq t < t_{l+1}$ as:

$$\mathcal{O}(t) = \left\{ \mu_l(S_i^l) \mid \forall S_i^l \in \tilde{\mathcal{S}}^l \right\}$$

where t_l denotes the time passed in terms of number of frames. The object candidates which are fixated upon must not be taken into account. This is defined by the S^2 proximity threshold $\epsilon > 0$ that is used to define a neighborhood $N_\epsilon(f^m) = \{f \in F : \|f - f^m\| \leq \epsilon\}$ of each viewing direction $f^m \in \mathcal{F}^k$. If a centroid of an object candidate is within this neighborhood, this means that the object candidate is not used in construction of APF. The S^2 proximity threshold ϵ needs to be carefully selected -

as it determines the maximum distance over which object candidates are considered to be looked at. If it is too low, the robot can fixate the objects candidates which are too close to each other. In contrast, if it is too large, a larger area is inhibited and some important object candidates might be missed.

The inhibited object candidates are defined by $\mathcal{V}(t) \subseteq \mathcal{F}$ depending on $t_k \leq t < t_{k+1}$:

$$\mathcal{V}(t) = \left\{ f' \in \mathcal{F}^k \mid \exists f^m \in \mathcal{F}^k \text{ s.t. } f' \in N_\epsilon(f^m) \right\} \tag{2.2}$$

Initially, $\mathcal{V}(0) = \emptyset$. As the exploration continues, this set starts to grow by addition of new inhibited areas. The remaining object candidates are then encoded by a Gaussian mixture model parametrized by viewing directions:

$$\varphi(f,t) = \sum_{f' \in \mathcal{O}(t_l) - \mathcal{V}(t_k)} w(f') e^{-\frac{1}{2\rho^2(\mu_l^{-1}(f'))}(f-f')^T(f-f')} \tag{2.3}$$

Here, note that $\rho^2(\mu_l^{-1}(f'))$ refers to the radius of the object candidate with centroid at f' which corresponds to the variance of Gaussian. The term $w(f')$ is the weight of each object candidate. The weights depend on the closeness of the respective object candidates to the robot's horizon [20, 21]. In other words, the distance between horizon $f_2^H = 0$ and the viewing direction namely f' is encoded in the weight $w(f')$. Also, the area $a(\mu_l^{-1}(f'))$ of the object candidate must also be encoded in the weight, since the object candidates with small area tend to be more foreground than bigger ones. The weight is calculated as below:

$$w(f') \propto \frac{1}{e^{|f_2'|} a(\mu_l^{-1}(f'))}$$

A sample APF surface is shown in Figure 2.3(a). As encoded by the parameters $a(\mu_l^{-1}(f'))$ and $\rho^2(\mu_l^{-1}(f'))$ for all $f' \in \mathcal{O}(t) - \mathcal{V}(t)$, the critical points of φ depend on

the size and the shape of the object candidates In the case of well-seperated Gaussians, the robot will fixate exactly on the centroid of object candidates; however, sometimes there may be some overlapping Gaussians, if the object candidates are too close to each other, in that case, the movements induced by them may not necessarily end on the center of object candidates.

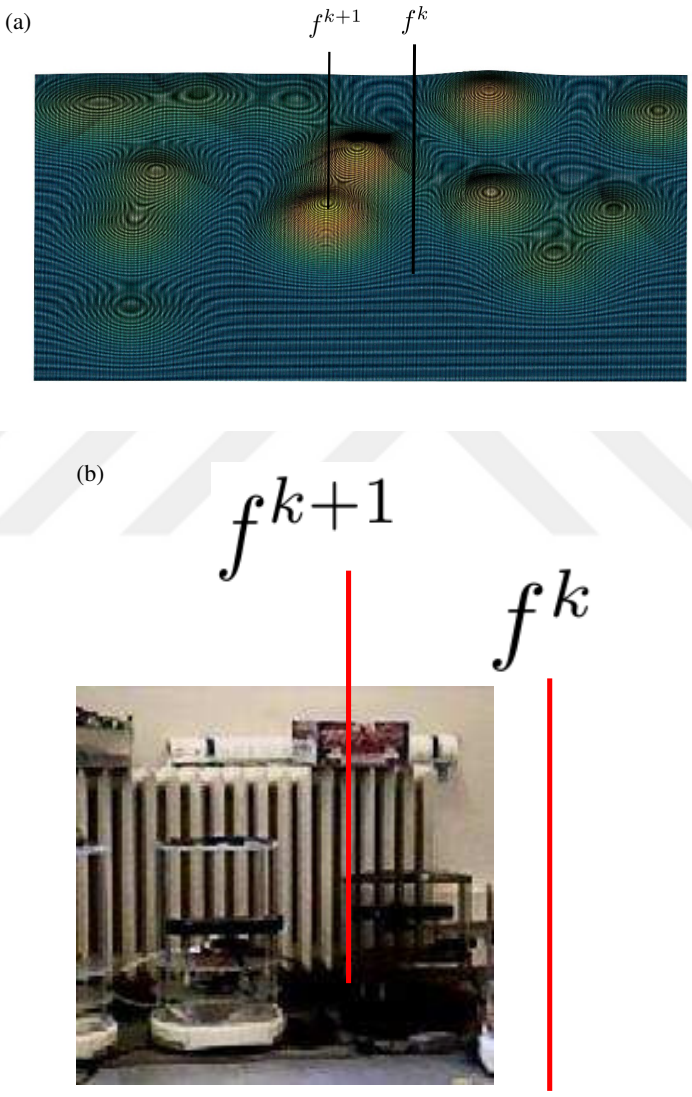


Figure 2.3. Generating a camera movement from f^k to f^{k+1}. (a) The APF $\varphi(f, t_k)$ encodes the object candidates $\mathcal{O}(t_k) - \mathcal{V}(t_k)$.
(b) The resulting movement ends at f^{k+1}.

2.3. Experiments

Experiments are done on a mobile robot having a pan-tilt camera with $\mathcal{F} = (-90°, 90°) \times (-90°, 90°)$ which can be seen in Figure A.2. Its main processor is Intel i5 CPU running at 1.7 GHz. The visual scene exploration starts with arrival of first object candidates. The parameter $\epsilon = 25°$.

First, the object discovery performance is evaluated. An example image sequence taken from one of our experiments which the robot is operated inside a laboratory is demonstrated in Figure 2.4(top row). The scene is highly cluttered. As seen in the figure, the robot's exploring trajectory is downwards. While visual inspection reveals that the first frame has 25 segments in total, only 17 of them are found to be spatio-temporally coherent as indicated by green crossed in in Figure 2.4(bottom row). For example, object candidate associated with the chair is initially not marked as coherent; however, as the robot's head moves; it becomes coherent in the third frame. In this sequence, there are 13 object candidates corresponding to wall, ceiling, 3 light fixtures, 2 picture frame, 2 desks, 3 monitors and 2 chair. 11 of them have been found by the

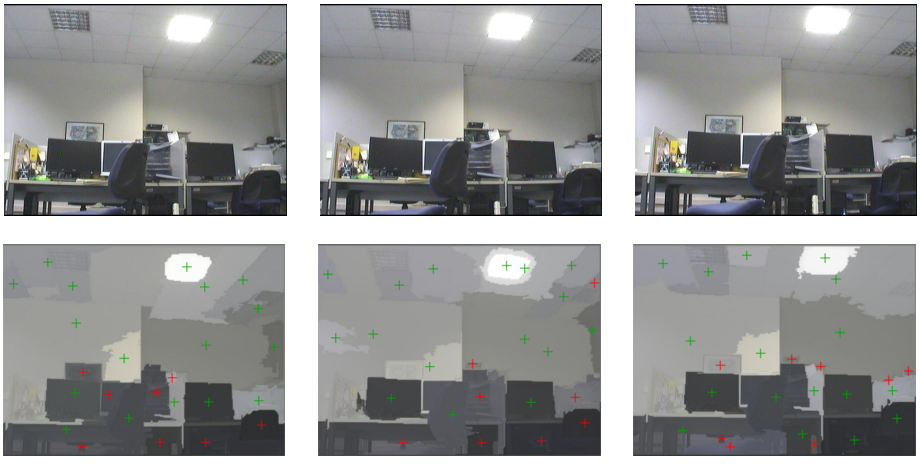


Figure 2.4. Determining object candidates. Top: a sequence of images from a sample video that the robot generates as it is doing visual exploration is shown. Bottom: Object candidates (green crosses) and segments that are not coherent (red crosses).

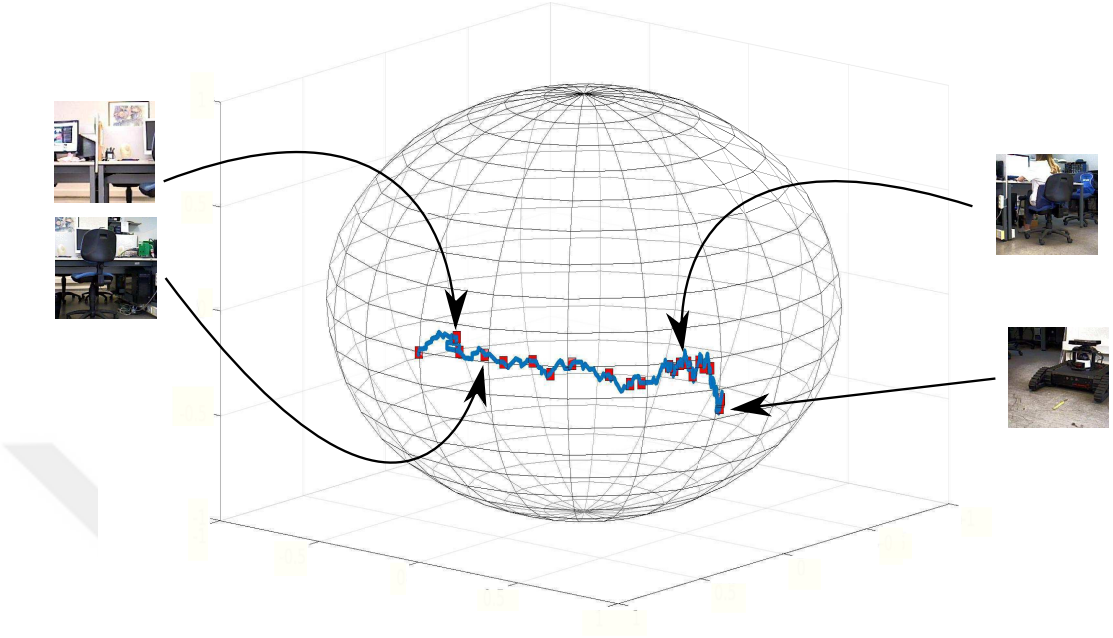


Figure 2.5. Visual exploration via a succession of 49 camera movements with sample views.

robot. The missing candidates are those associated with picture frame and monitor. The reason is that those segments appear in only of few frames. Additionally, in some cases, the robot generates more than one candidate for a single object. Due to illumination effects, a big object can be viewed as consisting of segments.

Second, the efficiency of scanpath generated in the laboratory is evaluated. The resulting scanpath generated through a succession of 49 movements is shown in Figure 2.5. The complete scanpath covers $(0,^{\circ} 75^{\circ}) \times (0^{\circ}, 25^{\circ})$. The best way to evaluate the scanpath is thinking of it as a Traveling Salesman problem among 49 movements. Each of the segment corresponding a movement ideally need to be visited only once; therefore, the suboptimal path can be found by using an inexact genetic algorithm, since the problem itself is a NP-complete. The comparison can be seen in Figure 2.6. In the figure, the red line shows the suboptimal solution, and the blue one indicates our APF based solution. It is observed that our proposed approach generates very similar

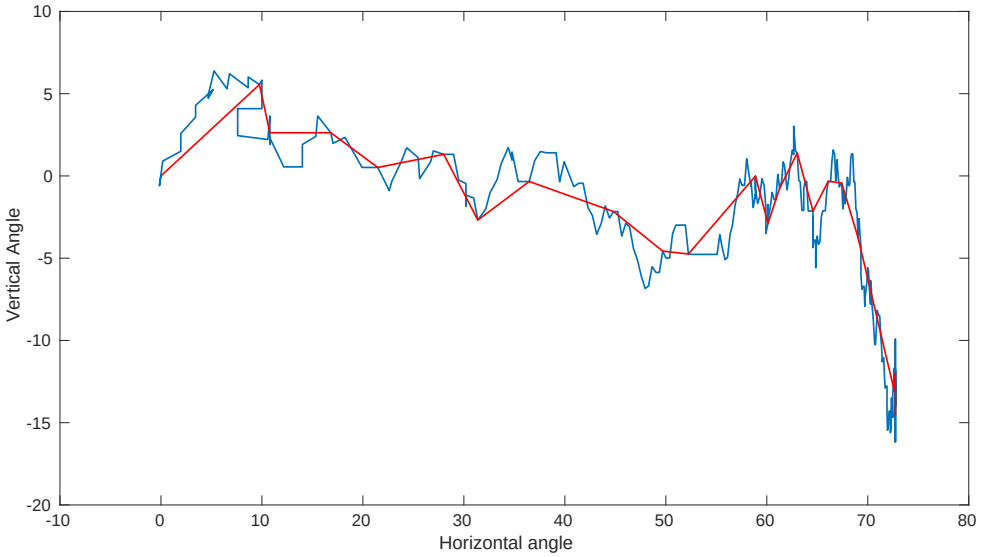


Figure 2.6. Comparative paths in $\mathcal{F}$. Proposed approach (blue) in $\mathcal{F}$ vs genetic algorithm (red)

path to suboptimal path. This enables our robot to explore the scene without solving a NP-complete problem which saves computation and power.

The majority of the objects in a scene should be discovered by the robot in order to cover the scene sufficiently. This is also evaluated in our experiments. 49 movements are generated from processing 500 frames in total and 21 object candidate per frame on average. The resulting object candidates (green crosses) are shown on stitched images in Figure 2.7. It is observed that the robot is able to discover objects that are located in the interior parts of the scene while it has a tendency to miss those that are located towards the scene borders. This is because APF is created to force the robot look towards its horizon.

Finally, we compare our object discovery performance in Table 2.1. The recall and precision are calculated as in [15]. The difference of our work is that our robot generates its own video while exploring the scene instead of getting it from an external user as explained previously. Precision measures the quality of object candidates by evaluating

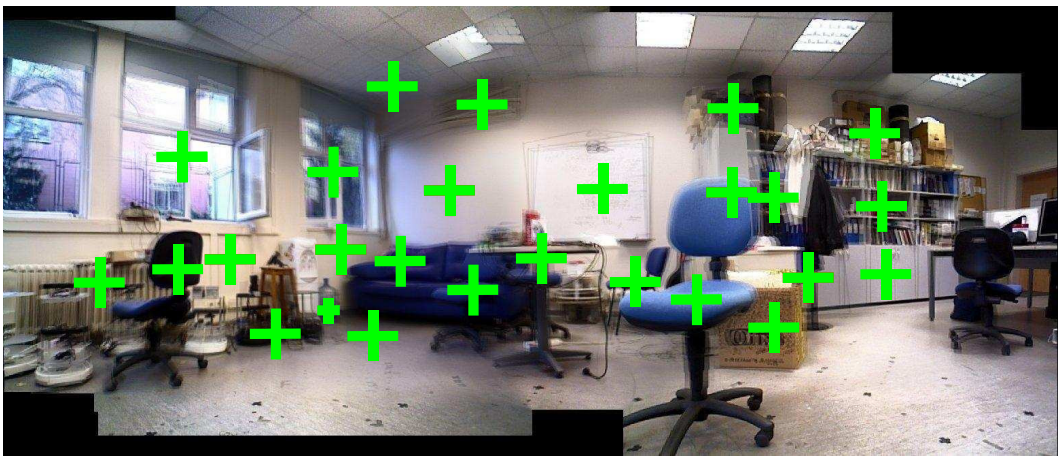


Figure 2.7. By coupling object discovery and scene exploration, the robot is able to find object candidates (green crosses at centroids) as it is visually exploring its scene. The explored scene is obtained via stitching of images from 49 movements.

the ratio of object candidates that actually correspond to object candidates. In our approach, the robot finds on average 21 object candidates per frame. The resulting recall is 53% with 36% precision. This is comparatively better than that presented in [15]. Even at 50 candidates per frame, our method outperforms. In that work, it is observed that as the number of object candidates per frame increases, recall becomes slightly better while precision continues to be relatively lower. All in all, our proposed method enables the robot cover the scene by discovering the objects around it in an autonomous manner. Additionally, one single loop of our algorithm takes less than one second on a mobile robot whose computation power is very limited and without any code optimization.

Table 2.1. Comparative evaluation.

	Methods		
	[15]		Proposed
Video	External		Self-generated
Visual exploration	No		Yes
# Object candidates/frame	50	200	21
Recall	0.2	0.55	0.53
Precision	0.1	0.1	0.36

3. SEMANTIC PLACE MODELS

The next step in semantic place understanding is to construct a semantic model of a place. This is an internal representation that incorporates the major constituent objects and their spatial relations within each place [5]. Such a model should enable robot-human interaction. In this chapter, we propose a novel approach to the construction of semantic place models. The main assumption is the robot has a pan-tilt head including a monocular RGB camera.

There has been extensive work that have focused on the generation of semantic place models as presented in Table 3.1. Most of the approaches use single image for object detection. In order to have sufficient coverage of the scene, there should be multitude of these images. The alternative approach is to use a video input. There are two main advantages of using a video input: First, it can cover the whole scene. Second, the reliability of object candidates is shown to improve when using multiple views [13, 14] or the temporal coherence of the visual stream [15, 22]. Another issue is how to generate these inputs. In most works, visual exploration is done via teleoperation meaning that an external user supplies the input. However, this is not suitable for autonomy. The robot needs to visually explore the scene at the same time and move its camera through different viewing directions as to ensure maximal scene coverage. One approach has been to have the robot point its camera in directions that cover as much new territory as computed using a SLAM-based iterated greedy search method [13]. However, even if mapping may not be available, the robot should explore the scene and create the semantic model. Our approach addresses these issues via having object discovery and object coupling as the first step.

3.1. General Approach

Consider a robot in a particular place p. The robot navigates through a series of M_p base points (location-heading combination) $x_m^p = \begin{bmatrix} c_m^{p\,T} & \alpha_m^p \end{bmatrix}^T$ in this place.

Table 3.1. Properties of semantic place models. Object candidates: Segs (Segmentation based); Image: S (Single) - V (Video); Visual Exploration: T (Tele-operation) - A (Autonomous); Location/View Coverage: S (Single) - M (Multiple)

	Sensory data	Object candidates	Recognition	Image	Graph edges	Visual Exploration	Location coverage	View coverage
[23]	Stereo + Camera	SIFT	Maximum Likelihood	S	R^3	T	S	M
[24]	Stereo Camera	SIFT	SIFT Template Matching	S	R^3	T	M	S
[25]	RGB + Laser	SIFT	KD-Tree or Attention Based Search	S		T	M	S
[26]	RGB-D	Segs	Maximum Margin	S	R^3	T	M	M
[27]	Camera + 3D Laser	Segs	KD-Tree	V	R^2	T	M	S
[28]	RGB-D	Segs	SVM	S	R^2	T	S	S
[29]	RGB	Segs + SIFT	Maximum A Posteriori	S	R^2	T	S	S
Our	RGB	Segs	CNN	V	S^2	A	M	M

The base space $\mathcal{X} \subseteq R^2 \times S^1$ is the set of all possible locations and headings of robot. The number of locations M_p should be large enough as to enable the robot to have a sufficient coverage of the place. The place p is assumed to be learned based on a collection of appearances from M_p locations c_m^p.

The construction of the semantic place model (SPM) is coupled with object discovery and scene viewing. This is an egocentric representation where edges are defined

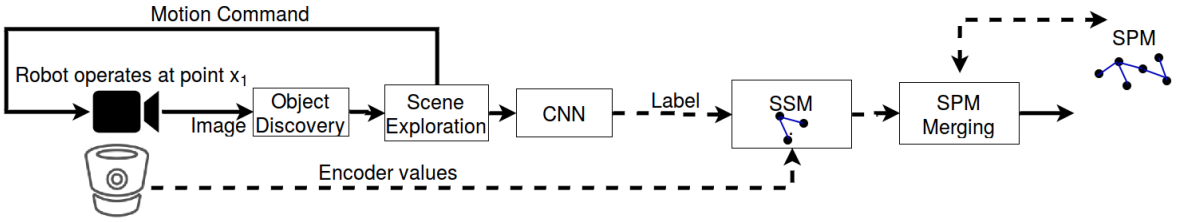


Figure 3.1. Constructing a semantic place model.

by distance information in the robot's viewing space $\mathcal{F} \subseteq S^2$. The nodes correspond to labeled object candidates while edges correspond to relative movement data among pairwise object candidates. The robot assigns labels to the object candidates via an object recognition module.

Integral to the approach is the object recognition module. As object discovery and object recognition are decoupled, it is possible to use a state-of-the-art method for object recognition. Here, we use GoogleNet [3]. GoogleNet is pre-trained on ImageNet [30] that has 1000 classes. Since there are roughly 10000-30000 object classes [2], the system needs to be fine tuned to recognize a given set $\mathcal{L}$ object classes [31] that are not included in ImageNet. This is done by changing the last layer with a randomly initialized layer and re-training the new layer with new dataset. In addition, we have added an 'unknown'class that includes irrelevant images random object images. Hence, the reliability of the model depends on the robustness of object recognition module.

The flowchart of algorithm can be seen in Figure 3.1. At each base point x_m^p, the robot visually explores its surroundings and determines object candidates as explained in Chapter 2.

- At each base point x_m^p, the robot constructs a semantic scene model (SSM). A SSM is an egocentric representation where edges are defined by distance information in the robot's viewing space $\mathcal{F} \subseteq S^2$. The nodes correspond to labeled object candidates while edges correspond to relative movement data among pairwise object candidates.
- As the robot navigates through different base points, it merges the resulting SSM's incrementally in order to create a Semantic Place Model (SPM).

3.2. Semantic Scene Model

A SSM is constructed from the objects and their spatial relations as seen by robot from base location x_m in place p. The model is represented by a an attributed graph

$G(x_m) = (\Lambda^m, \mathcal{E}^m, \mathcal{A}^m)$. The nodes $\Lambda^m = \left\{\lambda(S_i^m) \mid \forall S_i^m \in \tilde{\mathcal{S}}^m\right\}$ correspond to the set of labeled object candidates. A SSM is constructed by first constructing a temporary SSM at each viewing direction f^k at x_m and then tracking coherent object candidates across consecutive viewing directions.

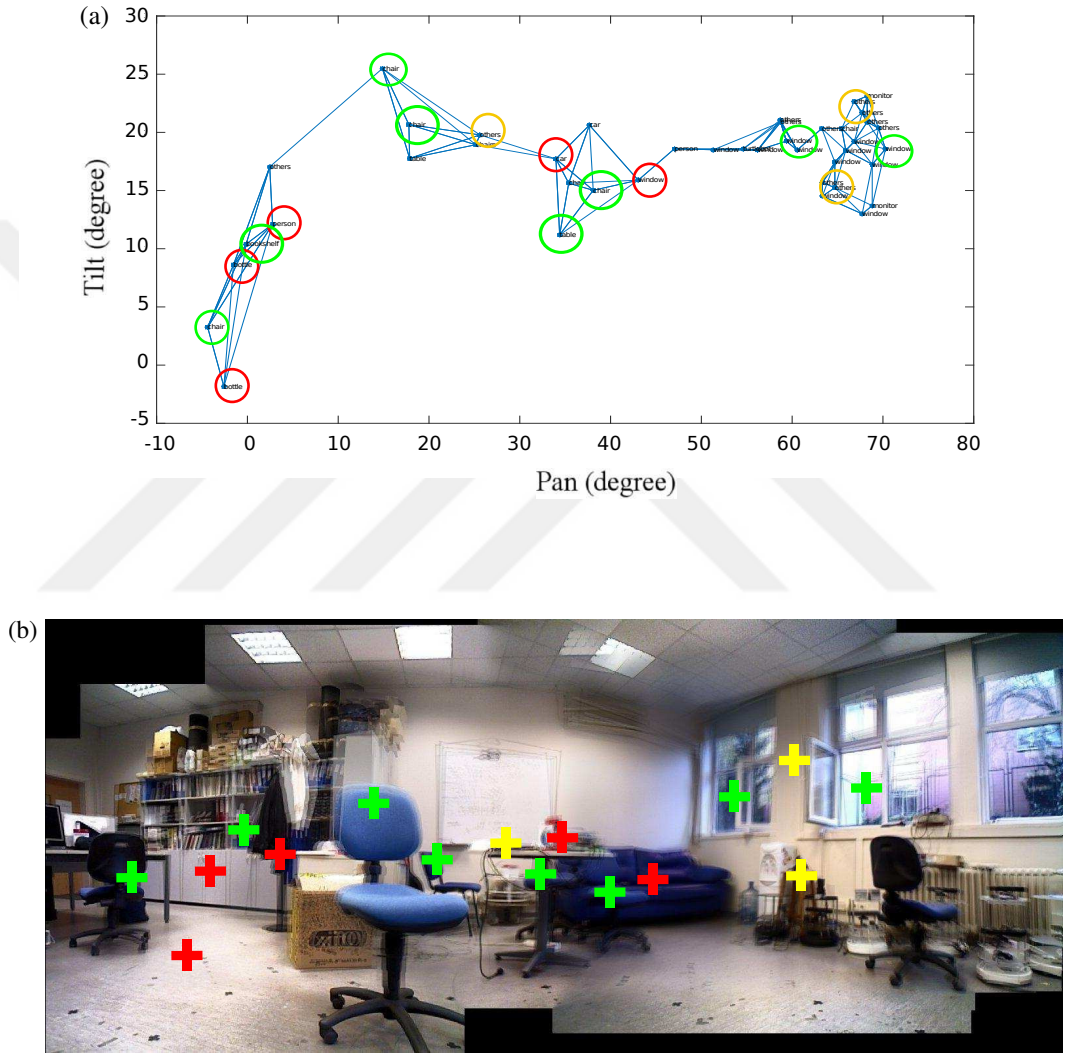


Figure 3.2. Semantic scene model. (a) The model consists of 41 nodes (objects) and edges (relative distance in $\mathcal{F} \subseteq S^2$). (b) Green marks are correctly recognized while reds are incorrect, yellow marks are objects that are (correctly) found to be unknown.

A sample SSM is shown in Figure 3.2(a). It is observed that this model contains 41 objects. It includes objects as seen from a single location, but from 49 different

viewing directions as shown in Figure 3.2(b) using a stitching algorithm [32]. Sample 17 objects are shown on the stitched image where those with green marks have been correctly recognized while those with red are incorrect. Yellow marks indicate objects that are (correctly) found to be unknown. The robot can easily interact with its surrounding including humans using this model.

3.2.1. Temporary SSMs

The robot constructs a temporary SSM at each viewing direction f^k, $k \in \mathcal{K}$ at x_m. The temporary SSM is an attributed graph $G'(x_m, f^k) = (\Lambda_{m_k}, \mathcal{E}_{m_k}, \mathcal{A}_{m_k})$ where Λ_{m_k} are the nodes, $\mathcal{E}_{m_k}$ are the edges and $\mathcal{A}_{m_k}$ are the associated attributes. Its construction is done as follows:

(i) The robot considers the set of coherent segments $\tilde{\mathcal{S}}^k$ associated with each viewing direction f^k. It assigns each segment $S \in \tilde{\mathcal{S}}^k$ a label $\lambda(S)$ via the object recognition module. The set $\Lambda_{m_k} = \left\{ \lambda(S) \mid S \in \tilde{\mathcal{S}}^k \right\}$ is defined as being the resulting set of labeled segments.

(ii) The edges $\mathcal{E}_{m_k}$ are formed considering the proximity of the segments in the S^2 and the maximum number of allowable edges. These are controlled by the object candidate proximity threshold τ_d and spatial relations threshold N_r respectively. The former parameter specifies the greates lower bound that specifies how close two object candidates need to be in order to have their spatial relations considered. The latter specifies the maximum number of spatial relations that can be specified. These parameters must be selected wisely to get a balanced SSM graph structure. A set of experiments are done with the possible values of τ_d and N_r to find the optimal values of them as explained in Section 3.4.

(iii) The attributes are the associated viewing directions f^k.

3.2.2. Constructing SSMs

Once, visual exploration at a base point is finished, the associated SSM is constructed. The SSM is constructed as an attributed graph that consists of all coherently labeled object candidates and their relations. It is defined by $G(x_m) = (\Lambda_m, \mathcal{E}_m, \mathcal{A}_m)$. The nodes Λ_m encodes object candidates that are coherent over consecutive fixation points while the edges $\mathcal{E}_m$ encodes their spatial relations. Its construction is based on the temporary SSMs that have been constructed as the robot visually explores the scene.

(i) Coherent object candidates are determined via tracking object candidates in consecutive temporary SSMs $G'(x_m, f^k)$ and $G'(x_m, f^{k+1})$. The tracking is based on considering the proximity of respective nodes in S^2. Namely, the robot determines all node pairs $S \in \Lambda_{m_k}$ and $S' \in \Lambda_{m_{k+1}}$ whose viewing directions are within N_ϵ neighborhood - $\left\| f^k - f^{k+1} \right\| < N_\epsilon$. The parameter N_ϵ is a priori specified. This parameter is the same N_ϵ in Section 2.2. Generally, the object candidates in this neighborhood are the candidates of same object. This is manually tuned parameter according to the experiences on the robot.

(ii) Next, for each corresponding node pair (S, S'), the robot compares their respective labels $\lambda(S)$ and $\lambda(S')$.

(iii) All object candidates with identical labels are then marked and tracked over the next τ_j consecutive temporary SSMs. If they continue to be coherent, then the respective objects are determined to be coherent. The object candidate duration threshold τ_j is set considering the average number of temporary SSMs in which object candidates appear. This number is experimentally determined to be $\tau_j = 4$. In this case, the object candidates are observed to be generally correct. These objects are added to the set Λ_m while their spatial relations are added to the set $\mathcal{E}_m$.

A sample SSM construction is shown in Fig. 3.3. The SSM contains objects and their spatial relations based on visual data from four consecutive base points. It is

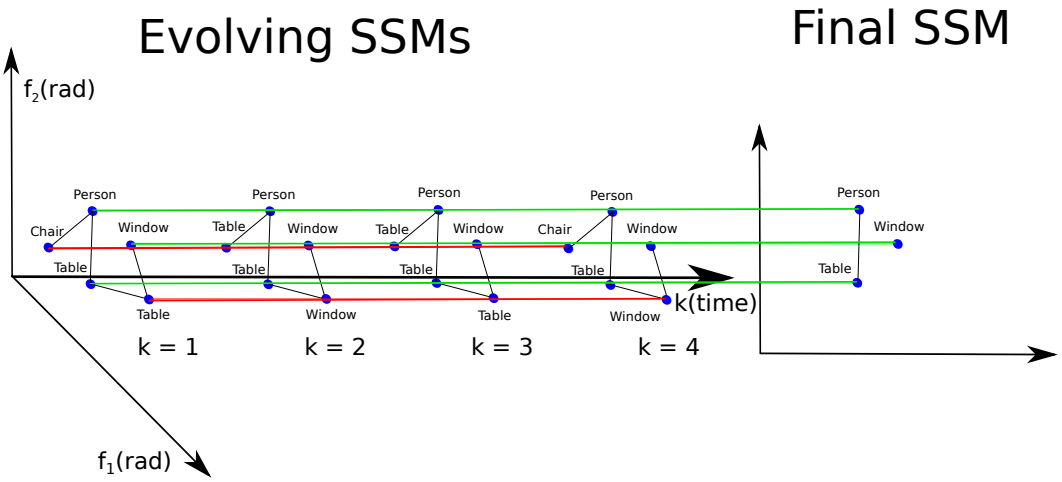


Figure 3.3. A sample SSM construction based on visual data from four viewing directions. The blue lines indicates the coherently labeled nodes while incoherent ones are shown by red. The final SSM only includes the coherent ones.

observed that only coherent object candidates are finally incorporated into the SSM.

3.3. Semantic Place Model

A semantic place model (SPM) incorporates the semantic information collected from multiple locations. Again, it is an attributed graph $G^p = (\Lambda^p, E^p, \mathcal{A}^p)$. Here, Λ^p denotes the set of labeled objects in the place, E^p refers to their spatial relations expressed in $\mathcal{F}$ and $\mathcal{A}^p$ refers to the attributes of labeled objects. In our case, they correspond to the viewing directions. A SPM is constructed by merging SSMs that constructed at individual base points as the robot navigates through in the place. The keypoint here is that the scenes from consecutive base points are likely to have overlapping scene parts. As such, objects common to different scenes are expected to appear in both of the semantic scene models. The robot determines these overlaps and uses them to create a SPM of the place. It achieves this via merging the current SSM with its so-far constructed SPM. The construction of the SPM consists of two steps: First, matching nodes between the current SSM and SPM are determined. Next, the SSM must be registered with the SPM.

3.3.1. Matching SSM with SPM

The matching problem is formulated as a linear assignment problem - namely finding matching nodes in the associated graphs. For this, we use the Hungarian Method [33] to find the optimal matches. In this approach, a cost matrix $C = [c_{ij}]_{\lambda_i \in \Lambda^p, \lambda_j \in \Lambda^m}$ is constructed with each component c_{ij} measuring the cost of match of each node (namely object $\lambda_i \in \Lambda^m$) of the SSM with each node $\lambda_j \in \Lambda^p$ of the current SPM using a cost function.

$$\begin{aligned} c_{ij} = \; & \alpha_1(1 - v(\lambda_i, \lambda_j)) + \alpha_2 \left| e_1^T \mu(\lambda_i)_2 - e_1^T \mu(\lambda_j)_2 \right| + \\ & \alpha_3 \sum_{\lambda' \in \mathcal{N}(\lambda_i)} \sum_{\lambda'\prime \in \mathcal{N}(\lambda_j)} [(1 - v(\lambda(\lambda), \lambda(\lambda')))] \end{aligned} \tag{3.1}$$

The cost function consisted of three terms as weighted by parameters $\alpha_i > 0$,

- The first term checkes whether labels of the respective nodes are identical or not. Here, $v(\lambda_i, \lambda_j) \in \{0, 1\}$ is a binary variable that checks whether their labels are the same or not.

$$v(\lambda_i, \lambda_j) = \begin{cases} 1, & \text{if } \lambda_i = \lambda_j \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

- The second term computes their vertical proximity as measured by the difference between their tilt angles as measured by $e_1^T \mu(\lambda_i) \in S^1$ and $e_1^T \mu(\lambda_j) \in S^1$ where $e_1 = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ is the 2-D unit vector.
- The last term checks the amount of overlap in their neighboring objects $\mathcal{N}(\lambda_i)$ and $\mathcal{N}(\lambda_j)$. The neighboring objects are those with which there exists an edge $(\lambda, \lambda') \in \mathcal{E}_m$:

$$\mathcal{N}(\lambda) = \{\lambda' \in \Lambda_m \mid \exists(\lambda\lambda') \in \mathcal{E}_m\} \tag{3.3}$$

The weights $\alpha_i > 0$, $i = 1, 2, 3$ are selected as to determine the relative weights of the respective terms. These weights are set as $\alpha_1 = 1$, $\alpha_2 = 0.2$ and $\alpha_3 = 0.2$. These parameters are set as follows. Recall that there are maximum of 4 neighbors of each node in SSM. If the nodes compared do not have the same label, the all neighbors of them need to have the same label and also the distance between the nodes compared must be so small. Otherwise, since α_1 parameter is 5 times bigger than α_2 and α_3 and only 4 neighbors are allowed in SSM, the cost between nodes becomes much larger than 1. In contrast, if the nodes compared have the same label, the neighbors should have the different labels and also the distance between nodes should be larger to make the cost between nodes larger than 1.

After the initial match, the match values are compared with an object matching cost threshold τ_m. Only matches that are lower than τ_m are considered to be correct. Its value is determined experimentally. In the experiments, $\tau_m = 2$ in order to allow some deviations in both neighborhood and location if respective nodes have the same labels. Otherwise, minimal location and neighborhood differences should be tolerated. Finally, a set of matching nodes is determined as:

$$\mathcal{Q}_m^p = \{(\lambda, \lambda^*) \mid \lambda \in \Lambda_m, \lambda^* \in \Lambda^p\} \tag{3.4}$$

3.3.2. SSM-SPM Registration

The registration of the SSM with the SPM is done via aligning the matched nodes and then combining the two respective graphs accordingly. As such, the relative positions of objects in the S^2-space are correctly set. This makes SPM cover a large part of the place. To do this, the average node to node distance with the matched nodes in the SSM and SPM is calculated. If N_m is the number of matched nodes, this is computed as:

$$d_s = \frac{1}{N_m} \sum_{(\lambda,\lambda^*)\in\mathcal{Q}_m^p} |\mu(\lambda_i) - \mu(\lambda_j)| \tag{3.5}$$

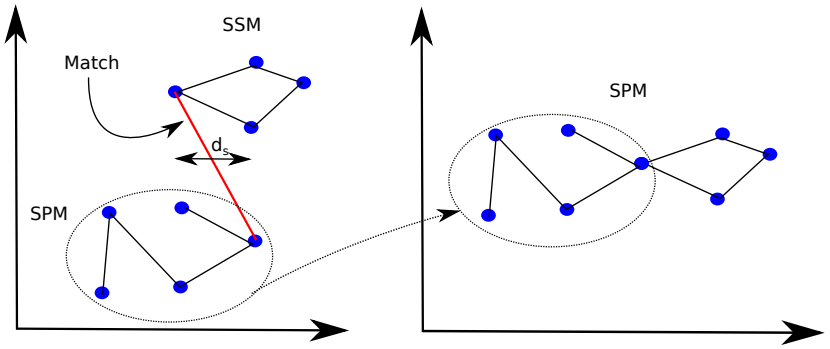


Figure 3.4. Merging SSM into SPM

The value d_s gives us the relative alignment of models. This value is used to update the attributes of the objects in the SSM before they are added to the SPM. A sample case is as shown in Figure 3.4. In the figure, the red line shows the match between the current SPM and an SSM. The resulting SPM covers larger portion of the scene than previous methods.

3.4. Experiments

The experiments are done on a mobile robot with a pan tilt head as discussed. As object recognition module, GoogleNet is fine-tuned to recognize the objects that the robot may face in our department. The objects are taken from ImageNet. These are sofa, window, building, storage drawers, cup, chair, person, tree, monitor, the rest are our robots turtlebot, jaguar and minik which are collected by our robot. Recall that there is also an 'unknown'class. As the deep learning framework, we choose Caffe [34] along with Nvidia Digits, since Caffe has a huge community. As discussed, as fine-tuning requires less number of images to train, the training time is considerably decreased compared to training from scratch. Even for this case, it takes nearly 3 days to train on a GTX970 GPU. There are 6 parameters that need to be set. The parameters τ_d and N_r are selected explained in the sequel. The object matching cost parameter is set as $\tau_m = 2$. The object candidate duration parameter is $\tau_j = 4$ - based on observations that object candidates that prevail accordingly are found to be reliable. The weighting parameters are set as explained.

Table 3.2. Comparative recognition performance. - (Not available)

	Precision		
	Our	[35]	[36]
Bookshelf	50	-	-
Chair	52	51.9	75.9
Jaguar(robot)	82	-	-
Monitor	71	51.7	85.1
Person	19	84.4	96.1
Table	38	-	-
Turtlebot(robot)	62	-	-
Window	46	-	-
Average	56	57.2	86.5

First, the recognition performance with the fine-tuned system is investigated. To do this, total of 957 object candidates from 8 categories that are labeled manually. The results are presented in Table 3.2. The person class has the lowest recognition rate, since generally humans pass through the scene and do not stay there for a time interval requiring to be detected as object candidates. Table 3.2 also shows results in comparison with two other methods. The first method is that of the Pascal 2012 Object Classification Leader [36]. It is observed that our average precision is 35% lower. This is expected since with an on-robot application there are time computation limitations and hence in contrast to [36], an ensemble of different recognition modules cannot be employed. The second comparison is with a object centric pooling method [35]. In this case, precision is only 2% higher than our average precision. The table shows that the recognition module has a reasonable recognition rate for each class in general. As discussed time is also an important consideration in robotic application. So, we should mention it. The query time for each image is nearly 3 seconds.

In addition to per object recognition performance, performance in scenes explored by robot is evaluated. The evaluations are done in three different scenes. The results

Table 3.3. Object recognition performance

Scene #i	Object recognition		
	Correct	Wrong	Unknown
1	31	11	6
2	37	10	11
3	17	34	10

are presented in Table 3.3. The performance in first two scenes is quite high. For instance, the second scene contain 46 objects which is manually determined. The robot recognizes 21 of them. The rest are labeled as unknown objects. The robot has found 58 object candidates through an exploration. It is observed that 37 of object candidates are correctly labeled while the labels of 10 are wrong. The remaining objects are labeled as unknown with 8 being so while there are 3 objects that should have been recognized. The average precision in these three scenes is 0.62 which is very close to our class based average precision of 0.56. This actually supports the idea that our performance is highly dependent on the performance of object recognition as discussed. Normally, we expect that there should be some matches around windows; however, as seen in Table 3.2, the performance of window is relatively low. This is because the illumination of windows can change drastically even during a test run.

Next, we consider the construction of the SSMs. First, the effects of the N_r and object candidate proximity threshold τ_d are investigated. We consider $N_r \in \{1, 4, 5, 10\}$ and $\tau_d \in \{5°, 15°, 30°, 40°\}$. The resulting SSMs are showed in Figure 3.5. As expected, decreasing N_r makes the SSM graph structure sparse, whereas increasing N_r makes it dense. The same evaluation applies to the τ_d as well. Hence, it's important to select N_r and τ_d to get a reasonable model. The resulting semantic scene models are studied. For instance, the model of second scene contains 42 nodes as shown in Figure 3.7(a). The robot tends to associate multiple object candidates with larger sized objects (such as floor or wall). Also, neighboring nodes having same labels are merged. It's observed that a balanced graph is observed with $N_r = 4$ and $\tau_d = 15$. Again, the time is

important. Note that it depends on the number of objects in the scene, for this case it takes around 0.037 seconds/objects.

We also consider the merging of a SSM with the current SPM. First, we consider SSMs that are constructed without any tracking across temporary SSMs. The results are as seen in Figure 3.6. It is observed that the merging is such that while common objects are not duplicated, newly appearing objects are added. Of course, the resulting SPM is highly dependent on the object recognition performance. As we discussed previously, the merging performance is also dependent on the recognition performance. For instance, the first two scenes are successfully matched as seen in Figure 3.7(a). The red lines indicate the high score matches; however, the blue lines represent the low score matches with the same labels. The other matches are not showed on the figure, since they are just low score matches with different labels. The corresponding stitched images are also shown in Figure 3.7(b). Yet, the recognition performance in third scene is quite low. So, the matching performance is also low. Hence, the generated semantic place model becomes unreliable. Note that for three scenes, the algorithm merges the chairs and table in the middle of the scenes. The SPM created from these two SSMs is presented in Figure 3.8. It is created from 6 matches which are tables and chairs in

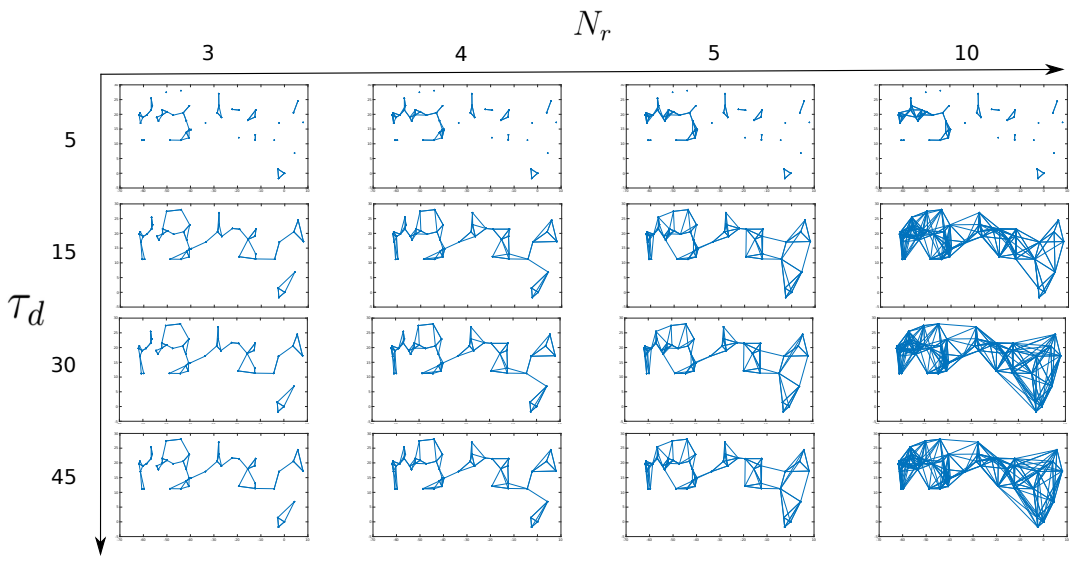


Figure 3.5. SPM with varying N_r and τ_d

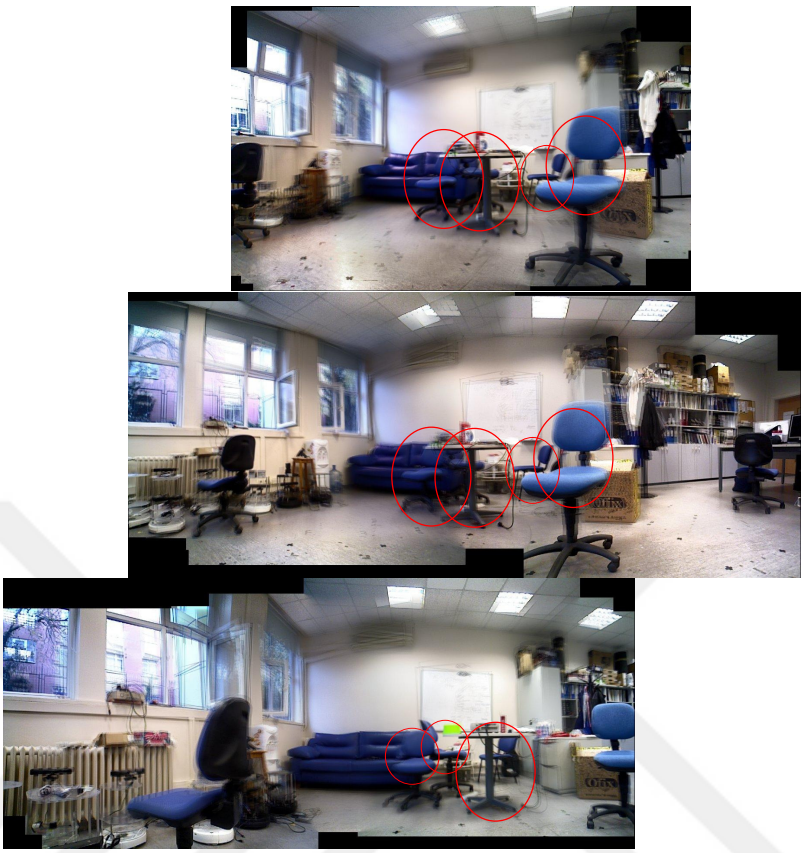
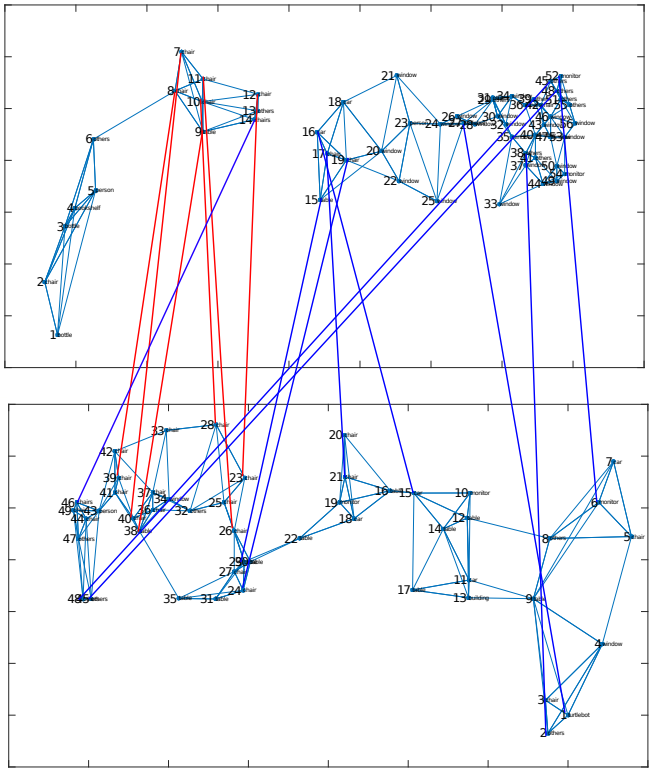


Figure 3.6. Matched objects (red circles) across three different scenes in a place.

the middle of the scene. It contains 100 nodes. 62 of them are correct, 22 of them are 'unknown'. Recall that the labels are not tracked in this experiment, so the number of nodes are quite high compared to other experiments. Table and chairs are found. Windows are also found successfully. There are naturally some wrong labels such as person, even though there is no person in the scene; however, these wrong results can be eliminated by tracking the labels as we discussed before. The time of the matching algorithm depends on the sizes of the graphs to be matched. For the matching of two graphs containing around 40 objects each with $N_r = 4$, it takes 0.189 seconds.

Next, we consider the SPMs that are formed by merging SSMs that are constructed via tracking the coherency of objects. It is observed that object recognition performance is improved as seen in Table 3.4. For example, the percentage of objects that are correctly recognized increases from 62% to 69.5% while the percentage

(a)



(b)

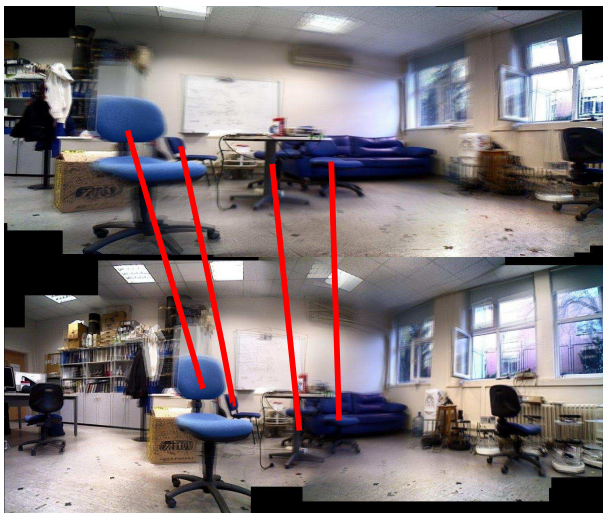


Figure 3.7. Merging the SSM with the SPM. (a) Matching SSM with SPM. Note that some objects are matched (red) while some matches (blue) have higher costs - even though their labels are the same. (b) Matched object candidates are those that continue to be seen in the current scene.

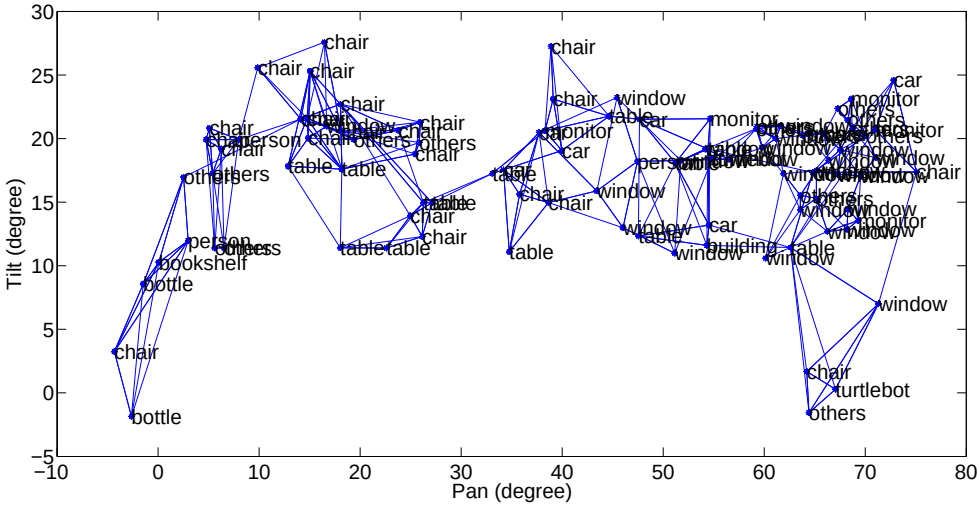


Figure 3.8. Place 1 (Laboratory) : SPM constructed based on SSMs associated with two scenes. The SSMs are formed without tracking the coherency of objects. There are 6 objects that are determined to be common.

of incorrect recognitions drop from 16% down to 7.5%. In the constructed SSMs, our algorithm detected 6 matches. The prominent objects are concentrated around table such as person, chair, monitor. The resulting SPM is also shown in Figure 3.9(a). It covers approximately 110° pan and 40° degree tilt angles and contains 81 objects. This approach reduces the total number of unique objects on model, since the robot does not put the objects on the model, if it is not sure of their labels. Closer inspection reveals that 61 of them are correctly recognized, 7 of them are misclassified and 19 of them are 'unknown'. Hence, the SPM is more accurate in comparison to the SPM that was formed without any tracking of the objects' coherency. Even if SPM is mostly correct, a few of the small black colored segments are misclassified as being 'monitor'due to bad illumination. Also, a segment belongs to white board is misclassified as table for multiple times. The coherency cannot be reached for the windows, since they are dark due to the night. However, the chairs, table, monitor and person are classified correctly and are successfully added to the SPM. Hence, the overall performance with this method improves the performance as much as 48% meaning that SPM is more reliable.

Table 3.4. Object recognition performance in SSMs formed via tracking objects' coherency

Scene #i	Object recognition		
	Correct	Wrong	Unknown
1	24	3	8
2	37	4	11

The experiments are also repeated in a second place (lounge) in order to validate our results in a different place. This scene is shown in Figure 3.10(b). Since it is a lounge, it mostly contains chairs, tables, bookshelf, door, windows and background objects. The illumination is quite satisfactory; however, the windows are too bright that might mislead the recognition algorithm. The resulting SPM is shown in Figure 3.10(a). As seen, the model is quite accurate except the bookshelf part. The chairs and tables are successfully detected; however, the algorithm falsely labels the door as bookshelf for multiple times. The algorithm might be confused due to the wooden shelf-like windows near the door. SPM covers 55° pan and 32° tilt. There are total of 53 objects in the model. 30 of them are 'unknown'. 18 are them correct and 5 are incorrect. In this experiment, again final SPM consists of only temporally coherent object candidates.

There are some important conclusions about the experiments that should be discussed. First, the robot can autonomously create the semantic place models in different places. Second, the robot achieves nearly 90% in laboratory scene and 78% in student lounge scene. The performance is slightly lower in student lounge, since the robot looks at the objects closer in the laboratory scene. This effects the performance, since the objects appears clearly in close up scenes. If the object is far away from camera, there might be some other objects show up in respective object candidate. In general, the performance is quite good and the constructed models are accurate.

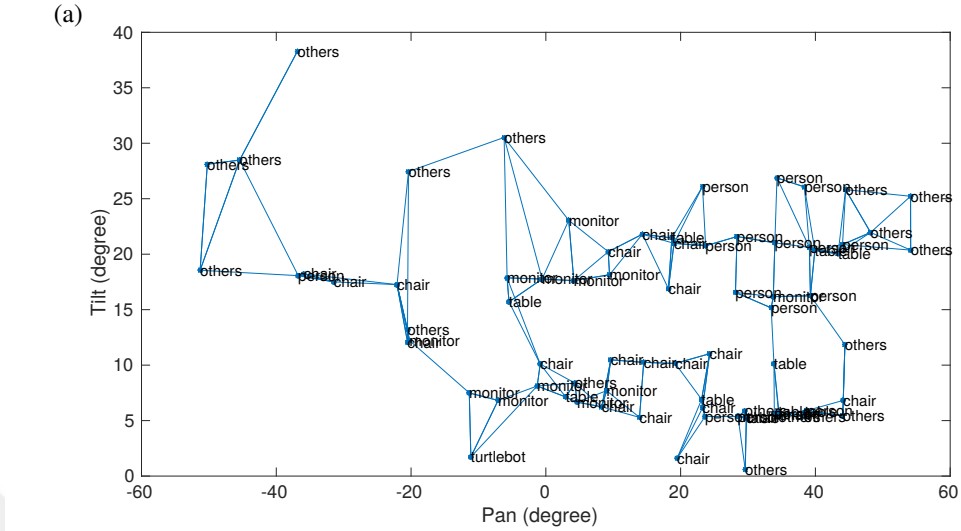




Figure 3.9. Place 1: Constructing a SPM based on SSMs associated with 2 scenes. (a) SPM constructed by merging SSMs from two basepoints containing 6 object matches. (b) SPM coverage as obtained via stitching respective scenes.

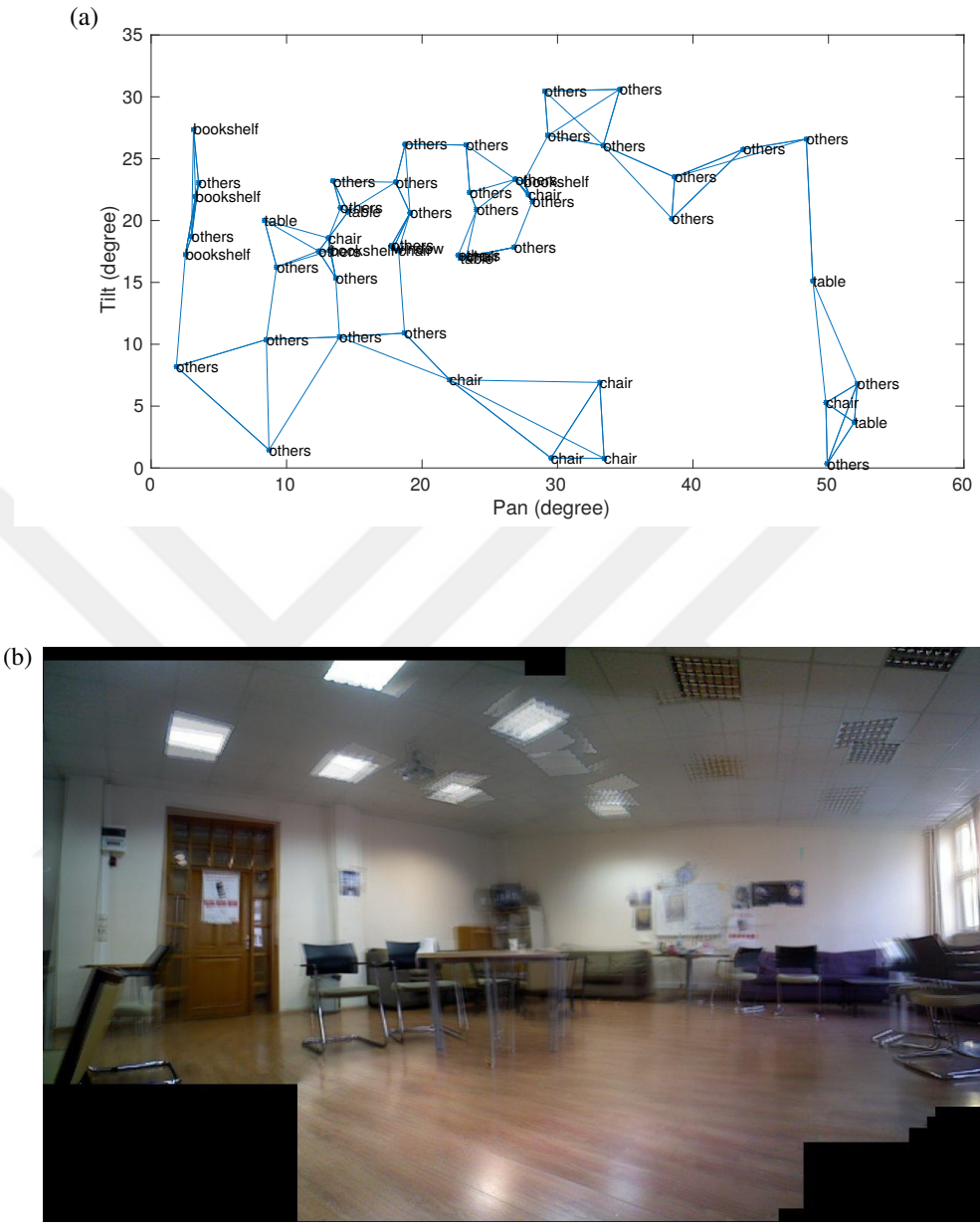


Figure 3.10. Place 2 (Lounge): Constructing a SPM based on SSMs associated with 2 scenes. Here, the SSMs are constructed with the tracking of objects' coherency. (a) SPM via merging two SSMs with 5 common objects. (b) SPM coverage as obtained via stitching respective scenes.

4. UNSUPERVISED OBJECT LEARNING

The object recognition is a crucial step for a robotic application that requires visual reasoning. The aim of object recognition is to associate the object candidates with previously learned models.

Most previous works need human supervision [37]. State-of-the-art deep convolutional neural networks [3,38–40] require high amount of labeled data to achieve a good performance. Although, they beat all the recognition challenges, the labeled dataset dependency avoids researchers from teaching their algorithm high amount of object classes, since preparing a dataset is costly in many ways such as time. Therefore, unsupervised methods have been proposed [41].

In most unsupervised methods, each frame is assumed to contain single dominant object possibly in a cluttered background without any clue about object scale, type or position [42]. One approach considers learning at the feature level only such as learning the codebook associated with the BOW model while the learning of the objects remains supervised [39,43], while [44] proposed to learn higher-level features that results in the learning of individual objects. Finally, most methods focus on learning the object model in an unsupervised manner such as bag-of-words representations [45], constellation of parts [1, 46, 47] or class mask end class edge model [37]. In case of a single unknown class, dense correspondences between images are used to capture the sparsity and visual variability of the common object over the entire database [48]. The idea has been extended to considering a fixed number of predefined categories [42] by using saliency as a guidance. Additionally, [49] uses semantic relation between objects to extend its object library; however it again needs fixed number of categories.

The robotic researchers also consider using object recognition. Generally, on-robot object recognition systems use supervised methods. The most guaranteed way to teach an object to robot is extracting features from everywhere of the object. [50]

uses a turn table to present new objects to the robot. [51] uses low level features to train a fully connected neural network. [13] is another work to use low level features in order to recognize objects in frames collected by the robot. [52, 53] fuse the semantic information to boost the performance of object recognition. The semantic information is given to robot externally. [54] uses a Bayesian classifier along with low level features to recognize the indoor objects. Additionally, since the cloud companies offer object recognition services, [55] uses these recognition services to skip training stage; however, the providers train their architectures with high amount of manually labeled data.

It is surely not possible to train the on-robot recognition system with all objects that the robot might see during its lifetime especially if it is running in an uncontrolled environment. Ideally, the robot should be able to learn new objects as the time flows. This has been considered in robotics via considering unsupervised object learning. [56] learns object models autonomously. They propose a method to detect unknown objects and get multiple views on it to construct a point cloud model of it. [57] uses spatial and semantic relations between objects to make suggestions to learn new object classes. [58] uses a map to detect changes in environment to get unknown objects which are then used to train a segmentation algorithm. Most works require a 3D mapping to detect unknown objects in the scene; however, mapping might not be available. That's why, our proposed work is independent from mapping. Also, rather than creating a model of the new objects, we propose to update the dataset by adding new entries where multiple views of the same object are collected. So, the resulting trained object recognition model becomes more generic.

4.1. Methodology

This thesis also proposes a method that enables robot to add new objects to its dataset in an unsupervised manner. This is done considering object candidates that are labeled as 'unknown'. The following are assumed:

- Initially, the robot has a dataset which is manually labeled in which, there is an 'unknown'class including different object classes and background objects.
- The object candidates are determined as explained in Chapter 2.
- These object candidates are then sent to the object recognition module. For each object candidate, the object recognition module generates both the label and high level feature vector of each object candidate in parallel. This saves processing time, since the robot does not have any additional feature extraction.

The flowchart of the unsupervised object learning is presented in Figure 4.1. The algorithm is as follows:

- If the label is 'unknown', the clustering algorithm incrementally clusters the feature vector with the feature vectors of previous 'unknown'object candidates.
- If the new feature vector belongs to one of the existing clusters, it is added to the corresponding cluster; otherwise, a new cluster is created. After each iteration, the cluster means are updated. This approach is very similar to K-means method,

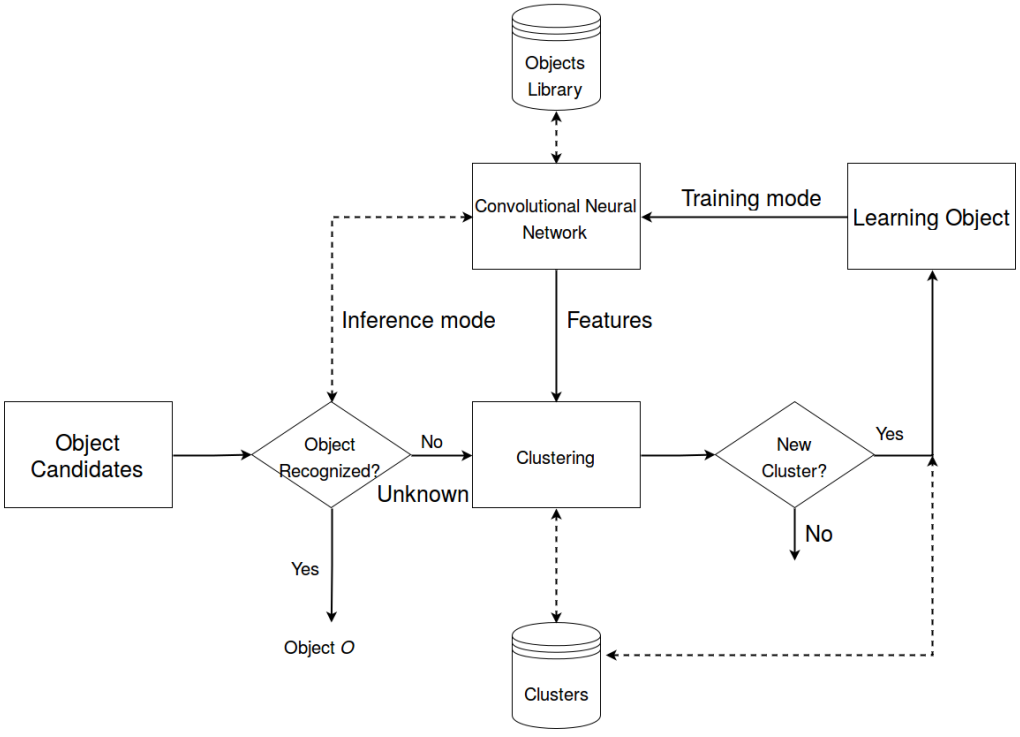


Figure 4.1. Sample images of learned objects.

but our method is incremental, since the object candidates are collected at each fixation.

- The number of object candidates of each clusters are tracked and if one of them exceeds a pre-defined threshold, the robot extracts the cluster and puts it to its existing dataset as a new object class.

4.1.1. Clustering

If the robot is able to recognize the object in the given object candidate, it goes to the next object candidate in the image. If there is no recognition, then it goes into clustering in the appearance space. The goal of this stage is to accumulate learning data over time via clustering in the appearance space. First, each object candidate is encoded by a N_I-dimensional descriptor q. The descriptors can be constructed using a variety of features - as long as similar appearances generate similar descriptors and while being computationally simple to compute for being practically applicable.

In our case, we use 1024-D feature vector of GoogleNet which is taken from last pooling layer of network. The reason that we choose this vector is that the last layers of Convolution Neural Networks extracts the most complex features from images; therefore, these features should represent the objects better than low level ones such as color histogram, edges, textures. Putting the same objects in same cluster is the ultimate goal of this work. That's why, the similarity of images should be defined in highest level possible rather that color or texture similarity. Therefore, these features are highly suitable for our task. The descriptors of a complete dataset can be seen in Figure 4.2. The each row of the image is containing a 1024-D feature vector of an image. In this case, our robot collected 672 images which are labeled as unknown. Another advantage of these features is that they can be extracted at the same time with labeling, so our system does not do extra operation for feature extraction. The visualization of this process can be seen in Figure 4.3.

The descriptor is then considered with respect to the existing N_c clusters in the appearance space. This can be done using any incremental clustering approach. Here, we use a modified version of K-means clustering algorithm. The difference is that the number of clusters N_c is allowed to change while being incremental. The robot assigns the descriptor vector q either to an existing cluster considering its proximity or initiate a new cluster if the closest distance is greater then a pre-specified object clustering threshold $\tau_h > 0$. The parameter τ_h is determined by manually by considering the distances of the samples in the appearance space. This procedure is repeated over time.

4.1.2. Learning

After each addition, existing clusters are checked for their population size. All clusters with at least $\tau_c \in Z^+$ are removed from the appearance space and become input to object learning module. The cluster cardinality threshold τ_c is the minimum number for fine tuning the convolutional neural network for a new object. In this case, fine-tuning of the GoogleNet is repeated. Let us note that since fine-tuning is not incremental, the data of previously learned objects need to be input as well. However, in future, if fine-tuning can become incremental, this need not to be the case.

4.2. Experiments

The experiments are done with our APES robot that has a looking around ability with a pan tilt camera. As the experiments are done indoors, the places contain mostly objects such as table, monitor, human. At the onset, the robot knows 14 object classes. As explained, it has learned these objects in a supervised manner using the deep learning library Caffe [34]. For the training part, we are using GPU; however, the robot uses CPU for the inference part.

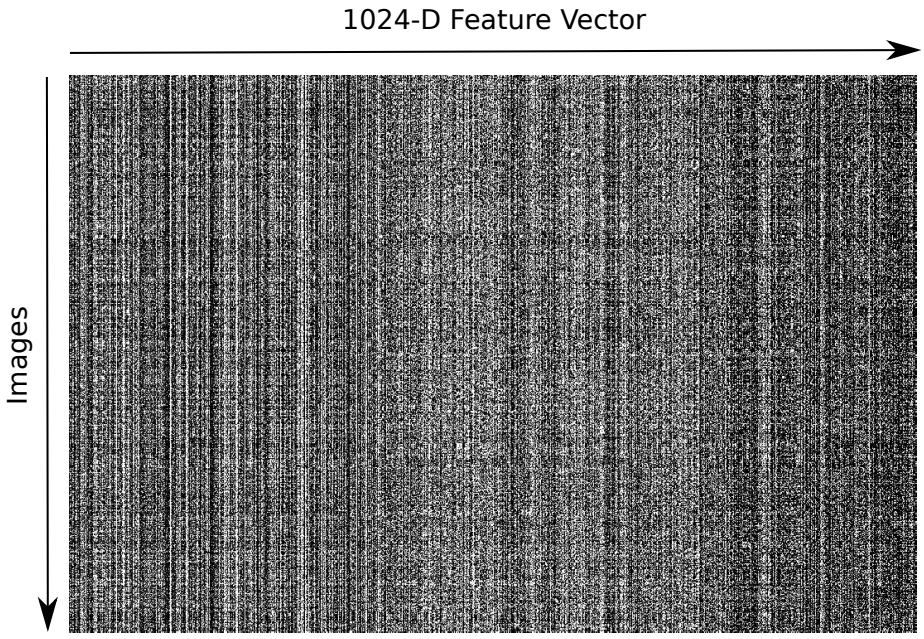


Figure 4.2. Stacked feature vectors of 672 object candidates. Each column is representing a feature of all images and each row is containing a feature vector of each image.

The robot runs over several hours for several days in order to get a huge number of object candidates - since clustering needs a lot of unknown objects. The object clustering threshold is set as $\tau_h = 10$. The minimum cluster cardinality is $\tau_c = 150$. Some of the clusters are demonstrated in Figure 4.4. Note that these are just for demonstration purposes, so the clusters contains high number of images. The number of images that each cluster contains varies. Most of the clusters can only get up to 30-40 object candidates which is not considered as a significant cluster, since there may be some errors in clustering. Hence, resulting cluster cannot collect more object candidates, because there is a small similarity between features of object candidates in it. It is observed that the two clusters (robots and window) end up having over τ_c samples. These clusters are added to the training set. To have a balanced training set, typically a fine-tuning is done with 100-150 object candidates per class. After fine tuning, of course, there may be some mis-classifications for newly added classes. However, their numbers typically do not exceed the threshold, since their recognition

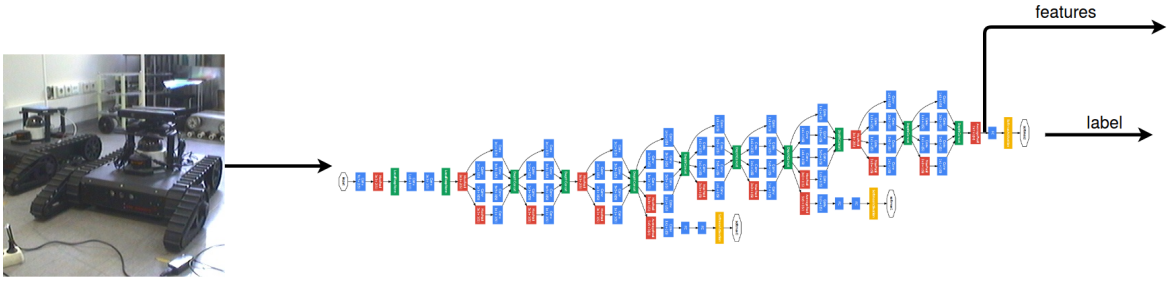


Figure 4.3. Usage of GoogleNet both as a classifier and a feature extractor. The image of GoogleNet is taken from [3].

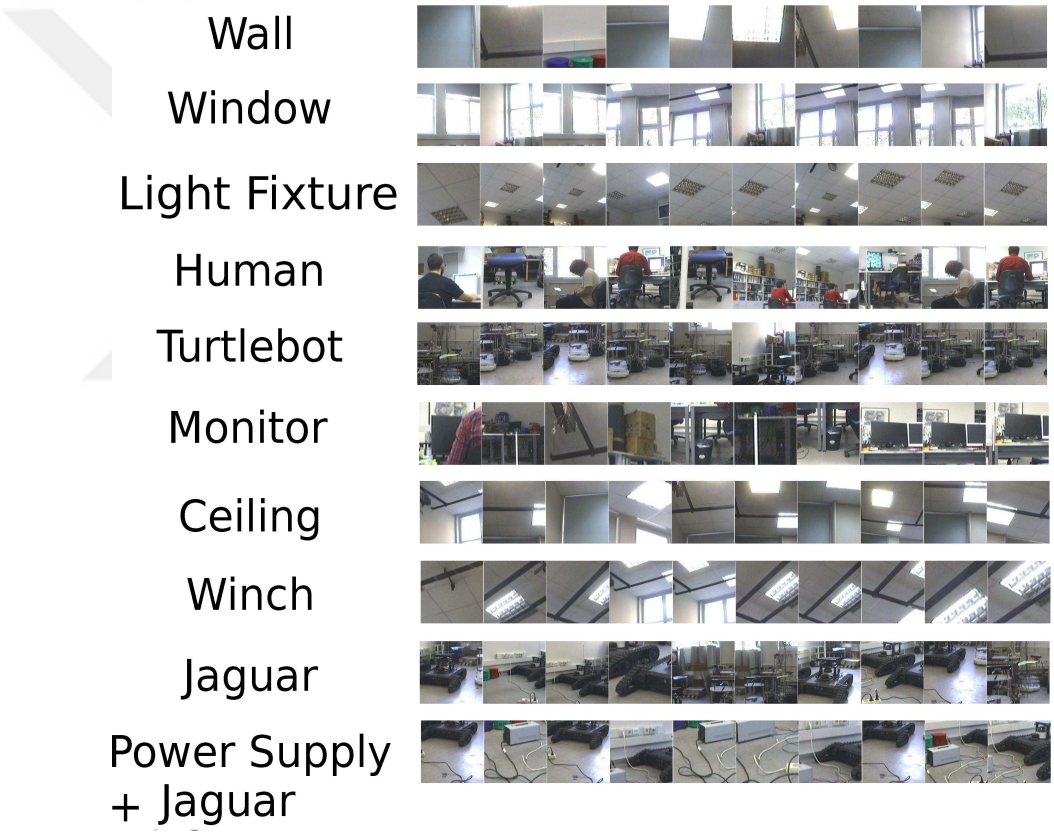


Figure 4.4. Sample images from the resulting clusters in the appearance space. Two clusters have samples that exceed τ_c.

performance is quite high, hence there is a small chance them to be labeled as unknown again.

We evaluate how well the robot is able to learn new objects by studying their recognition performance in comparison to that of supervised learning. The results can be seen in Table 4.1. It is observed that unsupervised module performs quite satisfactory and it actually performs the same with supervised module, since the quality of window cluster is quite high; however, supervised learning performs better than unsupervised learning for robots class.

Table 4.1. Comparative recognition precision.

Learning	Learned Object	
	Window	Robot
Supervised	0.76	0.89
Unsupervised	0.76	0.85

5. CONCLUSION

This thesis presents a novel approach to the construction of semantic place models. Our first contribution is coupling the object discovery and visual scene exploration. In this approach, the robot discovers objects and constructs a family of APF to move its head through the most important parts of the scene. As the robot does not look the same object twice by using an inhibition mechanism, our exploration is more efficient as compared to an exhaustive search approach. In the second part of the thesis, we show that object candidates and the respective camera movements can be used in constructing a semantic place model. In this approach, the object candidates are labeled semantically by using an object recognition system. This is used to update the semantic place model by using a simple merging algorithm. This makes it possible to merge not only multitude of views, but also views from different location of the robot's base. We also experimentally showed that the object recognition performance is comparable to the state-of-the art. Yet, our performance is still highly dependent on the performance of object recognition module.

Finally, in the scope of this thesis, we consider how to enable a robot to become aware of objects in a place based on their appearances. To do this, we proposed to use high level features of unknown objects to cluster them in appearance space which allows robot to add new objects into its training set in an unsupervised manner. In our experiments, we showed that our robot can successfully add two new objects with comparable performance to the supervised learning.

For future work, we will focus on increasing the reliability and performance of the resulting SPMs, since the recognition performance is important for both model reliability and matching performance. Also, we will focus on using these constructed semantic place models in various robotics applications such as intelligently searching for an object in a place in order to enable robot to manipulate it.

REFERENCES

1. Fergus, R., P. Perona and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning", *IEEE Conference on Computer Vision And Pattern Recognition*, Vol. 2, pp. 264–271, 2003.

2. Biederman, I., "Recognition-by-components: A theory of human image understanding", *Psychological Review*, Vol. 94, pp. 115–147, 1987.

3. Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

4. Krizhevsky, A., I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012.

5. Rensink, R. A., "The Dynamic Representation of Scenes", *Visual Cognition*, Vol. 7, No. 1-3, pp. 17–42, 2000.

6. Hosang, J., R. Benenson and B. Schiele, "How good are detection proposals, really?", *Proceedings of the British Machine Vision Conference*, BMVA Press, 2014.

7. Scholl, B. J., "Objects and attention: the state of the art", *Cognition*, Vol. 80, No. 1–2, pp. 1 – 46, 2001.

8. Davis G., D. J., R. C., T. M. and F. E., "Segmentation, attention, and phenomenal visual objects", *Cognition*, Vol. 80, No. 1–2, pp. 61 – 95, 2001.

9. 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.

10. Alexe, B., T. Deselaers and V. Ferrari, "Measuring the Objectness of Image Windows", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 11, pp. 2189–2202, 2012.

11. Carreira, J. and C. Sminchisescu, "CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 7, pp. 1312–1328, 2012.

12. Endres, I. and D. Hoiem, "Category-Independent Object Proposals with Diverse Ranking", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 36, No. 2, pp. 222–234, 2014.

13. 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.

14. Pillai, S. and J. Leonard, "Monocular SLAM Supported Object Recognition", *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

15. 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 Conferrence on Robotics and Automation*, pp. 127–134, 2015.

16. 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", *Proceedings of Robotics: Science and Systems*, 2016.

17. Demir, M. and H. Bozma, "Video Summarization via Segments Summary Graphs", *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 19–25, 2015.

18. Hwang, A. D., H.-C. Wang and M. Pomplun, "Semantic guidance of eye movements in real-world scenes", *Vision Research*, Vol. 51, No. 10, pp. 1192 – 1205, 2011.

19. Pajak, M. and A. Nuthmann, "Object-based saccadic selection during scene perception: Evidence from viewing position effects", *Journal of Vision*, Vol. 13, No. 5, p. 2, 2013.

20. Bindemann, M., "Scene and screen center bias early eye movements in scene viewing", *Vision Research*, Vol. 50, No. 23, pp. 2577 – 2587, 2010.

21. Liversedge, S., I. Gilchrist and S. Everling (Editors), *The Oxford Handbook of Eye Movements*, Oxford University Press, 2013.

22. 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", *Proceedings of Robotics: Science and Systems*, Ann Arbor, Michigan, 2016.

23. Ranganathan, A. and F. Dellaert, "Semantic modeling of places using objects", *Proceedings of the 2007 Robotics: Science and Systems Conference*, Vol. 3, pp. 27–30, 2007.

24. Vasudevan, S., S. Gachter, V. 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.

25. Zender, H., O. M. Mozos, P. Jensfelt, G.-J. Kruijff and W. Burgard, "Conceptual spatial representations for indoor mobile robots", *Robotics and Autonomous Systems*, Vol. 56, No. 6, pp. 493–502, 2008.

26. Anand, A., H. S. Koppula, T. Joachims and A. Saxena, "Contextually guided semantic labeling and search for three-dimensional point clouds", *The International Journal of Robotics Research*, Vol. 32, No. 1, pp. 19–34, 2013.

27. Posner, I., M. Cummins and P. Newman, "A generative framework for fast urban labeling using spatial and temporal context", *Autonomous Robots*, Vol. 26, No. 2-3, pp. 153–170, 2009.

28. Cadena, C. and J. Košecká, "Semantic parsing for priming object detection in indoors RGB-D scenes", *The International Journal of Robotics Research*, 2014.

29. Li, L.-J., R. Socher and L. Fei-Fei, "Towards total scene understanding: Classification, annotation and segmentation in an automatic framework", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2036–2043, 2009.

30. Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

31. Yang, J., Y.-G. Jiang, A. G. Hauptmann and C.-W. Ngo, "Evaluating bag-of-visual-words representations in scene classification", *Proceedings of the international workshop on multimedia information retrieval*, pp. 197–206, 2007.

32. Preibisch, S., S. Saalfeld and P. Tomancak, "Globally optimal stitching of tiled 3D microscopic image acquisitions", *Bioinformatics*, Vol. 25, No. 11, pp. 1463–1465, 2009.

33. Kuhn, H. W., "The Hungarian method for the assignment problem", *Naval research logistics quarterly*, Vol. 2, No. 1-2, pp. 83–97, 1955.

34. Jia, Y., E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding", *arXiv preprint arXiv:1408.5093*, 2014.

35. Russakovsky, O., Y. Lin, K. Yu and L. Fei-Fei, "Object-centric spatial pooling for image classification", *European Conference on Computer Vision*, pp. 1–15, 2012.

36. Everingham, M., S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective", *International Journal of Computer Vision*, Vol. 111, No. 1, pp. 98–136, 2015.

37. Winn, J. and N. Jojic, "LOCUS: learning object classes with unsupervised segmentation", *IEEE International Conference on Computer Vision*, Vol. 1, pp. 756–763, 2005.

38. Simonyan, K. and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.

39. Eitel, A., J. T. Springenberg, L. Spinello, M. Riedmiller and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 681–687, 2015.

40. He, K., X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

41. Oliveira, M., L. S. Lopes, G. H. Lim, S. H. Kasaei, A. D. Sappa and A. M. Tomé, "Concurrent learning of visual codebooks and object categories in open-ended domains", *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2488–2495, 2015.

42. Zhu, J. Y., J. Wu, Y. Xu, E. Chang and Z. Tu, "Unsupervised Object Class Discovery via Saliency-Guided Multiple Class Learning", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 37, No. 4, pp. 862–875, 2015.

43. Coates, A. and A. Y. Ng, "Learning feature representations with k-means", *Neural Networks: Tricks of the Trade*, pp. 561–580, Springer, 2012.

44. Le, Q. V., "Building high-level features using large scale unsupervised learning", *IEEE International Conference On Acoustics, Speech and Signal Proccesing*, pp. 8595–8598, 2013.

45. 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.

46. Weber, M., M. Welling and P. Perona, *Unsupervised Learning of Models for Recognition*, Vol. 1, pp. 18–32, Springer, 2000.

47. Loeff, N., H. Arora, A. Sorokin and D. Forsyth, "Efficient Unsupervised Learning for Localization and Detection in Object Categories", *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, pp. 811–818, 2005.

48. Rubinstein, M., A. Joulin, J. Kopf and C. Liu, "Unsupervised Joint Object Discovery and Segmentation in Internet Images", *IEEE Conference on Computer Vision And Pattern Recognition*, 2013.

49. Lee, Y. J. and K. Grauman, "Object-graphs for context-aware visual category discovery", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 34, No. 2, pp. 346–358, 2012.

50. Vincze, M., M. Bajones, M. Suchi, D. Wolf, A. Weiss, D. Fischinger and P. da la Puente, "Learning and Detecting Objects with a Mobile Robot to Assist Older Adults in Their Homes", *European Conference on Computer Vision*, pp. 316–330, Springer, 2016.

51. Caron, L.-C., D. Filliat and A. Gepperth, "Neural network fusion of color, depth and location for object instance recognition on a mobile robot", *European Conference on Computer Vision*, pp. 791–805, Springer, 2014.

52. Ruiz-Sarmiento, J.-R., C. Galindo and J. Gonzalez-Jimenez, "Exploiting semantic knowledge for robot object recognition", *Knowledge-Based Systems*, Vol. 86, pp. 131–142, 2015.

53. Ruiz-Sarmiento, J.-R., C. Galindo and J. Gonzalez-Jimenez, "Scene object recognition for mobile robots through Semantic Knowledge and Probabilistic Graphical Models", *Expert Systems with Applications*, Vol. 42, No. 22, pp. 8805–8816, 2015.

54. Espinace, P., T. Kollar, N. Roy and A. Soto, "Indoor scene recognition by a mobile robot through adaptive object detection", *Robotics and Autonomous Systems*, Vol. 61, No. 9, pp. 932–947, 2013.

55. Kehoe, B., A. Matsukawa, S. Candido, J. Kuffner and K. Goldberg, "Cloud-based robot grasping with the google object recognition engine", *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 4263–4270, IEEE, 2013.

56. Fäulhammer, T., R. Ambruş, C. Burbridge, M. Zillich, J. Folkesson, N. Hawes, P. Jensfelt and M. Vincze, "Autonomous learning of object models on a mobile robot", *IEEE Robotics and Automation Letters*, Vol. 2, No. 1, pp. 26–33, 2017.

57. Young, J., V. Basile, L. Kunze, E. Cabrio and N. Hawes, "Towards Lifelong Object Learning by Integrating Situated Robot Perception and Semantic Web Mining", *Proceedings of the European Conference on Artificial Intelligence (ECAI) 2016 conference*, The Hague, Netherlands, 2016.

58. Finman, R., T. Whelan, M. Kaess and J. J. Leonard, "Toward lifelong object segmentation from change detection in dense RGB-D maps", *European Conference on Mobile Robots (ECMR)*, pp. 178–185, IEEE, 2013.

59. Quigley, M., K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, "ROS: an open-source Robot Operating System", *International Conference on Robotics and Automation Workshop on Open Source Software*, Vol. 3, p. 5, 2009.

60. Andreopoulos, A. and J. K. Tsotsos, "50 Years of object recognition: Directions forward", *Computer Vision and Image Understanding*, Vol. 117, No. 8, pp. 827 – 891, 2013.

61. Felzenszwalb, P. F. and D. P. Huttenlocher, "Efficient graph-based image segmentation", *International Journal of Computer Vision*, Vol. 59, No. 2, pp. 167–181, 2004.

62. Jouili, S., I. Mili and S. Tabbone, "Attributed graph matching using local descriptions", *International Conference on Advanced Concepts for Intelligent Vision Systems*, pp. 89–99, Springer, 2009.

APPENDIX A: HARDWARE & SOFTWARE

This section explains the hardware and software aspects of the work done in this thesis.

A.1. Hardware

The proposed approaches have been implemented on the APES robot as shown in Figure A.2. The robot has the following hardware components:

- The robot has a kobuki (turtlebot) base.
- It has a pan-tilt head with a 3D Kinect sensor(Asus Xtion). The camera attachment has been designed and 3D-printed. The drawing is shown in Figure A.4. It is designed to have a flat surface where Asus Xtion is placed. Also, it has small holes on its each corner; so that, the camera can be locked safely. The pan-tilt motors are controlled by a Arduino card along with an Arduino Motor Shield.
- The robot has a Intel i5 2.4 GHz as its processor. The processor is connected to both the Kinect sensor(Asus Xtion) and the Arduino card.

The electrical and electronic connections are as shown in Figure A.1. If all the connections are done as shown in this diagram, the robot should start. Do not connect any additional power source, the robot uses single power source since it has a motor shield.

A.2. Hardware Setup

The hardware setup is as follows:

- Just check the connection between Arduino and power source. Connect the power supply to the robot. The robot has two cables(red and black). Plug these cables

to the plugs of the power supply. Set the voltage to 12V.

- Connect Arduino, Asus Xtion to the USB ports of the laptop .
- If you want to move the robot base, you should also connect the kobuki base to the laptop.
- Turn on the kobuki base. If the robot is ready, the Kobuki base will make a beep sound.

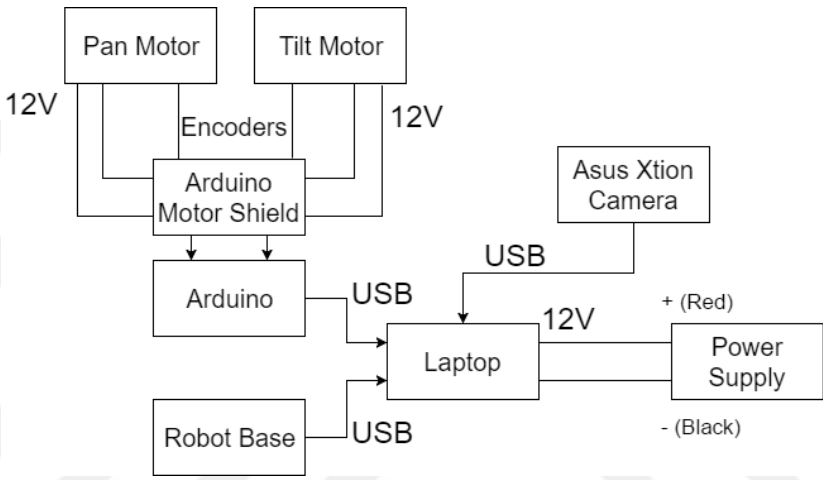


Figure A.1. Electrical and electronic connections on the robot

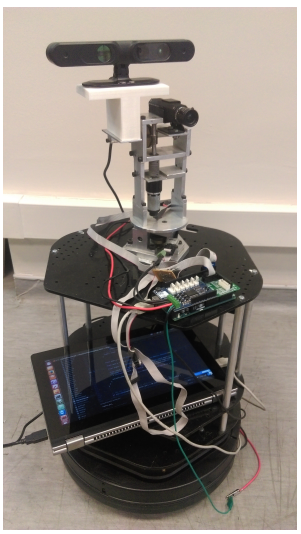


Figure A.2. The mobile robot with a pan-tilt camera.

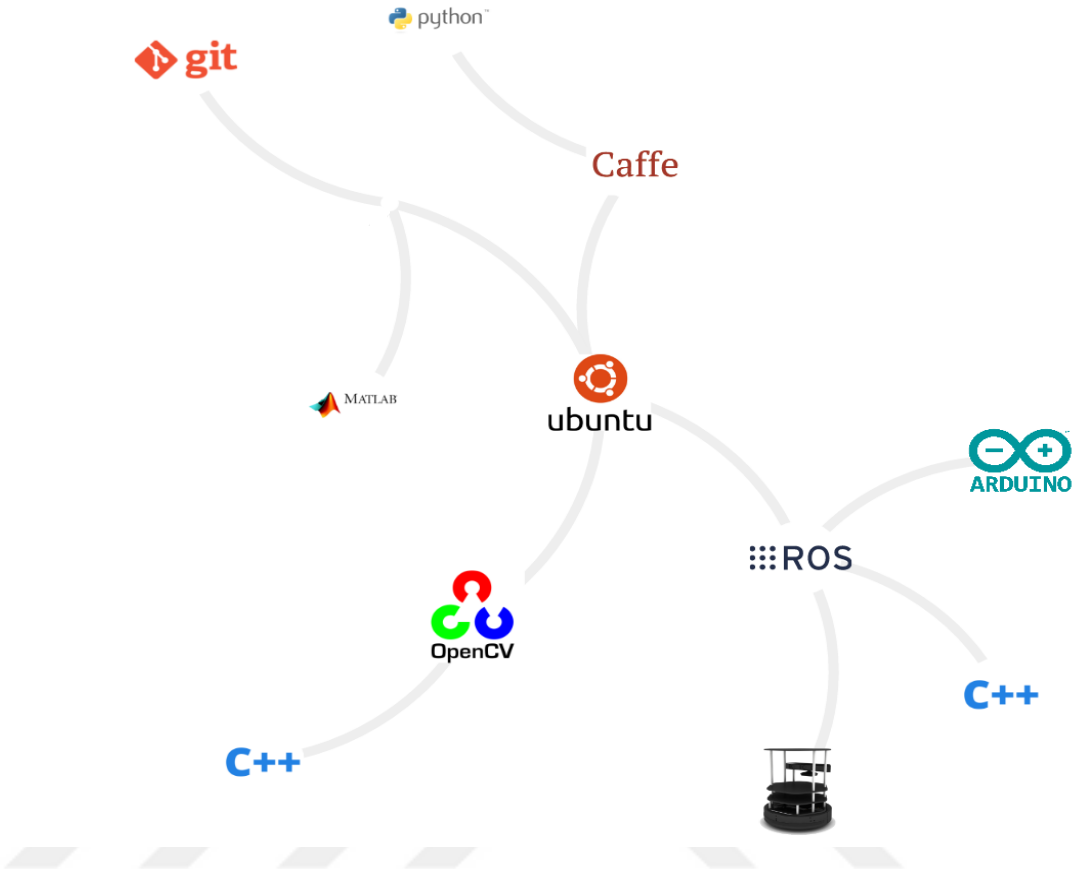


Figure A.3. The technologies used in these thesis: software libraries, programming languages and hardwares

A.3. Required Software & Libraries

The robot is programmed within C++ / ROS [59] framework. There are some scripts which are written in Python - as Python has superior libraries such as Numpy, Matplotlib, Scipy which enable us to use MATLAB-like matrix operations. Python scripts can be found under each ROS packages' scripts sub folder.

OpenCV is the best library in vision and image processing areas. It includes all the core algorithms in an efficient manner. We use its C++ API to do image processing tasks. It also makes easier to work on matrix in C++ via its MATLAB-like matrix syntax.

For object learning, we use the deep learning Caffe [34]. It can be run via command line, but it also has a Python API that makes integration easier. Another reason that we selected is that it has a huge community and also a model zoo where you can download the others' models and run them on your machine. In this way, it is possbile to integrate state of the art methods in our system with minimal effort.

Digits is a deep learning platform which eases the training and testing of object recognition. It uses Caffe as backend. So, its models can be used in any computer where Caffe is installed. It also produces nice reports and visualization about training and testing procedures.

ImageJ is used for batch operations on images. It has huge community support. Additionally, it includes implementation of state-of-the-art algorithms as Image Stitching as used in the theis.

Inkscape is used to create visualization. Nearly all graphs in this thesis are post-processed by Inkscape.

A.3.1. Software Design

APES has been incrementally built, there are codes written by other former lab members. Also, all the codes are not in single node for modularity. The major classes of ros-node-attentive-robot are:

- Robot.h is a sort of container class including the pan-tilt values, turn number and robot base position. Also, it has a sub-class called util which includes some utilities for APES.
- Place.h is one of the most critical class. It contains the Scene.h and has all the utilities to save all places in an object.
- Scene.h is a member of Place.h. It includes graph model information constructed from a specific place.

- ObjectProcessor.h is containing the APF creation functions.
- FixationExtractor.h is used to extract fixations from images.

Git is used for version controlling system. If you want to analyze my codes in detail, it is highly recommended to do it by git. All the Github repos can be seen in Table A.1. To download them use: git clone https://github.com/cagbal/repo_name.git

Table A.1. Github repos written in the scope of this thesis

Function	Github Repo Name
Object Recognition, Incremental Clustering, feature extraction	https://github.com/cagbal/isl_object_recognition.git
ROS Attention Node	https://github.com/cagbal/ros-node-attentive-robot.git

A.3.2. Running the Software

The steps below are required to train a new object recognition model:

(i) open a terminal
(ii) write cd /code/digits_clones/dev/DIGITS
(iii) write ./digits-devserver
(iv) now, open a web browser and go to http://0.0.0.0:5000

The following steps are required in order to connect to the robot. To start a new experiment:

(i) open a terminal and write roscore
(ii) write rosrun apes_arduino apes_arduino_node
(iii) write rosrun apes_head apes_head_node
(iv) write rosrun attentive_robot attentive_robot_node

(v) write roslaunch openni2_launch openni2.launch

(vi) write rosrun object_recognition object_recognition_node

(vii) write rosrun object_recognition clustering

(viii) write rosrun spm spm_node

(ix) collect the data from /home/turtlebot/cagatay_ws/src/attentive_robot

(x) if you want to get the recognition results, subscribe to object_recognition/label and publish to apes_robot/fixation_point where the robot already publishes to this topic to get label

All models, datasets can be seen from that panel. If you want to update the model on APES. Train a new model and download it. Put the downloaded model to the model folder inside the object recognition module.

Also, the parameters used during the experiments are presented in Table A.2.

Table A.2. Parameter List

Parameter	Name	Unit	Chapter	Typical Value
ϵ	S^2 proximity threshold	Degree	2,3	25
τ_d	Object candidate S^2 proximity threshold	Degree	3	15
N_r	Spatial relations threshold	-	3	4
τ_j	Object candidate duration threshold	-	3	4
$\alpha_{1,2,3}$	Weights of terms of SSM-SPM matching cost function	-	3	1, 0.2, 0.2
τ_m	Object matching cost threshold	2	3	
τ_h	Object clustering threshold	-	4	10
τ_c	Cluster cardinality threshold	-	4	150

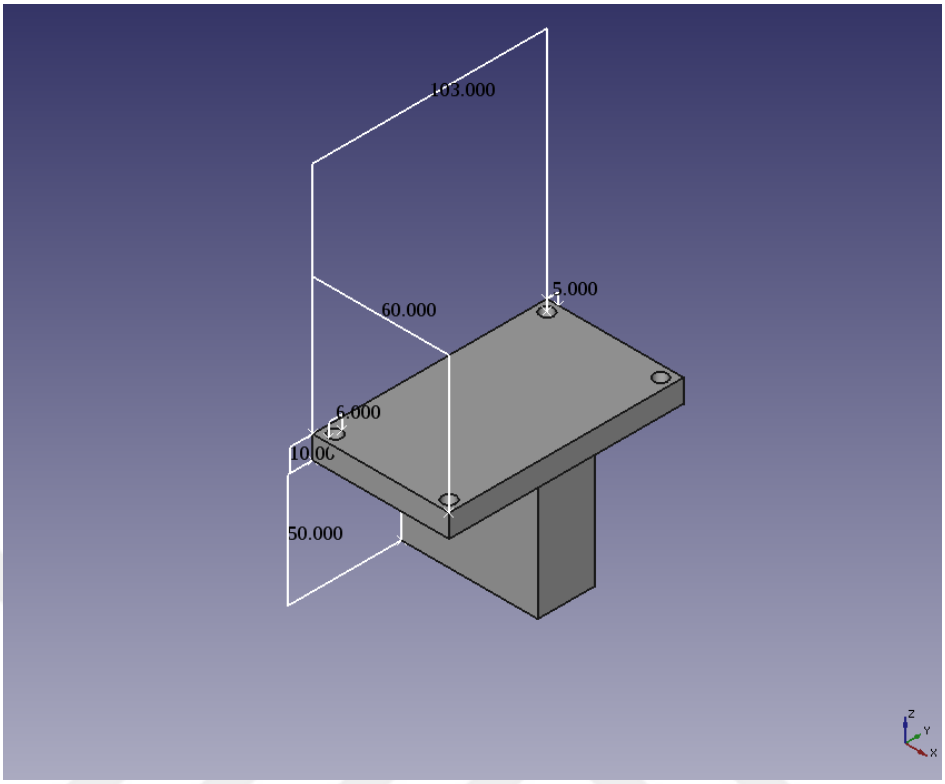


Figure A.4. 3D Drawing of the camera attachment. All dimensions are millimeter.

APPENDIX B: OBJECT DISCOVERY

Finding object candidates is crucial for this thesis to decide where to move the robot's head. Low level segments are tracked temporally and if a segment appears consistently over a temporal window, it is determined as spatio-temporally coherent segment and hence it is worth looking at. It may be an important entity of the place. For this, we adapt a method that has been proposed for generating video summaries [17] as showed in Figure B.1.

First, the incoming visual data is segmented into homogeneous color regions $\mathcal{S}^l = \{S_i^l\}_{i=1}^{N_l}$ with cardinality N_l. Note that segmentation is common as an initial step to most object discovery and recognition methods [60]. Thus, any off-the-shelf segmentation method with a low complexity can be used. In this work, we use the graph-based segmentation algorithm [61] which is known to be one of the best performing algorithms that is also computationally efficient ($O(n \log n)$ time for n image pixels) [36]. It also allows us to control size and structure of the segment by changing parameters. Hence, it is suitable for on-robot applications. In this algorithm, the number and size of the segments generated depend on three parameters: smoothing factor σ, merging threshold τ_m and minimum segment size δ. As is the case with most work, these parameters are carefully tuned as to have segments that are as large and few as possible while retaining their consistency. They are set as $\sigma = 0.7$, $\tau_m = 150$ and $\delta = 1000$.

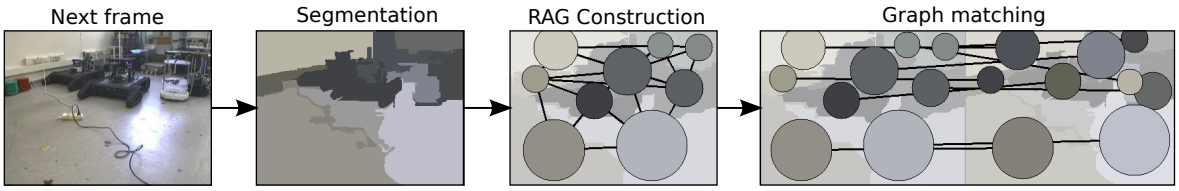


Figure B.1. Determining object candidates.

Next, the segmented regions and their relationships ar by the nodes and the edges of a region adjacency graph (RAG) $G^l = (\mathcal{S}^l, \mathcal{E}^l, \mathcal{A}^l)$ where $\mathcal{S}^l$ is the set of nodes, $\mathcal{E}^l$ is the edge set and $\mathcal{A}^l$ is the attribute set that contains attributes related to nodes and edges. Each segment $S_i^l \in \mathcal{S}^l$ is associated with a node and spatial relationship between segments is encoded by the edges $\mathcal{E}_{ij}^l$. A node is associated with a set of attributes as derived from the respective segment $\mathcal{S}_i^l$. These are its area $a(S_i^l) > 0$, centroid $\mu(S_i^l) \in P^l$ where P^l is the set of object candidate centroids in l^{th} frame, an enclosing disk as defined by its radius $\rho(S_i^l) > 0$ and mean color $c(S_i^l)$.

The third step is to match a newly formed RAG G^l with preceding RAGs $G^{l'}$, $l' < l$. This enables the robot to track segments that have appeared previously as well as determining newly appearing segments. Their labels can then be assigned accordingly. This part is motivated by work on generating video summaries [17, 62]. The graph matching algorithm is based on the distance matrix $C^{ll'}$ with elements $C_{ij}^{ll'}$ defined based on weighted Manhattan distance between the node signatures $s(S_i^l)$ and $s(S_j^{l'})$:

$$C_{ij}^{ll'} = \left\| s(\mathcal{S}_i^l) - s(\mathcal{S}_j^{l'}) \right\|_w \tag{B.1}$$

where $w = [w_p, w_a, w_c, w_e]^T$ is a weight vector which encodes the significance of each type of attribute, namely position, area, color and edge, to be used in graph matching. In particular, $w = \begin{bmatrix} 0.8 & 0.7 & 0.5 & 0 \end{bmatrix}$, thus the position and area similarity of segments are relatively weighted more in comparison to color and edge attributes. The resulting permutation $\pi^{ll'}$ defines one-to-one optimal matching between the nodes of two graphs. However, in practice, some of these assignments may lead to wrong matches with high costs. In order to minimize such cases, only assignments $\pi^{ll'}(i)$ with matching cost $C_{i\pi^{ll'}(i)}^{ll'}$ less than the segment matching threshold τ_s are considered to be valid matches. The same matching procedure is applied to previous graphs to check if any of unassigned nodes contains a match. As such, segments can be tracked even after a short disappearance. Finally, object candidates $\tilde{\mathcal{S}}^l$ are determined via tracking

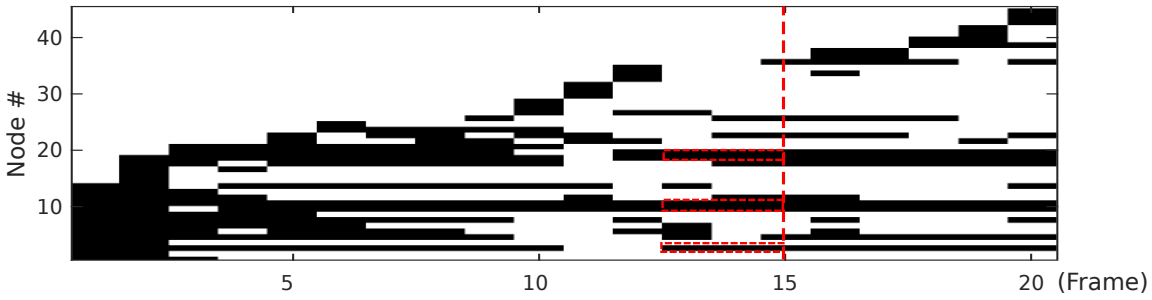


Figure B.2. Node existence matrix. If the robot's input highlight spatio-temporally coherent segments.

spatio-temporally coherent segments over time. To do this, an evolving node existence matrix $\mathbf{M}$ encoding the results of the matching for each RAG across the base points $\mathcal{K}$ is maintained. For each $l \in \mathcal{L}$, the matrix entry $M_{li} = j$ if the segment S_j^l is matched to the segment S_i^m for some $l - \tau_w < m < l$. The window parameter τ_w determines the size of the temporal window meaning that it determines how many previous frames are checked. As such, its value is closely related with the frame rate and saccade velocity. As they increase, so should τ_w. Our robot's has a frame rate of 1 frames per second, so our window parameter τ_w is set as 10. This enables the robot can track the disappearing object candidates for a shorter time periods than τ_w.

If there is no matching, the associated node is added to the matrix as a new y-axis entry. Object candidates whose life spans exceed a pre-defined threshold τ_s are determined to be spatio-temporally coherent. A larger τ_s leads to a more strict coherency, since the object candidates must be seen for a longer time frame in order to be determined as spatio-temporally coherent. Here, it is $\tau_s = 5$.