UNIVERSITY OF TECHNOLOGY, SYDNEY

Affordance-Map : Learning Hidden Human Context in 3D Scenes Through Virtual Human Models

by

Jayaweera Mudiyanselage Lasitha Chandana Piyathilaka

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Faculty of Engineering and IT
Centre of Autonomous Systems

February 2016

Declaration of Authorship

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student:

Date:

UNIVERSITY OF TECHNOLOGY, SYDNEY

Abstract

Faculty of Engineering and IT

Centre of Autonomous Systems

Doctor of Philosophy

by Jayaweera Mudiyanselage Lasitha Chandana Piyathilaka

Ability to learn human context in an environment could be one of the most desired fundamental abilities that a robot should possess when sharing workspaces with human co-workers. Arguably, a robot with appropriate human context awareness could lead to a better human robot interaction. This thesis addresses the problem of learning human context in indoor environments by only looking at geometrics features of the environment. The novelty of this concept is, it does not require to observe real humans to learn human context. Instead, it uses virtual human models and their relationships with the environment to map hidden human affordances in 3D scenes.

The problem of affordance mapping is formulated as a multi label classification problem with a binary classifier for each affordance type. The initial experiments proved that the SVM classifier is ideally suited for affordance mapping. However, SVM classifier recorded sub-optimum results when trained with imbalanced datasets. This imbalance occurs because in all 3D scenes in the dataset, the number of negative examples outnumbered positive examples by a great margin. As a solution to this, a number of SVM learners that are designed to tolerate class imbalance problem are tested for learning the affordance-map. These algorithms showed some tolerance to moderate class imbalances, but failed to perform well in some affordance types.

To mitigate these drawbacks, this thesis proposes the use of Structured SVM (S-SVM) optimized for F1-score. This approach defines the affordance-map building problems as a structured learning problem and outputs the most optimum affordance-map for a given set of features (3D-Images). In addition, S-SVM can be learned efficiently even on a large extremely imbalanced dataset. Further, experimental results of the S-SVM method outperformed previously used classifiers for mapping affordances.

Finally, this thesis presents two applications of the affordance-map. In the first application, affordance-map is used by a mobile robot to actively search for computer monitors in an office environment. The orientation and location information of humans models inferred by the affordance-map is used in this application to predict probable locations of computer monitors. The experimental results in a large office environment proved that the affordance-map concept simplifies the search strategy of the robot. In the second application, affordance-map is used for context aware path planning. In this application, human

context information of the affordance-map is used by a service robot to plan paths with minimal distractions to office workers.

Acknowledgements

I am sincerely in debt to my supervisor, Associate Professor Sarath Kodagoda for believing in me and for giving me enough flexibility to work on my lifelong passion. His continuous support and guidance made this difficult endeavor achievable.

I would also like to thank Professor Massimo Piccardi for the fruitful discussions we had together. He always had enough time for me in his busy schedule.

To all the academic and support staff at CAS, thanks for taking me in and making me part of the family. Never have I felt more at home than in these past four years.

I would also like to thank Australian Government and University of Technology Sydney for awarding me with International Postgraduate Research Scholarship (IPRS) and Australian Postgraduate Award (APA). Without these scholarships I certainly would have not studied for a PhD.

Finally, I am ever in debt to my family. Specially my parents who made great sacrifices to make me who am I today. A very special thanks to my wife Uthsu for continuously believing me in and encouraging me in this difficult journey. Thanks my lovely daughter Sethmi for making my life joyful and meaningful.

Contents

List of Figures

List of Tables

Abbreviations

3D	Three Dimensional
2D	Two Dimensional
DCM	Different Cost Model
FNB	Flexible Naive Bayes
GPMD	Global Path with Minimum Distractions
HMM	Hidden Markov Model
IFTM	Infinite Factored Topic Model
KDE	Kernel Density Estimation
MRF	Markov Random Fields
PRM	Probabilistic Road Map
RGB-D	Red, Green, Blue and Depth
SVM	Support Vector Machine
S-SVM	Structured Support Vector Machine
SLAM	Simultaneous Localization and Mapping
SMOTE	Synthetic Minority Over Sampling Technique

Nomenclature

$f(\cdots)$	A scalar valued function
$\boldsymbol{f}(\cdots)$	A vector valued function
$P(x)$	Probability of a random variable x
$P(x\|z)$	Conditional probability of x, given evidence z
$P(x, z)$	Joint probability of x and z
$DT[\cdot]$	Distance transform
$[\cdot]^T$	Transpose of a vector or a matrix
$\|\cdot\|$	The magnitude of a vector
$\Phi(\cdot)$	Feature function
$E[\cdot]$	The expectation of a random variable
$K((\cdot),(\cdot))$	Kernel function

Specific Symbol Usage

A	Affordance-map
i_k	i^{th} Point cloud image
y_i	i^{th} Label
$\mathbf{x}_n$	Feature vector at i^{th} grid location
$\mathbf{g}_i$	i^{th} Grid location of the 3D map
C_k	k^{th} Class label
$\mathbf{w}$	The normal vector to the separating hyperplane of the SVM classifier
b	The offset of the hyperplane from the origin along the normal vector of the SVM classifier
ξ_i	i^{th} Slack variable

α_i	i^{th} Lagrange multiplier
$R_z(\theta_k)$	The rotational matrix about the z axis
H_l	Human skeleton model
$\bar{y}_k$	k^{th} Ground truth label
$\triangle(\bar{y}', \bar{y}_k)$	Loss function
μ_s	The mean of position and orientation of the robot in human skeleton co-ordinate system
Σ_s	The covariances of position and orientation of the robot in human skeleton co-ordinate system

Glossary of Terms

Affordance	All action possibilities latent in the environment, objectively measurable and independent of the individual's ability to recognize them.
Environment	A complex 3D unstructured place . Assumed to have some structural characteristics such as planar surfaces.
Features	Distinctly identifiable points in an image of a 3D environment.
Freespace	Areas in the environmental model or map that are known to be free of objects, obstacles and surfaces.
Iteration	A single step which is determined by optimisation.
Obstacle	An object within the environment which a robot can collide with.
Occlusion	Not visible from a viewpoint due to an obstruction.
Pose	The combination of position and orientation of an object in 3D space with respect to a reference coordinate frame.
Planning	The act of generating a path (and motion) course which the robot can then follow to get between two poses.
Surface	This is the face of an object in the environment.
Surface Normal	A 3D vector perpendicular to a surface.
Unstructured	Real-world environment that cannot be set up to facilitate experiment.
Viewpoint	A position in space and an orientation of a sensor that a corresponding robot pose can achieve.
Voxel	Volumetric Pixel which represents a 3D cube-like volume in Euclidean space.

Chapter 1

Introduction

1.1 Human Context

Human Robot Interaction (HRI), which is the study of interactions between humans and robots, has gained significant attention over recent years. This is due to the fact that many robotics applications require robots to work alongside humans in a safe and acceptable manner, as opposed to conventional robotics where robots operate in isolation. Therefore, it is believed that the ability of a robot to understand the human context in its surrounding environment is of paramount importance for a better human-robot interaction.

In robotics, learning human context often involves tracking humans to learn their motion patterns [3], human activity detection [4, 5] and modeling relationships between humans and their surroundings [6]. Almost all of these techniques require robots to detect and track humans for a considerable amount of time before being used to model human context. On the other hand, detecting, tracking and activity detection of humans in a cluttered environment are still largely open problems. Further, these tasks become more challenging and complicated when robots have to accomplish them while moving in a socially acceptable manner. Often these existing techniques require a considerable amount of re-engineering when they are introduced into new environments.

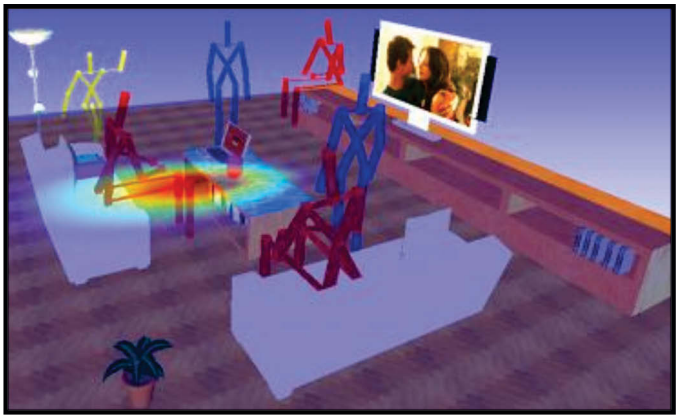


FIGURE 1.1: A 3D living room scene

1.2 Affordances

Introduced by Gibson in 1977 [7] affordance theory defines the word "affordance" as all action possibilities latent in the environment, objectively measurable and independent of the individual's ability to recognize them [1, 7, 8]. Affordance theory argues that action possibilities are motivated by how the environment is arranged. For example, chairs and sofas support the activity 'sitting' and they are physically designed to support that affordance which encourages the actor for sitting. Therefore, we believe this strong relationship between human context and environmental affordances could be used to learn human context even when humans are not observable. The rationale here is that it is possible to learn environmental affordances by only looking at geometric features of the environment and this forms the basis for the concept called affordance-map which is introduced in this thesis.

1.3 Motivation to Learn Hidden Human Context

The human context becomes 'hidden' when humans are not observed in a new environment. For example, consider the living room scene shown in Fig. 1.1. The humans have an amazing ability to look at the scene in Fig. 1.1 and infer the human context in the environment.

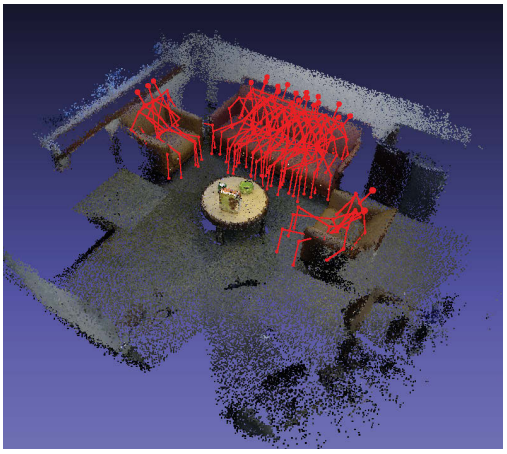
If someone asked you to place a TV on this environment, you would probably place it in front of the sofas . For this you would imagine a few sitting humans and decide the final position of the TV based on the pose of these imagined humans so most of them could view the TV screen properly. If a service robot is required to locate the TV in this room, first it could infer sitting humans on sofas using their geometric features, and then it could use the human pose information to estimate the location of the TV. Such an approach would considerably reduce the robot's search space for the TV. This observation motivates us to learn the hidden human context in 3D scenes using the concept,'affordance-map', which involves mapping possible human affordances in 3D scenes though virtual human models. This affordance-map learns the human context in a given environment without observing any real humans and bypasses challenges associated with human detection.

As robots use grid based maps for localization, path planning and obstacle detection, affordance-map could be used by a robot to improve the human robot interaction. Therefore, service robots operating in indoor environments could largely benefit from learning the hidden human context. For example, a domestic service robot could use the human context information embedded in an affordance-map to arrange objects in a human preferred manner before humans arrive from work. Or, it could use pose information of virtual human models embedded in an affordance-map to search and localize various objects. Even when humans are present in the environment, an affordance-map could be used by a robot to carry out its task more efficiently. Firstly, it could use an affordance-map to infer the possible human locations which could be used by the robot to efficiently detect humans. Secondly, it could use information from the affordance-map to plan paths that minimize interferences with humans. The affordance-map could even provide strong priors for human activity detection when humans are partially observed or the views of them are completely obstructed.

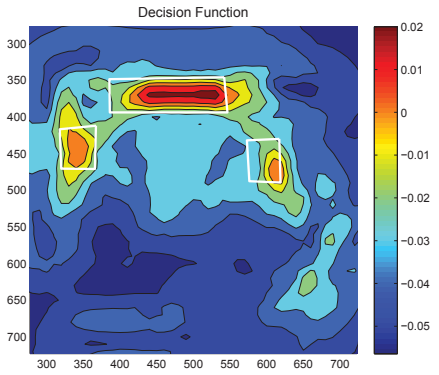
1.4 Objectives and Problem Statement

The main objective of this thesis is to build the affordance-map for a given 3D scene, which later can be used to learn the human context of the environment. This affordance-map consists of two sub-maps as shown in Fig. 1.2. First, it predicts virtual human skeleton

models in locations that support the tested affordance as shown in Fig. 1.2(a). Then it outputs a confidence value for each predicted skeleton that indicates how likely the skeleton model is being supported by the surrounding environment as shown with a contour map in Fig. 1.2(b).



(a) Virtual Skeleton Map



(b) The contour map of affordance likelihoods

FIGURE 1.2: The affordance-map of a living room

The affordance-map building process can be formulated as a supervised learning problem. Given a set of 3D pointcloud images $\{i_1, ..., i_k\} \subset I$ and their associated affordance-map A, the goal is to learn mapping $g : I \rightarrow A$ which can be later used to automatically build an affordance-map in unseen pointclouds.

As some locations of the map could have multiple affordances-labels, mapping $g : I \rightarrow A$ becomes a multi label classification problem. As affordance types are not mutually exclusive, one way to simplify this problem is to divide each 3D image space into four dimensional grid poses ((x, y, z) location of the skeleton model and θ orientation) and build an independent binary classifier for each affordance type. Each binary classifier predicts a binary label vector $\bar{\mathbf{y}}_k = (y_1, .y_i.., y_n)$ where $y_i \in \{+1, -1\}$ for the feature vector $\bar{\mathbf{x}}_k = (x_1, ..., x_n)$, $x_i \in \Re^n$ calculated for each grid location of the 3D scene. The label y_i becomes $+1$ if that location supports the tested affordance and -1 if not.

1.5 Contributions

This thesis presents the following four contributions to the field of indoor mobile robots.

- The affordance-map concept is proposed as a novel method for learning hidden human context in indoor environments using 3D pointcloud data.

  This task is formulated as a multi-label binary classification problem which assigns a binary label for each grid location of the map using 3D geometric features obtained through virtual human models. The learned affordance-map consists of virtual skeletons and affordance likelihood values mapped for each grid location of the 3D environment.

- Critically evaluation of existing solutions to the class imbalanced problem associated with the Support Vector Machine (SVM) based learners when used for mapping affordances.

  As the number of negative examples in the dataset outnumbers positive examples by a large margin, the dataset of the ground truth 3D scenes is considered to be highly imbalanced. Consequently, initial tests on affordance mapping showed that the SVM learners tend to provide sub optimum results when trained with highly imbalanced data. As the first solution to the problem, a number of widely used SVM based algorithms are experimentally analysed both quantitatively and qualitatively for learning affordances. However, the experimental results proved that the results could be further improved. Therefore, as final practical solutions for affordance learning and class imbalanced problems, this thesis formulates the affordance mapping process as a structured output learning problem that can be optimized on F1-score. Experimental results are presented to show the effectiveness of the proposed structured output learning over existing methods for mapping affordances.

- The application of human context information embedded in the affordance-map for active object search.

  The active object search involves executing a series of sensing actions in order to bring the object to the field of view of the sensor. Rather than using common

object-object relationships to solve this problem, the proposed affordance-map based approach uses human skeleton information embedded in the affordance map to model human-object relationships to predict probable locations that give the best views of the object. Then the derived robot-human-object relationships are used to build an effective object search strategy. The effectiveness of the proposed approach is experimentally verified using a mobile robot that actively searches for computer monitors in an office environment.

- The application of human context information embedded in the affordance-map for context-aware path planning.

  In this application, context-aware path planning is defined as the path that minimizes the interferences caused to the people working in an office cell. The challenge in this task is to detect and avoid human working spaces which are considered to be non-trivial for a moving robot to detect. Therefore, the major contribution of this approach is using an affordance-map to assign social cost values implicitly without directly identifying workspaces or real humans. To achieve this, human skeleton informations and their class likelihood values are used to assign social cost value to each grid location of the map. These social cost values represent the degree of interference caused by a moving robot to humans. These cost maps are used to develop a context-aware global path planning strategy by using the well known A* algorithm.

1.6 Thesis Outline

This thesis consists of eight chapters.

Chapter 2 presents background materials required for understanding the affordance mapping concept. It first discusses previous research work that is closely related to the affordance mapping. Then it briefly introduces various 3D map building algorithms. Finally, it introduces various classifiers and defines various performances measures that are widely used for measuring binary classification results.

Chapter 3 formulates the affordance mapping as a binary classification problem and applies the widely used binary classifiers namely Support Vector Machine (SVM) and Naive Bayes classifier to build the affordance-map. It also introduces the skeleton features used for mapping affordances and the experimental setup used for analyzing the performances of the two classifiers. Finally, it presents experimental results of the two classifiers followed by a discussion of limitations of the SVM and Naive Bayes classifier.

Chapter 4 address the class imbalance problem associated with SVM learners. The behaviors of a number of SVM based algorithms that are designed to handle class imbalance problem are experimentally tested for learning affordances. The results of these experiments are also presented in this chapter by recording F1-score for various class imbalances. Finally, the limitations and applicability of these algorithms in learning the affordance-map are also discussed.

Chapter 5 formulates the problem of mapping affordances in 3D scenes as a Structured SVM (S-SVM) learning problem that can be optimized on F1-score. First, the theories behind this formulation are introduced. Then the performances of the S-SVM algorithm for affordance learning are analyzed both quantitatively and qualitatively. Finally, the advantages and limitations of the S-SVM algorithm for affordance learning are discussed by comparing the results with other related algorithms.

Chapter 6 introduces an application of the affordance-map. It shows how the human context embedded in the affordance-map can be used for active object search. In this method, the human skeleton map which is derived from the affordance-map is used to infer locations that give the best views of the searched object. To verify the proposed algorithm, results of a mobile robotic active object search experiment, which is carried out in a large office environment are also presented in this chapter.

Chapter 7 explains another possible application of the affordance-map. This chapter presents a method that uses a human skeleton map to develop a context-aware robotic path planning strategy. In this path planning method, a social cost map derived from the skeleton map is added to the occupancy cost map to plan the shortest path with minimal obstructions to humans. This method is tested in an office environment using the popular A^* algorithm.

Chapter 8 concludes the thesis with a summary of the key findings and the contributions. It also points to several interesting future research areas that could benefit from this thesis.

1.7 Publications

The work in this thesis has been previously presented in the following publications.

1. Lasitha Piyathilaka and Sarath Kodagoda. Affordance-map: A map for context-aware path planning. In *ACRA, Australasian Conference on Robotics and Automation 2014*. ARAA, 2014

2. Lasitha Piyathilaka and Sarath Kodagoda. Active visual object search using affordance-map in real world : A human-centric approach. In *ICARCV, The 13th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2014

3. Lasitha Piyathilaka and Sarath Kodagoda. Learning hidden human conetext in 3d office scenes by mapping affordances through virtual humans. *Unmanned Systems*, 2015

4. Lasitha Piyathilaka and Sarath Kodagoda. Gaussian mixture based hmm for human daily activity recognition using 3d skeleton features. In *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*, pages 567–572. IEEE, 2013

5. Lasitha Piyathilaka and Sarath Kodagoda. Human activity recognition for domestic robots. In *Field and Service Robotics Conference*, pages 567–572. Springer, 2013

Chapter 2

Related Work and Background

2.1 Introduction

This chapter presents the background material required for the affordance mapping process described in this thesis. It consists of two parts. The first part presents the closely related work relevant for human context understanding and learning affordances. The second part briefly introduces various classification methods and their frequently used performance measures.

2.2 Related Work

2.2.1 Learning Human Context

Automatic recognition of human context in an environment is a well studied research area with a large body of previous work. Most of this previous work relies on the fact that humans need to be detected and their activities need to be identified in order to successfully learn the human context in an environment. Therefore, most researchers have focused their studies on solving problems associated with human tracking and human activity detection.

One such popular approach is learning human context by tracking human motion patterns using various sensors installed in smart environments. These sensor networks include motion sensors, door sensors to detect human movements and sensors installed in equipment and cupboards [11] [12]. The major challenge with such systems is the requirement of large number of sensors and their adaptability to new environments. In addition, due to the low resolution and discrete nature of the data gathering the low level human context information could be missed.

Various researchers have also focused on the use of wearable sensors like gyros as a way of continuous human and activity detection [13]. However, the use of wearable sensors are too intrusive discouraging the applicability. Human context detection based on fixed video cameras have been heavily exploited in computer vision research. This includes human detection, tracking and human activity detection. However, in general, clutter contributes to low detection accuracies [14]. In addition, video data can be obscured due to lighting conditions, type of costumes and background colors. Further, video recordings raise privacy issues in many scenarios.

Some researchers have also looked into the use of robotic systems for human activity recognition so the human context can be learned. The main advantage is all sensors can be mounted on the robot so it can do active sensing. In [15], audio-based human activity recognition using a non-markovian ensemble voting technique is presented. Although some of the human activities produce characteristic sounds from which the robot can infer activities, only some of the activities generate distinguishable sounds. Therefore, such a system may only be used as a complement to the existing sensory systems. Recent advancements in video game consoles have invented low cost RGB-D cameras like Microsoft Kinect^{TM}. These cameras provide a wealth of depth information that a normal video camera fails to provide. In addition, RGB-D data from these sensors can be used to generate a skeleton model of a human with semantic matching of body parts. These skeleton features have been used by several researchers to detect humans and their activities [4, 5]. The other main advantage of 3D sensing is location information of humans can be easily calculated.

However, learning human context by watching real humans could become problematic in most scenarios. First, robots will have to observe humans for a considerable period of time

before learning the human context in a new environment. On the other hand, many robots are often required to operate in environments where real humans would not be observed at all, but robots still need the human context information to carry out various tasks. A good example of this is a situation where a service robot is required to arrange small items in a house before humans arrive home from work. In this scenario, the robot needs human context information to arrange objects in a human preferred manner and would have to gather human context information without observing real humans.

Even though humans are visible, learning human context from them could become problematic in most scenarios. For example, learning human context could become challenging when the human subject moves away from the sensory range or when the sensor inputs become obscured. One way to fix this problem is to mount sensors on a mobile robot and guide the robot towards humans. However, when the robot starts to move it could create several other social issues. For example, the robot might interfere with the activity that the human is performing or it could collide with the human if the possible motion path of the human is blocked. The other major issue with observing real humans is privacy of humans. It is no doubt that the presence of a robot inherently affects a human's sense of privacy [16]. Privacy relates to the right of an individual to decide what information about himself or herself can be shared with others. Research in the field of human-computer interaction has shown that the user should have sufficient level of awareness and control of shared information in order to achieve widespread acceptance of new technologies [17]. Therefore, when sensors like cameras are used to learn human context privacy issues could become paramount.

2.2.2 Human Context and Object Affordances

Because of problems associated with human detection and tracking researchers have tried various other techniques that could be used for learning human context. One such popular method, the concept of affordance has become a focus of attention within the cognitive vision and robotics community lately. Psychologist James J. Gibson originally introduced affordance in his 1977 article 'The Theory of Affordances' [7] and explored it more fully in his book 'The Ecological Approach to Visual Perception' [18] in 1979. The Affordance

theory states that the world is perceived not only in terms of object shapes and spatial relationships but also in terms of object possibilities for action (afordances). In other terms, perception drives actions. Therefore, it could be argued that human action possibilities and thus the human context could be inferred by looking at the environmental properties. Further in his book "The Ecological Approach to Visual Perception" he explains 'sittable' affordance as follows.

> " Different layouts afford different behaviors for different animals, and different mechanical encounters. The human species in some cultures has the habit of sitting as distinguished from kneeling or squatting. If a surface of support with the four properties (horizontal, flat, extended and rigid) is also knee high above the ground, it affords sitting on. We call it a seat in general, or a stool, bench, chair, and so on, in particular. It may be natural like a ledge or artificial like a couch. It may have various shapes, as long as its functional layout is that of a seat. The color and texture of the surface are irrelevant. Knee high for a child is not the same as knee-high for an adult, so the affordance is relative to the size of the individual. But if a surface is horizontal, flat, extended, rigid, and knee high relative to a perceiver, it can in fact be sat upon. If it can be discriminated as having just these properties, it should look sit-on-able. If it does, the affordance is perceived visually. If the surface properties are seen relative to the body surfaces, the self, they constitute a seat and have meaning"
> **- Gibson 1977, *The Ecological Approach to Visual Perception.***

Motivated by these arguments in affordance theory, a few researchers have recently attempted to learn action possibilities hidden in the environment by only looking at geometric properties. These approaches bypass the need for human detection and tracking.

In [1], researchers use virtual human models to recognize objects that have 'sittable' affordance without using common approaches such as 3D features for object recognition. They use a human model with a sitting pose to calculate distance features to learn an affordance model, which is modelled as a probability density. These affordance probabilities are calculated for each 3D grid location of the room and the locations with highest

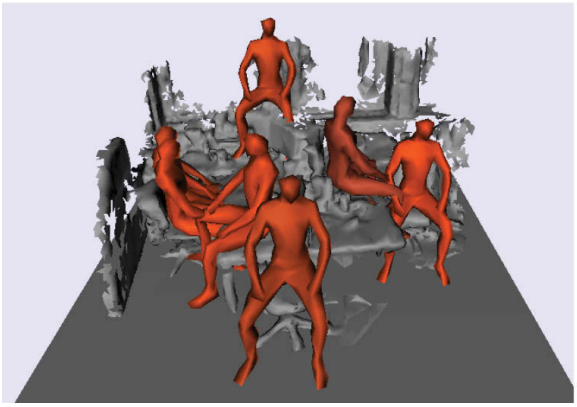


FIGURE 2.1: Predictions results for 'sittable' locations in an office space by the algorithm discussed in [1]

.

probabilities are classified as 'sittable'. The affordances model is trained on a synthetic dataset with different 'sittable' furniture types. Although they achieved good recognition accuracies with synthetic datasets, they failed to record good results when tested on a real 3D environment as shown in Fig. 2.1.

Although the results of [1] are encouraging, it has the following limitations when applied for affordance mapping. First, the learning process is not discriminative. Therefore, it cannot distinguish the difference between 'sittable' and 'non-sittable' locations correctly. Because of this, it would classify the locations with high prediction probability as 'sittable' in a room where any 'sittable' locations are not observed. On the other hand, in this method parameters of the training model are trained using furniture downloaded from a synthetic dataset and their arrangements in real rooms are not considered. However in real world scenarios, the distance features easily could be affected by nearby objects, walls and the floor. Perhaps this could be the major reason for this algorithm recording high false positive detections in real 3D scenes.

Recently, a few researchers introduced virtual humans models to learn human object relationships (Affordances) in 3D scenes [19]. First, object human relationships are learned from labeled objects. Then learned models are used to arrange objects in a human preferred manner in a synthetic environment. However, they assumed that the objects have been detected first before modelling affordances and therefore cannot be used in a new

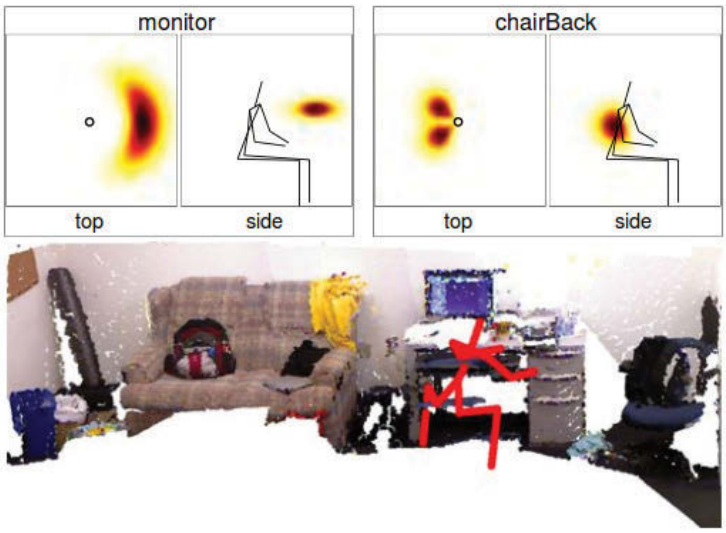


FIGURE 2.2: A Human model inferred by the algorithm discussed in [2]. It is highly influenced by the features of the monitor and its location is not supported by a chair.

3D room without object labels. As an extension for this research, in [2] researchers propose to use Infinite Factored Topic Model (IFTM) to model object human relationships and use them to improve the object detection accuracy. They used hallucinated humans to derive object human relationship features. However, object locations and locations of human models are not initially known but learned jointly from environmental features during the training process. However, the main intention of this method is to improve 3D object detection accuracy rather than correctly learning the correct location of the human model. Therefore, locations of the human models inferred by this method might not be optimum and heavily dependent on the presence of object types that were used for training affordance models.

The affordance mapping process presented in this thesis differs from these existing algorithms in three ways. Firstly, the proposed affordance learning process is formulated as a multi-label binary classification problem. Therefore, unlike existing methods, it can correctly label each grid locations of the 3D scene either with a positive label or negative label with a confidence value. Secondly, object human relationships are modelled implicitly and are included as features in the training phase. Therefore the model is not required to

detect objects or their labels to learn the affordance-map. Thirdly, this thesis proposes a framework to build an affordance-map in a large room efficiently with multiple types of affordances.

2.3 Types of Classifiers

This section briefly defines basic types of classifiers often used in machine learning. Classifiers can be categorized in different ways but in this thesis the following types of classifiers are frequently used. Hence a brief introduction is given for each of these classifier types.

2.3.1 Binary Classifier

The binary classifier is a classifier that divides a given set of examples into two classes using a classification rule. In most practical binary classification problems, the two classes are not symmetric. Some binary classification algorithms are effected by this class imbalance and therefore class imbalances need be taken into consideration when designing the classification rule. Logistic regression, Support vector machine, Relevance vector machine, Perceptron, Naive Bayes classifier, k-nearest neighbors algorithm, Artificial neural network and Decision tree learning are the most common types of binary classifiers.

2.3.2 Multi Class Classification

In multi class classification, a set of examples is divided among multiple classes. One example can not be included in two or more classes and each example has only one class label. Some of the multi class classification rules like Bayesian Networks [20] and Structured Output SVM [21] support multi class classification in their algorithms. However all binary classifiers can be converted to multi-class classifiers by using methods such as One-vs-Rest and One-vs-One [22].

2.3.3 Multi Label Classification

In multi label classifications, multiple labels are assigned to each instance of the classification problem. Therefore each instance could have multiple class labels. The most common way to train a multi label classification problem is by independently training a binary classifier for each label. Then these classifiers are combined and instances with positive labels from each classifier are assigned with multiple labels.

2.4 Evaluation Measures

Evaluation measures play a major role in both assessing the classification performance and training the classification model. The accuracy is the most commonly used measure for these purposes. However, with imbalanced data, accuracy is not a proper measure since the minority class has very little impact on the accuracy as compared to that of the majority class [23]. For binary class problems, the information retrieval community has defined following measures that can be obtained from the confusion matrix:

	Condition Positive	Condition negative
Prediction Positive	TP	FP
Prediction Negative	FN	TN

Table 2.1: Contingency table for Binary Classification

Specificity or True negative rate (TNR)

$$SPC = \frac{TN}{TN + FP} \tag{2.1}$$

Negative predictive value (NPV)

$$NPV = \frac{TN}{TN + FN} \tag{2.2}$$

Fall-out or false positive rate (FPR)

$$FPR = \frac{FP}{FP + TN} \tag{2.3}$$

False negative rate (FNR)

$$FNR = \frac{FN}{FN + TP} \tag{2.4}$$

Accuracy (ACC)

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

If only the performance of the positive class is considered, as in the case of mapping affordances, two measures are important: True Positive Rate (TPR) and Positive Predictive Value (PPV). True Positive Rate is defined as recall (R) in information retrieval which is the percentage of retrieved instances that are relevant:

$$PPV = R = \frac{TP}{TP + FN} \tag{2.6}$$

Positive Predictive Value is defined as precision (P) which is the percentage that a retrieved instant is relevant.

$$TPR = P = \frac{TP}{TP + FP} \tag{2.7}$$

F1-scores have been commonly used in tasks in which it is important to retrieve elements belonging to a particular class correctly without including too many elements of other. Therefore the F1-score is widely used to evaluate binary classifiers. It is particularly preferable over other performance measures for highly unbalanced classes. The F1-score is the harmonic mean of the precision and recall and given by 2.8. Because of the F1- score calculates the harmonic mean it tends to be closer to the smaller of the two. Therefore, a high F1-measure value means that both recall and precision are high.

$$F1Score = \frac{2 * P * R}{P + R} \tag{2.8}$$

Chapter 3

Basic Classifiers for Mapping Affordances

3.1 Introduction

Binary classification forms the basis for the concept of affordance-map. Since affordance labels are not mutually exclusive, affordance mapping can be categorized as a multi-label classification problem. Given n number of feature vectors $\bar{\mathbf{x}}_k = (\mathbf{x}_1, ..., \mathbf{x}_n)$ for each grid location of the k^{th} 3D image, this classifier tries to assign n number of affordance label vectors, $\bar{\mathbf{a}}_k = (\mathbf{a}_1, ..., \mathbf{a}_n)$ to n grid locations of the image.

One way to simplify this problem is to consider each affordance separately and learn an independent set of classifiers. Then it becomes a binary classification problem and can be defined as a function h that maps a single feature vector $\mathbf{x}$ at each grid location $\mathbf{g}_i = \{x_i, y_i, z_i, \theta_i\}$ of the image to a single label $y \in \{-1, +1\}$.

$$h : \mathcal{X} \rightarrow \mathcal{Y} \tag{3.1}$$

There are two types of basic binary classifiers: generative classifiers and discriminative classifiers. The generative model learns the joint probability $P(y, x)$ for a given set of x input data and their labels y , while the discriminative model learns the conditional

probability $P(y|x)$. Naive Bayes classifier is a popular generative classifier and Support Vector Machine (SVM) is a widely used discriminative classifier. Both these classifiers are capable of predicting class labels with prediction scores and therefore ideally suited for building affordance-maps. In the case of SVM classifier, the prediction score denotes the sign distance from the observation to the decision boundary, while the Naive Bayes classifier directly outputs class posterior probability.

The purpose of this chapter is to analyse the applicability of SVM and Naive Bayes classifiers for mapping affordances. It first introduces the underlying theories behind these two classifiers. Then features based on virtual human models are introduced. These features model the relationships between the humans and the environment and defines the underlying human context of the environment. These features are later used to learn the parameters of the SVM and the Naive Bayes classifiers. Then experiments are carried out on a dataset consisting of a number of 3D images captured in office and domestic environments. Finally, experimental results are evaluated qualitatively and quantitatively to select the best binary classifier for building the affordance-map.

3.2 Naive Bayes classifier

The naive Bayes classifier is a simple probabilistic classifier based on Bayes theorem with strong independence assumptions. It is a simple technique for constructing classifiers that assigns class labels to problem instances, represented as vectors of feature values. The naive Bayes classifiers assumes that each particular feature in a $\mathbf{x}$ feature vector is independent of any other feature, given the class label, C_k. In other terms, it assumes that each feature contributes independently to the probability of the class label, irrespective of any possible correlations between individual features. The value of this assumption is that it dramatically simplifies the representation of the conditional probability $P(\mathbf{x}|C_k)$, and the problem of estimating it from the training data. In fact, accurately estimating $P(\mathbf{x}|C_k)$ typically requires many training examples. For example, for a binary classifier with n number binary features would require the calculation of $2(n^2 - 1)$ number of probability values to estimate $P(\mathbf{x}|C_k)$. On the other hand, feature independence assumption in the

naive Bayes approach reduces the required number of parameters to $2(n-1)$, which can be calculated with much lesser number of training examples.

Despite their oversimplified independent assumptions and naive design, naive Bayes classifiers have performed well in a number of real-world application domains [24, 25]. Furthermore, naive Bayes classifier can deal with a large number of variables and large data sets, and it is capable of handling both discrete and continuous attribute variables.

3.2.1 Probabilistic Approach

In fact, naive Bayes is a simplified conditional probability model. Given a problem instance to be classified, represented by a feature vector $\mathbf{x} = (x_1, \ldots, x_n)$, it assigns probabilities

$$P(C_k|x_1, \ldots, x_n) \tag{3.2}$$

for each k class label.

However, if the number of features n is large or the features are multidimensional then modeling $p(C_k|\mathbf{x})$ using probability tables becomes infeasible. Therefore, Bayes' theorem can be used to reformulate the above equations to make them more tractable as follows.

$$P(C_k|\mathbf{x}) = \frac{P(C_k)\ P(\mathbf{x}|C_k)}{p(\mathbf{x})}. \tag{3.3}$$

The above equation can also be expressed as follows using Bayes' terminology.

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \tag{3.4}$$

In classification, there is little interest on the denominator because it does not depend on C and is effectively a constant and often termed the normalizing constant.

The numerator is equivalent to the joint probability model:

$$p(C_k, x_1, \ldots, x_n) \tag{3.5}$$

which can be rewritten using the Chain rule of probability as shown in (3.6).

$$\begin{aligned} P(C_k, x_1, \ldots, x_n) &= P(C_k)\ P(x_1, \ldots, x_n | C_k) \\ &= P(C_k)\ P(x_1|C_k)\ P(x_2, \ldots, x_n | C_k, x_1) \\ &= P(C_k)\ P(x_1|C_k)\ P(x_2|C_k, x_1)\ p(x_3, \ldots, x_n | C_k, x_1, x_2) \\ &= P(C_k)\ P(x_1|C_k)\ P(x_2|C_k, x_1)\ \ldots p(x_n | C_k, x_1, x_2, x_3, \ldots, x_{n-1}) \end{aligned} \tag{3.6}$$

Now the "naive" conditional independence assumptions can be used to extensively simplify the equation (3.6) by assuming that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the class label C. This means that

$$\begin{aligned} P(x_i | C_k, x_j) &= P(x_i | C_k) \\ P(x_i | C_k, x_j, x_k) &= P(x_i | C_k) \\ P(x_i | C_k, x_j, x_k, x_l) &= P(x_i | C_k) \end{aligned} \tag{3.7}$$

and so on, for $i \neq j, k, l$. Thus, the joint model can be expressed as

$$\begin{aligned} P(C_k | x_1, \ldots, x_n) &\propto P(C_k, x_1, \ldots, x_n) \\ &\propto P(C_k)\ p(x_1|C_k)\ P(x_2|C_k)\ P(x_3|C_k)\ \cdots \\ &\propto P(C_k) \prod_{i=1}^{n} p(x_i|C_k)\,. \end{aligned} \tag{3.8}$$

Finally, the above independence assumptions convert the conditional distribution over the class variable C into the following simplified form

$$P(C_k|x_1,\ldots,x_n) = \frac{1}{Z}P(C_k)\prod_{i=1}^{n}P(x_i|Ck) \tag{3.9}$$

where the evidence $Z = P(\mathbf{x})$ is a scaling factor that depends only on $x_1,\ldots,x_n$ and is a constant if the values of the feature variables are known.

3.2.1.1 Classification Rule

The naive Bayes classifier combines the probability model derived in (3.9) with the decision rule. The most common rule is to pick the hypothesis that is most probable; this is known as the 'maximum a posteriori' or MAP decision rule. The corresponding Bayes classifier is the function that assigns a class label $\hat{y} = C_k$ for k classes as follows.

$$\hat{y} = \underset{C_j\in\{C_1,\ldots,C_k\}}{\operatorname{argmax}} P(C_k)\prod_{i=1}^{n}P(x_i|C_j) \tag{3.10}$$

3.2.1.2 Parameter Estimation

The class prior $P(C_k)$ can be calculated by assuming equal probabilities for each class (i.e., priors = 1 / (number of classes)) which is often called the 'Uninformative Prior'. In such cases, the decision rule given in (3.10) is unaffected by each class' prior probability. The other most common approach is estimating class' prior probability from the training set (i.e., (prior for a given class) = (number of samples in the class) / (total number of samples)). The feature distribution $P(x_i|C_k)$ can be assumed from a common distribution type or a nonparametric model can be generated from the training data. These assumptions on distributions of features are called the 'event model' of the Naive Bayes classifier.

3.2.2 Gaussian naive Bayes

When features are continuous, often it is assumed that they are distributed according to the Normal distribution. This is because parameters of a Normal distribution can

be calculated easily from the training dataset. For example, suppose the training data contain a continuous attribute, x_i. Then to learn the Gaussian Naive Bayes model, first the data is segmented by the class, and the mean and variance of x_i in each class are calculated. Let μ_c be the mean of the values in x_i associated with each class C_k, and let σ_c^2 be the variance of the values in x_i associated with class c. Then, the likelihood $P(x_i = v|C_k = c)$ of feature value v , can be computed by assigning v into the equation for a Normal distribution parameterized by μ_c and σ_c^2. That is,

$$P(x_i = v|C_k = c) = \frac{1}{\sqrt{2\pi\sigma_C^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}} \tag{3.11}$$

3.2.3 Flexible Naive Bayes

Although using a single Gaussian for density estimation is simple, most of the real-world data distributions are not truly Gaussian. Therefore, researchers have also explored a number of nonparametric density estimation methods. This has led to the introduction of the Flexible Bayes learning algorithm which uses a nonparametric density estimation technique but is similar to the Gaussian naive Bayes in all other respects. One such popular nonparametric density estimation is Kernel Density Estimation (KDE).

In Gaussian Naive Bayes, the density of each continuous attribute is represented as $p(x_i|C_k) = g(x, \mu_c, \sigma)$. In Kernel Density Estimation, the estimated density is averaged over a large set of kernels as,

$$p(x_i = v|C_k = c) = \frac{1}{nh} \sum_{i=1}^{n} K\Big(\frac{v - x_i^t}{h}\Big) \tag{3.12}$$

where K is the kernel, a non-negative function that integrates to one with mean zero, $h > 0$ is a smoothing parameter called the 'bandwidth' and x_i^t is t^{th} training example of the i^{th} feature. A range of kernel functions are commonly used: uniform, triangular, biweight, triweight, Epanechnikov and normals. The normal kernel is used frequently due to its simplicity in mathematical properties. In the case of normal kernel $K = g(x, 0, 1)$ and $h = \sigma$.

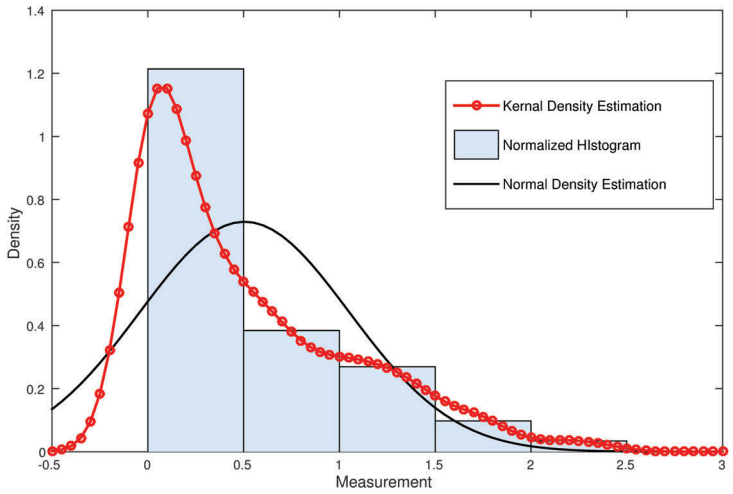


FIGURE 3.1: The effect of using a single Gaussian versus kernel method to estimate the density of a continuous variable

The sufficient statistics for a normal distribution are mean μ and variance σ^2 . Hence Gaussian Naive Bayes only has to store two parameters per each attribute but Flexible Bayes must store every continuous attribute value it sees during training. Therefore sufficient statistics for Flexible Bayes are the training instances themselves. On the other hand, to classify unseen test data Naive Bayes only has to evaluate g once, but Flexible Bayes must perform n evaluations, one per observed value of x_k in class C_k. This increases computation time substantially.

Despite these complexities, Flexible Bayes has performed well in domains that violate the normality assumption [25]. This can be understood by looking at a density distribution which is not a normal distribution. For example, consider the distance features for the joint 'Left Hip' for the affordance class 'sittable' shown in Fig. 3.1. It is clear that these features are not normally distributed and the normal density estimation does not represent it. However, Kernel density estimation better represents the density histogram of the test data and is smoother compared to the discreteness of the histogram. Therefore, KDE is a better representation of the test data than its normal density estimation counterpart.

The only free parameter that needs to be calculated in KDE is its bandwidth h, which exhibits a strong influence on the resulting estimate. The statistical literature presents a number of rule of thumb heuristics for calculating bandwidth but makes implicit and

explicit assumption of the density distribution that will be true in some distributions and not in others [26, 27].

3.2.4 Support Vector Machine (SVM)

This section presents the fundamental theories behind the SVM classification.

Given a classification problem represented by a dataset $\{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)\}$, where $\mathbf{x_i} \in \mathbb{R}^p$ represents a p dimensional data point and $y_i \in \{1, -1\}$ represents the label of the class of that data point, for $i = 1, ..., n$, the objective of the SVM learning algorithm is to find the optimal separating hyperplane which effectively separates data points into two classes.

The data points $\mathbf{x}$ are first transformed into a higher dimensional feature space by a non-linear mapping function Φ to allow better separation of the classes. This separating hyperplane residing in this transformed higher dimensional feature space is represented by

$$\mathbf{w} \cdot \Phi(\mathbf{x}) + b = 0 \tag{3.13}$$

where $(\cdot)$ denotes the dot product and $\mathbf{w}$ the normal vector to the hyperplane. The parameter $\frac{b}{\mathbf{w}}$ determines the offset of the hyperplane from the origin along the normal vector.

When the training data are completely linearly separable, the separating hyperplane with the maximum margin can be found by solving the following maximal margin optimization problem:

$$\begin{aligned} &\arg\min_{(\mathbf{w},b)} \{\frac{1}{2}\|\mathbf{w}\|^2\} \\ &y_i(\mathbf{w} \cdot \Phi(\mathbf{x_i}) - b) \geq 1 \\ &.s.t \forall i = 1, \ldots, n \end{aligned} \tag{3.14}$$

However, in most real-world problems, the datasets are not completely linearly separable even in higher dimensional feature spaces. If a hyperplane that can split the positive and negative classes does not exist, then the modified version of the original SVM called 'Soft Margin SVM' can be used. This method chooses the hyperplane that splits the examples as cleanly as possible, while still maximizing the distance to the nearest cleanly split examples by introducing non-negative slack variables, ξ_i, which measure the degree of misclassification of the data $\mathbf{x}_i$ [28].

Then the soft margin optimization problem can be reformulated as follows:

$$\begin{aligned} &\arg\min_{(\mathbf{w},b)} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ &y_i(\mathbf{w}\cdot\Phi(\mathbf{x_i}) - b) \geq 1 - \xi_i \\ &\qquad s.t \quad \forall i = 1,\ldots,n \end{aligned} \tag{3.15}$$

For misclassified examples, the slack variables become $\xi_i > 0$ and therefore the penalty term $\sum_{i=1}^{n}\xi_i$ represents the amount of total misclassification (training errors) of the model. As a result, the new objective function given in (3.15) has two objectives; firstly to maximize the margin and secondly to minimize the number of misclassifications or the penalty term. The parameter C in (3.15) controls the trade-off between these two objectives.

The optimization problem given in (3.15) is a quadratic optimization problem and can be easily solved by forming it as a Lagrangian optimization problem with the following dual form.

$$\begin{aligned} &\max_{\alpha_i}\{\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)\} \\ s.t. \quad &\sum_{i=1}^{n} y_i\alpha_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1,\ldots,n \end{aligned} \tag{3.16}$$

The Lagrange multipliers α_i should satisfy the following Karush-Kuhn-Tucker (KKT) conditions:

$$\begin{aligned} \alpha_i(y_i(\mathbf{w}\cdot\mathbf{x_i}-b)-1+\xi_i)=0, \quad i=1,\ldots,n \\ (C-\alpha_i)\xi_i=0, \quad i=1,\ldots,n \end{aligned} \tag{3.17}$$

The original SVM optimal hyperplane algorithm was a linear classifier. However, kernel trick can be utilized to convert original SVM classifier to a non-linear classifier. The resulting algorithm is similar, except that every dot product is replaced by a non-linear kernel function which allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space.

By applying a kernel function, such that $K(\mathbf{x}_i,\mathbf{x}_j)=\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)$ the dual otimization problem given in (3.16) can be converted to the following form.

$$\begin{aligned} &\max_{\alpha_i}\{\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)\} \\ s.t. \quad &\sum_{i=1}^{n} y_i\alpha_i=0, \quad 0\le\alpha_i\le C, \quad i=1,\ldots,n \end{aligned} \tag{3.18}$$

The above equation can be solved to find α_i and the value for $\mathbf{w}$ can be obtained as follows.

$$\mathbf{w}=\sum_{i}^{n}\alpha_i y_i\Phi(\mathbf{x}_i) \tag{3.19}$$

The data points with non-zero α_i are referred to as support vectors. Finally, the svm decision rule can be obtained as follows.

$$f(x)=sign(\sum_{i}^{n}\alpha_i y_i K(\mathbf{x}_i,\mathbf{x}_j)+b) \tag{3.20}$$

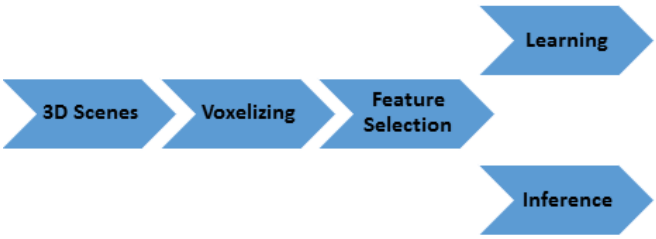


FIGURE 3.2: Steps in learning and inference stages

3.2.5 Learning and Inference Process

Both SVM classifier and Naive Bayes classifier employ supervised learning techniques which involve learning and inference stages. In the learning process, parameters of the classification model are learned from a labeled dataset. Then learned parameters are used to predict new affordance labels in new 3D scenes.

Fig. 3.2 summarizes the learning and inference process. Both learning and inference stages first require to voxelise the 3D scene into grid locations as explained in section 3.2.6. Then 3D features are calculated for each and every grid location as explained in section 3.2.7. These features and class labels from the dataset are later used to learn parameters of the classification model. This process is explained in section 3.3.4.

In inference stage, skeleton models are moved across every grid location with all possible orientations while calculating features for each and every grid location. These features and learned parameters of the classification model are then used to predict labels for the new 3D scene.

3.2.6 Voxelising 3D Scenes

The affordance map is a concatenation of classifier's outputs for each (x, y, z, θ) grid location of the 3D scene. Therefore, the first step of the affordance mapping involves voxelising the input 3D images into grid locations. This voxelising is done by dividing the input image into 10 cm x 10 cm x 10 cm grids. The rotation θ of each skeleton model is evaluated at 0.1 rad resolution at each grid level. In order to limit the search space, the grid search

along the z axis can be restricted as the human skeletons are always located close to the ground plane. Therefore, the height to the Torso from the ground plane can be estimated and the grid search along the z axis can be restricted to one grid level above and below the torso position. Even with these simplifications, the search space for a 10m x 10m room could extend up to 1,890,000 (100 x 100 x 3 x6 3) unique grid locations

3.2.7 Feature selection

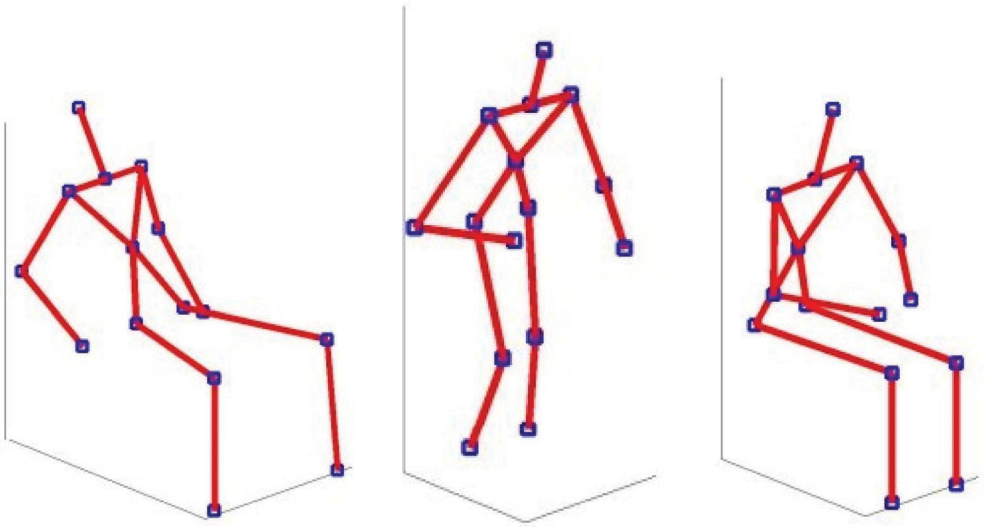
The proposed affordance-map predicts virtual skeleton models with likelihood values for each (x_i, y_i, θ_i) grid location of the 3D image. It is based on a binary classifier that predicts positive labels to the locations that support the tested affordance and negative labels to the locations that do not support the tested affordance. Therefore, the classifier's performance largely depends on the types of features used. These features should be highly informative such that the classifier would be able to predict class labels correctly.

In this thesis, a new set of features based on virtual human skeleton models is proposed for mapping affordances. These features directly model the relationship between the humans and the environment. Following sections describe different types of skeleton models and their associated features used by binary classifiers to build the affordance-map.

3.2.7.1 Virtual Human Skeleton Model

Instead of observing real humans in the environment, the proposed affordance mapping process uses virtual humans to model interaction between the environment and the humans. Although many human poses could be observed in a given environment, very few of them directly influence the context of the environment. For example, the most frequently observed human pose in an office environment is sitting and working at an office desk. Therefore, if the locations within the office room that support this affordance can be identified then the human context of the environment can be inferred easily.

For the purpose of mapping affordances, human skeleton models are obtained from a human activity detection dataset [29]. These human skeleton models are captured using a depth camera from real humans while they perform different activities. The K-mean



(a) Sitting-Relaxing (b) Standing-Working (c) Sitting-Working

FIGURE 3.3: Types of affordances and their associated skeleton models

clustering classifies all skeleton models into three clusters and the most frequently seen pose in the each cluster is shown in the Fig .3.3. Each of these skeleton models are associated with an affordance type that closely represents the type of activity that each skeleton model exhibits.

Each human model is a skeleton with 15 joint body positions in 3D. In order to obtain feature vectors, these virtual skeleton models need to be transformed to each (x, y, z, θ) pose of the map. This can be done by moving the 3D points of the human skeleton, H_l across given environment using the rigid body transformations of translation and rotation. Then each human skeleton model can be mapped to the coordinate system of the environment using (3.21), where $\mathbf{g}_k = (x_k, y_k, z_k, \theta_k)$ is the position and orientation of the skeleton's torso in the world coordinate system and $R_z(\theta_k)$ is the rotational matrix about the z axis (vertically up). It is to be noted that only rotation about the z axis is considered here.

$$H_w(\mathbf{g}_k) = [x_k, y_k, z_k]^T + R_z(\theta_k).H_l \tag{3.21}$$

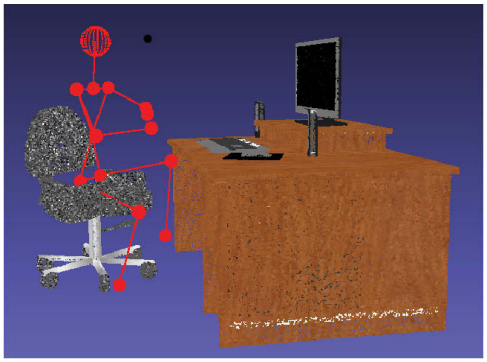


FIGURE 3.4: Virtual human models sitting near a computer

3.2.7.2 Distance and Collision Features

The features model the relationship between the human skeletons and the environment. This relationship is modeled through two geometric features; distance features and collision features. Selection of these features are motivated by two facts. First is the proximity of objects for effective interactions and the second is to prevent collisions with objects of the environment.

The virtual humans and their interactions with the environment can be illustrated by the example shown in Fig. 3.4. It can be seen from Fig. 3.4 that proper sitting of the virtual human model can be described by placing the spine supporting the vertical part of the chair while thighs supporting the horizontal part of the chair. Also the human model should not go through any 3D point of the chair. Therefore, if each point of the chair is converted in to a 3D distance field then each point of the virtual human model should lie with a specific distance value from the chair.

The first step of feature extraction involves modeling the environment. The 3D point clouds generated from RGBD SLAM algorithms [30] usually contain a large number of 3D points, and searching for a particular affordance in this large feature space is computationally infeasible. Therefore, it is required to convert these dense point clouds into much lighter abstract representations without losing much data. This is achieved by modeling the environment with a 3D Distance Transform Map $DT(\mathbf{x})$ and a 3D Occupancy Map $OC(\mathbf{x})$, where $\mathbf{x}$ is any 3D position of the environment. The 3D Distance Transform (DT) is a shape representation that indicates the minimum distance from a point in the

environment to the closest occupied voxel. In this approach, the 3D Distance Transform is calculated by using the occupied voxels of 3D point clouds, OC. The distance transform map $DT(\mathbf{x})$ of the occupancy grid map OC can be generated using an unsigned distance function (3.22), that represents the Euclidean distance from each location $\mathbf{x}$ of the environment to the nearest occupied voxel in $OC(\mathbf{x})$.

$$DT(\mathbf{x}) = \min_{O_j \in OC} |O_j - \mathbf{x}| \tag{3.22}$$

The distance transform map given by the the equation (3.22) can be calculated very efficiently, but yet can provide an informative representation of the dense input 3D pointcloud.

The distance features are obtained by moving the human model across the voxels in the environment and calculating a distance measure for each and every skeleton points of the human model. Once the environment is modelled by (3.22), we can effectively calculate distance features of a human skeleton with location and orientation $\mathbf{g}_k = (x_k, y_k, z_k, \theta_k)$ by (3.23), where n is the number of 3D points in the skeleton.

$$[d_1, d_2, ...d_n] = DT(H_w(\mathbf{g}_k)) \tag{3.23}$$

Similarly, we can check for any collisions of a skeleton at any location and orientation, X_k by (3.24). In case of a collision c_i is assigned as 1 and 0 otherwise.

$$[c_1, c_2, ...c_n] = OC(H_w(\mathbf{g}_k)) \tag{3.24}$$

3.2.7.3 Normal Features

The other set of features used for affordance detection is normal features. These normal features represent vertical and horizontal planes of the environments. The selection of these features is motivated by the fact that most of the affordances are supported by vertical and horizontal planes. For example, 'sitting' affordance is supported by a horizontal plane

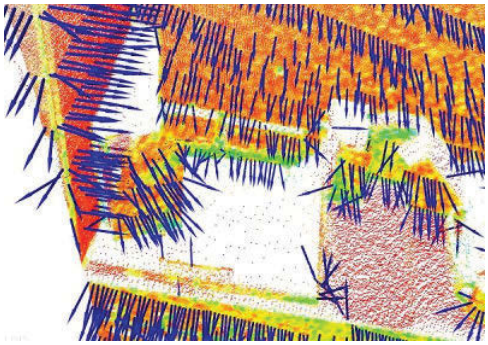


FIGURE 3.5: Surface Normals calculated from point clouds

under the lap of a sitting skeleton and the spine of the skeleton is supported by a vertical plane.

Given a geometric surface, it is usually trivial to infer the direction of the normal at a certain point on the surface as the vector perpendicular to the surface in that point. However, since the point cloud data only contains data points, calculation of surface normals involves two steps. First, surface meshing techniques are used to obtain the underlying surface from the acquired point cloud data. Then surface normals are calculated from this mesh. Fig. 3.5 shows surface normals calculated from a set of point-cloud data.

The normals features for the affordance detection are calculated as follows. First, a 1m x1m x 1m cubic volume is considered from the torso position of a skeleton model at location, (x, y, z, θ). Then it is voxelised into 10cm x 10cm x 10cm voxels. Finally surface normal values of points in each grid cell are averaged to find the normal features for each voxel. Normal features alone could be highly sensitive to observation noise and could affect the classification accuracy. However, averaging normal features over each grid location filters out sensor noise and improves the classifier's performance.

3.3 Experimental Setup

This section explains the experimental setup used for mapping affordances in 3D point-clouds. The same experimental setup is used to evaluate the performances of the SVM classifier and the Flexible Naive Bayes classifier.

FIGURE 3.6: Sample rooms from the dataset

3.3.1 Dataset

Both the SVM and Flexible Naive Bayes classifier discussed in previous sections are supervised learning algorithms. Therefore, they need to be trained first before being used for building an affordance-map in an unseen environment and require a number of 3D scenes to learn parameters of the classification models. These 3D scenes need to be highly informative, as the performances of the proposed affordance detection algorithms depends on the quality of the 3D scenes used for the training. Fortunately, the recent advancements in RGBD Simultaneous Localization and Map Building (SLAM) algorithms [30] build highly informative dense 3D scenes by stitching 3D point clouds frames acquired from low cost 3D depth cameras.

Although there are a number of publicly available 3D point cloud datasets [31, 32], most of them are found to be unsuitable for afordance mapping due to the following reasons. Firstly, 3D scenes in them are not dense enough throughout the environment as they are intended for some other purposes such as object detection [31] or localizations [33]. Secondly, there is not a single dataset that supports all of the affordances tested in this thesis. Therefore, a new dataset is created by capturing a number of dense 3D scenes using an ASUS Xtion depth camera and a 3D map building software [30]. The final dataset consists of 20 high quality 3D scenes captured in office and domestic environments. A few snapshots of the 3D scenes from our dataset are shown in the Fig. 3.6. This dataset is used for training affordances as well as for testing.

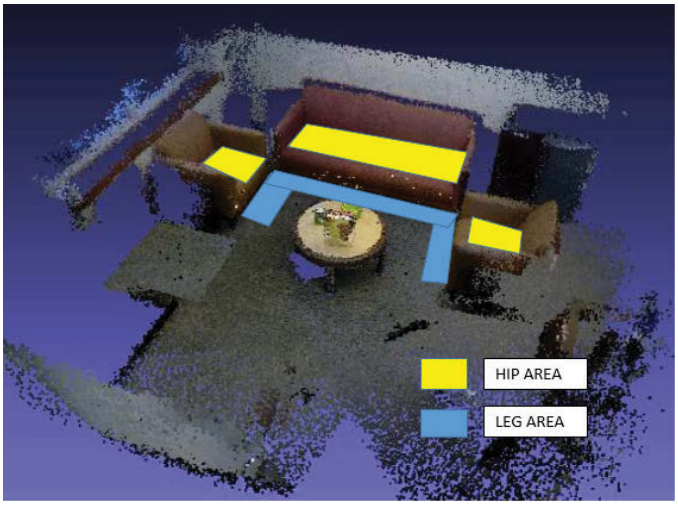


FIGURE 3.7: For sitting-relaxing affordance, class labels are identified by manually marking possible areas of hip and leg locations of the skeleton models

3.3.2 Ground Truth Labels

The proposed affordance-map building process is a supervised learning problem, which requires ground truth labels in order to learn parameters of the classifier. Therefore, all possible locations in 3D images that support the tested affordances need to be identified and labeled as the ground truth. This labeling process could be daunting if it was done manually and therefore a semi autonomous process is employed for ground truth labeling. This process assigns skeleton models to each and every grid location of the map and later uses a filtering process to identify positive labels. For example, the hips and feet of the 'sitting- relaxing' skeleton need to be supported by horizontal surfaces and this restriction can be used to automate the labeling process. First 'sittable' surfaces that support hip joints of skeleton models are identified and manually labeled with rectangular boundaries as shown in Fig. 3.7. Then the same process is carried out to mark the possible leg locations. Finally, all skeletons with hips and feet inside the marked boundaries are filtered out as the positive examples and all other skeletons are labeled as negative examples. The same process is carried out to identify class label for other affordances as well.

3.3.3 Balancing Class Imbalance

The labelling process explained above creates an imbalanced data set with an extremely higher number of negative examples. This is due to the fact that the number of locations that could support a particular affordance are very low when compared to the number of unsupported locations of the room. This class imbalance with a higher number of negative examples could become problematic for both the SVM and Naive Baye's classifiers. For the Flexible Naive Bayes classifier, computation burden of calculating likelihood of the negative class becomes exponentially high as the Kernel Density Estimation (KDE) incorporates all negative examples in its density function. This process becomes practically infeasible in affordance mapping due to the large search space involved. Therefore, KDE for the negative class likelihood is calculated using the K-mean clustering method proposed in [34]. The number of clusters is set equal to the number of positive examples in the dataset by artificially balancing the two classes. On the other hand, the SVM classifier has also reported problems when trained with imbalanced datasets [35, 36]. Therefore, the SVM training negative examples are randomly under-sampled without replacements until the number of examples in the two classes are balanced.

3.3.4 Parameter Selection

Selecting the optimum parameters is important for both SVM and Flexible Bayes Classifier in their training phases.

In the case of the SVM classifier, this involves selecting an appropriate suitable kernel and soft margin parameter C. Out of different kernels tested on the validation set the linear kernel reported the best results. The best value for the C is selected by grid search with exponentially growing sequence of, $C \in \{2^{-5}, 2^{-3}, \ldots, 2^{13}, 2^{15}\}$. The C value with the best cross-validation accuracy reported with the validation set is selected as the optimum C value. Finally, the SVM is trained on the whole training set with the selected C value.

In Flexible Naive Classifier, a suitable kernel and a correct bandwidth are the two main parameters that need to be set. In this experiments, Gaussian Kernel is selected and the bandwidth is set according to the rule of thumb method proposed in [37].

3.4 Results

To evaluate the performances of SVM and Flexible Naive Bayes classifier, a series of experiments are carried out. First, the data set is divided into three subsets ; training set, validation set and testing set. The training set is used to train the model parameters and the validation set is used to fine-tune the input parameters. Finally, trained classifiers are tested on the testing set. This process reduces the possibility of over-fitting the classifiers.

The k-fold cross validation is carried out to report the performances of the two classifiers. In k-fold cross validation, the dataset is divided into k folds and one of them is selected as the testing set. Then the affordances are trained on the rest of the $k-1$ folds and finally tested on the previously left testing set.

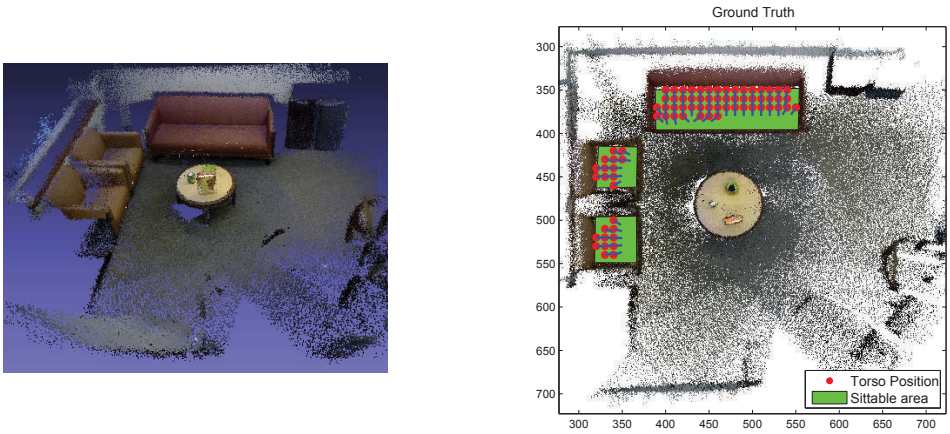
In following sections, experiment results are analyzed using two methods: qualitative analysis and quantitative analysis. In qualitative analysis results of the two classifiers are compared with the ground-truth affordance map, and in quantitative analysis results are compared against various performances measures. These analyses will be used to select the most suitable classifier for mapping affordances in an unseen 3D pointcloud.

3.4.1 Qualitative Analysis

The qualitative analysis is done by selecting a 3D image for each affordance from the dataset and comparing prediction results of the two classifiers. Both 2D affordance maps and 3D skeleton maps are used for this purpose.

3.4.1.1 Sitting-Relaxing Affordance

The Fig. 3.8 shows the 3D and 2D views of the living room which is used to test the Sitting-Relaxing Affordance. It consists of a sofa set, a coffee table in the middle and a few other pieces of furniture around. The sittable area of the sofa set is marked with green colored rectangles.

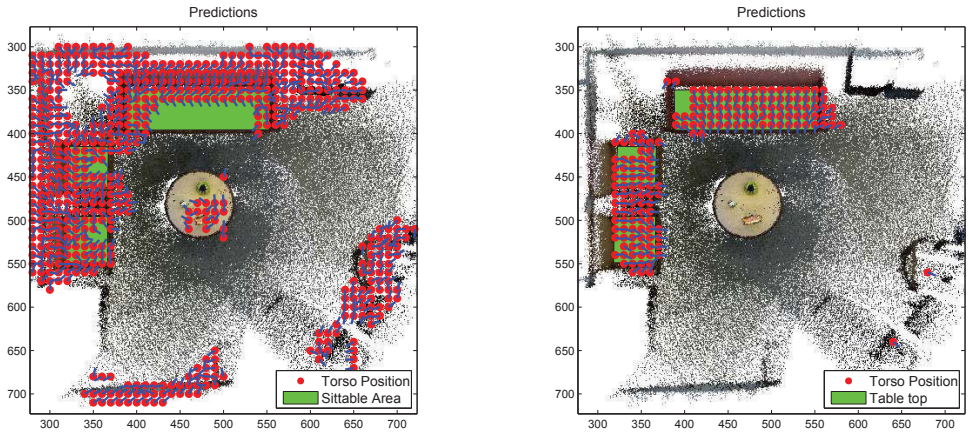


(a) 3D view of the living room

(b) Plan view with ground truth skeleton locations and their orientations

FIGURE 3.8: Test room for the Sitting-Relaxing Affordance. Best viewed in colour

The torso positions of the ground truth skeletons are marked with red circles and the small blue colored lines denote their orientations. The ground truth skeletons of the room are obtained by the method explained in section.3.3.2.



(a) Flexible Naive Bayes

(b) SVM Classifier

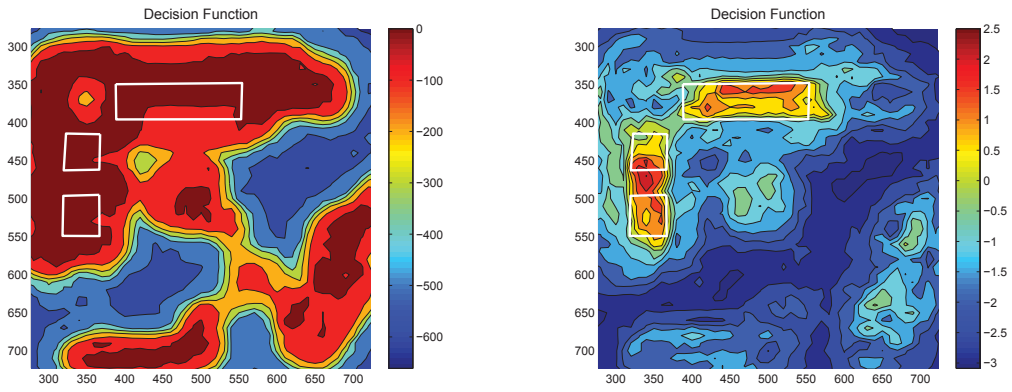
FIGURE 3.9: Classification results of the Sitting-Relaxing Affordance

The classification results of the two classifiers are shown in the Fig. 3.9 where red circles denote torso position of skeleton models with positive class predictions. The blue colored stroke represents orientation of the each skeleton. In this plan view, only the skeleton

model with the highest prediction weight along the Z axis and orientations, $\theta \in \{0..2\pi\}$ is shown for each (x, y) position of the map.

The flexible naive Bayes classifier has produced a large number of false positive predictions particularly at the occupied spaces of the room. Although the Naive Bayes classifier has performed well to discriminate the free spaces from the occupied spaces, it has failed to discriminate "Sittable" locations from other occupied spaces. This is mainly because naive Bayes classifier is not a discriminative classifier and therefore it could not correctly discriminate positive and negative classes. On the other hand, due to the higher number of negative class samples a clustering method was used as explained in section 3.3.4 to estimate kernel densities. Therefore, negative class likelihood is not properly represented. This could have affected the naive Bayes classifier to predict higher number of false positive predictions.

In contrast, the SVM classifier has predicted locations of the Sitting-Relaxing affordance reasonably well. Particularly, it has predicted orientations of the skeleton models well. However, a few false positive predictions can be seen in some locations especially at the space between the two small sofas.



(a) Flexible Naive Bayes

(b) SVM Classifier

FIGURE 3.10: Decision Function values that indicate the confidence level of the predictions

The values of the decision function for each prediction of the two classifiers are shown in the Fig. 3.10. Higher the decision function value, higher the confidence of the positive class predictions. Therefore, the decision function can be used to rank the class predictions according to the confidence level and is a good measure to analyse the quality of the results. The decision function for the the Naive Bayes classifier is the log likelihood ratio of the two classes given by the following equation.

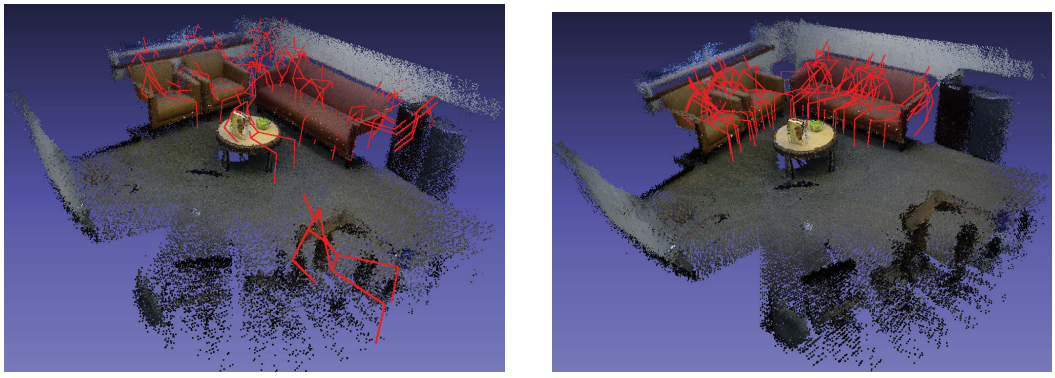
$$F_n(\mathbf{x}) = \log \frac{\prod_{i=1}^{n} p(\mathbf{x_i}|C=1)}{\prod_{i=1}^{n} p(\mathbf{x_i}|C=-1)} \tag{3.25}$$

where $\mathbf{x_i} \in \mathbf{x}$ is the i^{th} feature vector.

The decision function for the SVM classifier becomes,

$$F_s(\mathbf{x}) = \mathbf{w} \cdot \Phi(\mathbf{x}) + b \tag{3.26}$$

In Fig. 3.10 the white bordered rectangular areas denote the sofa set. As predicted, SVM decision function has produced better localized results, whereas the Naive Bayes Classifier has produced high confidence values in most of the area of the room. However, the SVM decision function has produced high confidence values outside the 'sittable' areas. Therefore, the decision boundary needs to be fine tuned to improve the classification accuracy.

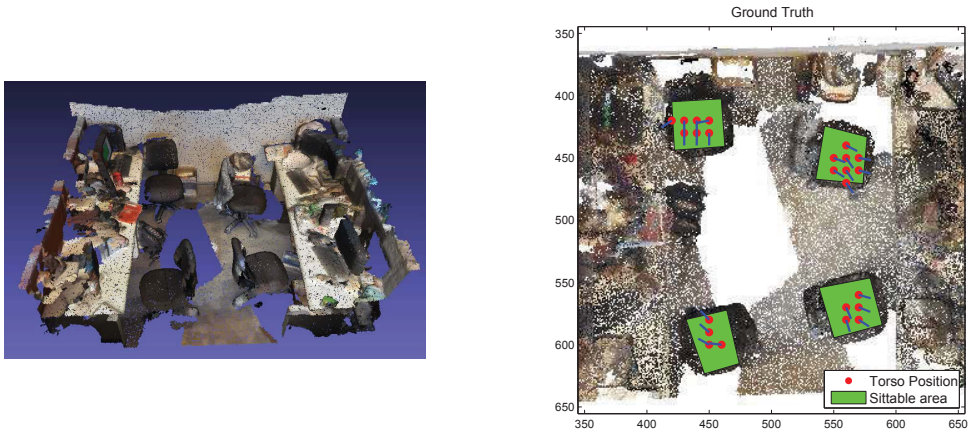


(a) Flexible Naive Bayes (b) SVM Classifier

FIGURE 3.11: Predicted 3D skeleton Map of the room

The Fig. 3.11 shows the predicted 3D skeleton map by the two classifiers. For the sake of clarity, only 20 randomly sampled skeletons are shown in Fig. 3.11. It is clear from these results that the Flexible Naive Bayes classifier has performed poorly by predicting skeletons in many non-sittable areas of the room, whereas, most of the skeletons predicted by the SVM classifier are positioned on the sofa set. However, a fewer number of skeletons predicted by the SVM classifier can be observed in non-sittable areas as well.

3.4.1.2 Sitting-Working Affordance



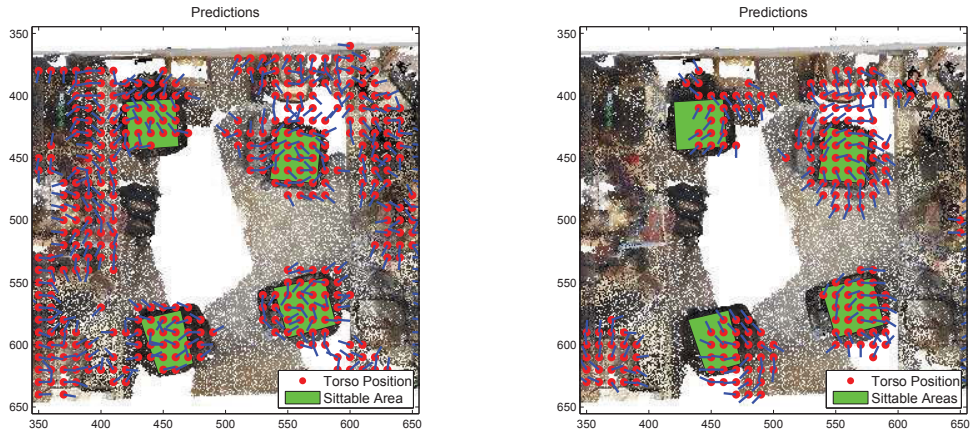
(a) 3D view of the office space

(b) Plan view with ground truth skeleton locations and their orientations

FIGURE 3.12: Test room for the Sitting-Working Affordance. Best viewed in colour

To analyse the quality of the prediction results of the sitting-working affordance, the office space shown in Fig. 3.12 is used. The 3D view of this office room is shown in Fig. 3.12(a) and the 2D plan view with the ground truth is shown in Fig. 3.12(b). The office area consists of four office chairs as shown with green rectangle areas in the Fig. 3.12(b).

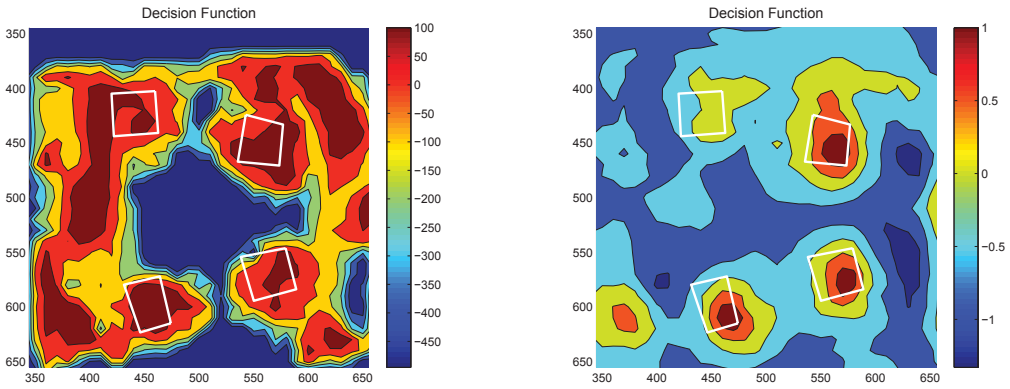
The classification results for the Sitting-Working affordance is shown in Fig. 3.13. Again, the SVM classifier has performed slightly better than the Flexible Naive Bayes classifier. Both of the classifiers have generated a number of false positive predictions. However, the SVM classifier has generated a lesser number of false positives and most of its predictions are located close to the chairs. The SVM classifier has predicted a higher number of false positive predictions near the bottom left corner of the image. The careful observation of



(a) Flexible Naive Bayes (b) SVM Classifier

FIGURE 3.13: Classification results of the Sitting-Working Affordance

this location revealed that there is a computer monitor on the table and pointcloud is not complete in that location. The vertical surface of the computer monitor and horizontal surface of the table resemble very similar features to a chair and that could have mislead the classifier.

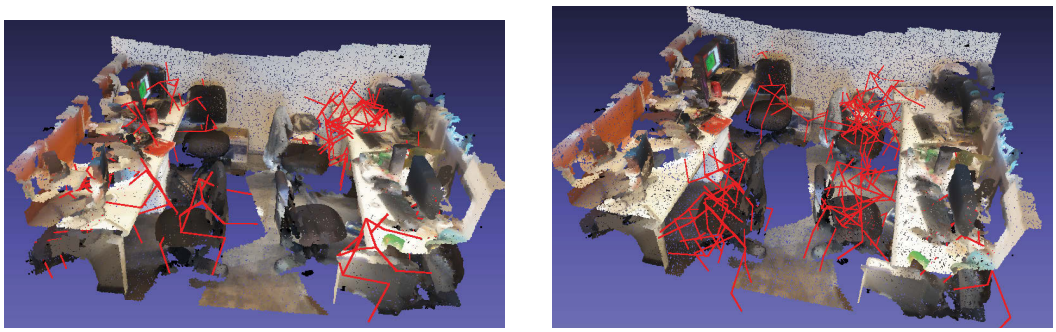


(a) Flexible Naive Bayes (b) SVM Classifier

FIGURE 3.14: Decision Function values that indicate the confidence level of the predictions

The predicted values for the decision functions given by (3.25) and (3.26) are shown as contour maps in Fig. 3.14. The white colored rectangles in Fig. 3.14 represent chairs. As

predicted, the SVM classifier's decision function has produced high confidence values near chairs whereas the Naive Bayes classifier's decision function has produced high confidence values in many areas of the room.

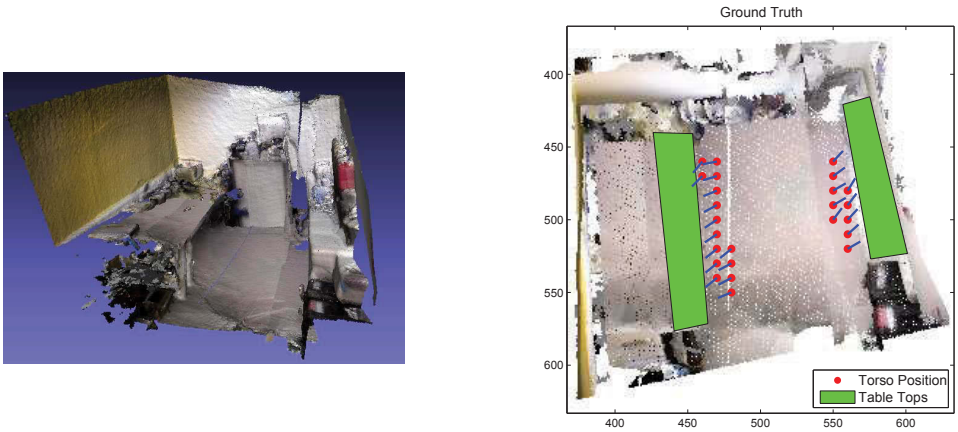


(a) Flexible Naive Bayes (b) SVM Classifier

FIGURE 3.15: Predicted 3D skeleton Map of the room

Fig. 3.15 shows the predicted 3D skeleton map. Both classifiers have predicted 3D skeletons in many random locations. However, the SVM classifier has produced many skeletons on chairs of the office room.

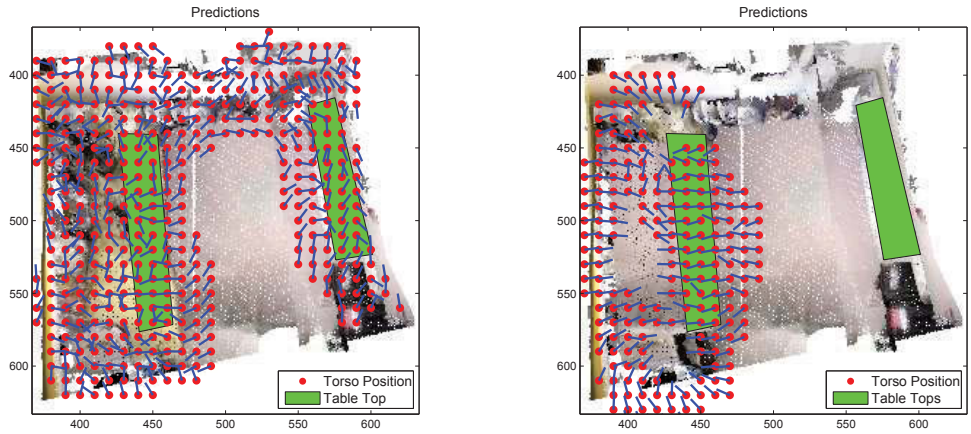
3.4.1.3 Standing-Working Affordance



(a) 3D view of the workshop (b) Plan view with ground truth skeleton locations and their orientations

FIGURE 3.16: Test room for the Standing-Working Affordance. Best viewed in colour

To evaluate the performances of the standing-working affordance, the lab space shown in the Fig. 3.16 is used. The Fig. 3.16(a) shows the 3D view of this lab space and the 2D plan view with the ground truth is shown in the Fig. 3.16(b). The green rectangle areas that can be seen in the Fig. 3.16 are lab workbenches.



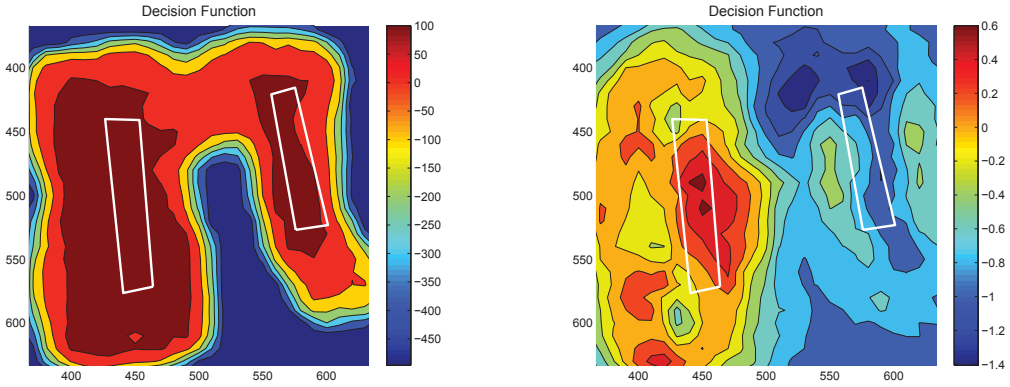
(a) Flexible Naive Bayes (b) SVM Classifier

FIGURE 3.17: Classification results of the Standing-Working Affordance

The prediction results for the Standing-Working affordance is shown in Fig. 3.17. The Flexible Naive Bayes classifier has produced a large number of false positive predictions and they are distributed throughout the room. Although SVM classifier has generated many false positives, they are located close to one of the workbenches and oriented towards it.

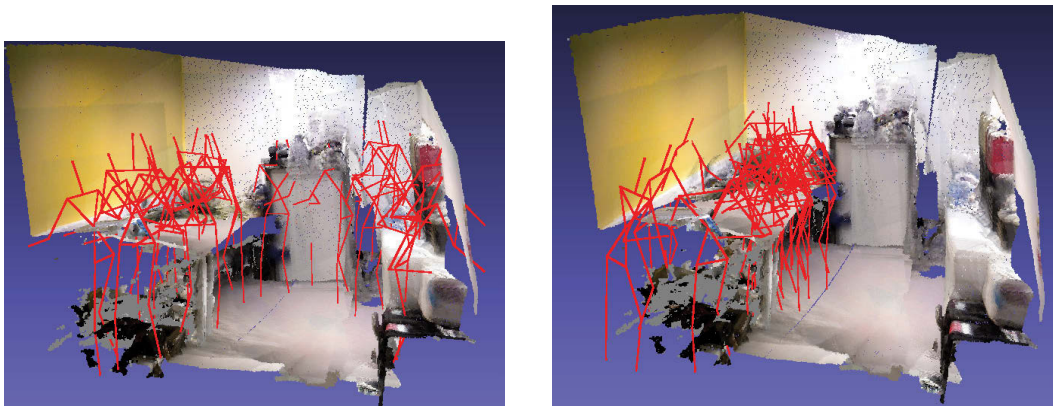
The decision functions values given by (3.25) and (3.26) are shown in the Fig. 3.18. The white color rectangles in the Fig. 3.18 enclose the workbenches. Although the SVM classifier has produced a higher number of false positives, it has assigned high confidence values to the skeletons located close to the ground truth. This is encouraging and the boundary of the decision rule need to be further fine tuned. However, the Flexible Naive Bayes classifier has assigned high confidences to many locations throughout the room generating a large number of false positives.

The Fig. 3.19 shows the predicted 3D skeleton map. It is clear from these results that the SVM classifier has performed slightly better than the Naive Bayes classifier as all of the predicted skeletons of the SVM classifier are oriented around the working bench.



(a) Flexible Naive Bayes (b) SVM Classifier

FIGURE 3.18: Decision Function values that indicate the confidence level of the predictions



(a) Flexible Naive Bayes (b) SVM Classifier

FIGURE 3.19: Predicted 3D skeleton Map of the room

3.4.2 Quantitative Analysis

This section presents the quantitative analysis of the results obtained by the K-fold cross validation. It involves quantifying the cross validation results using four performance measures: precision, recall, accuracy and F-measure. These measures are frequently used to analyse the performances when the dataset is highly imbalanced. The objective of this analysis is to select the best classifier for the affordance map building process. The class label that gives the maximum decision function value along the Z axis is selected

TABLE 3.1: Quantitative Analysis of SVM and Naive Baye's Classifiers

	Naive Bayes				**SVM**			
Affordance	Prec*	Rec**	F1-score	Acc***	Prec*	Rec**	F1-score	Acc***
Sitting-Relaxing	0.09	0.79	0.14	0.94	**0.21**	**1.0**	**0.32**	**0.98**
Sitting-Working	0.02	1.0	0.04	0.95	**0.04**	1.0	**0.08**	**0.97**
Standing-Working	0.02	1.0	0.05	0.96	**0.05**	1.0	**0.10**	**0.98**

Pre* = Precision , Rec** = Recall , Acc*** = Accuracy

as the best prediction for each (x, y) location and orientation (θ) of the map. For a positively predicted skeleton, if there exists a ground truth skeleton at location (x, y) and its orientation is within $\pm\pi/6$ of the ground truth skeleton then that prediction is classified as true positive.

The table 3.1 summarizes the quantitative results of the two classifiers. The SVM classifier has performed better than the Flexible Naive Bayes classifier in precision, F1-score and accuracy. The classification accuracies of both the classifier are very high although other performance measures are relatively very low. This is a clear indication that the dataset is highly imbalanced and it is skewed towards the negative class. The two classifiers have accurately predicted a large proportion of the negative class examples thus improving the overall accuracy. However, both classifiers have predicted a large number of false positives resulting in low precision values. High recall values and low precision values indicate that both the classifiers have a bias towards the positive class. SVM classifier has performed well with a high F1-score in sitting-relaxing affordance. However, its performances in the other two affordances are relatively low.

3.5 Discussion and Limitations

This chapter introduced SVM classifier and Naive Bayes classifier for affordance detection. The quantitative and qualitative results indicate that the SVM classifier has better classification capability than the SVM classifier. In particular, this is clearly noticeable with the 'sitting-relaxing' affordance.

Both classifiers have substantially improved the total classification accuracy although the improvements in other performances such as F1-score, recall and precision are relatively

low. These performance measures indicate that each test room is highly imbalanced with a higher number of negative examples than the number of positive examples. This is true because in any environment there could be only a small number of grid locations that support a particular affordance and the majority of the grid locations would not support it. The high accuracies mean both classifiers have classified a large proportion of the negative examples correctly with a high true-negative rate.

Both classifiers have also recorded high recall values and low precision values. This indicates that the two classifiers are generating a higher number of false positive predictions. That means many negative examples are classified as positive, and this is clearly visible from the qualitative analysis results. This is mainly due to the under representation of the negative class examples in the training phase. In the case of Flexible Naive Bayes classifier, the negative class likelihood distribution can be improved by incorporating more negative examples into the KDE model. However, as the number of examples is increased, the computational complexity of the KDE model gets more demanding. This could become problematic in the inference stage as the Kernel Density Estimation needs to be performed in every grid location of the room. Therefore, such an approach is practically infeasible in most cases. In addition, independent assumptions in the Naive Bayes classifier would not be valid for high dimensional data and that could affect the classifier's performances. Because of these reasons the performances of the Naive Bayes Classifier in mapping affordances may not be further improved.

In the case of SVM classifier the separating hyperplane of two classes can be further fine tuned by incorporating more examples from the negative class to the training phase. This would improve the performances of the SVM classifier in affordance learning. High negative examples in the training set give rise to a higher class imbalance ratio. However, some previous research has concluded that SVM learners tends to provide sub-optimum solutions when trained on highly imbalanced datasets [35, 36]. The next chapter explores this issue and discusses effects of class imbalance on the affordance learning process. It also introduces a few SVM based algorithms that can be learned with imbalanced datasets.

Chapter 4

Evaluation of SVM Learners for Mapping Affordances with Highly Imbalance Datasets

4.1 Introduction

In the previous chapter, SVM and Naive Bayes classifiers are introduced for mapping affordances. The quantitative and qualitative experimental results of these two classifiers demonstrated that the SVM classifier marginally outperformed the Naive Bayes classifier. However, SVM classification results on affordance mapping are still not acceptable for a real world robotic applications and therefore it needs to be further improved.

One logical way to improve the SVM learner is to add more examples to the training stage, so the separating hyperplane of the two classes would be further fine tuned. As all of the positive class examples are used to train the current SVM model, only the negative class examples can be further added to the training set. However, this approach makes the training set highly imbalanced with a higher number of negative examples than the number of positive examples. Although SVMs often work effectively with balanced datasets, they could produce suboptimal results with imbalanced datasets. More specifically, previous research work has shown that an SVM classifier trained on an imbalanced dataset often

Table 4.1: Summary of the class imbalances in the dataset

Affordance	**# Positive Examples**	**# Negative Examples**	**Imabalance Ratio**
Sitting-Relaxing	2457	1285326	1:523
Sitting-Working	213	2570439	1:12067
Standing-Working	391	1284935	1:3286

produces models which are biased towards the majority class and have low performance on the minority class [35, 36].

This chapter explores the behavior of the SVM classifier for affordance learning with different class imbalances. First, it discusses the reasons why SVMs are sensitive to class imbalances. Then it presents various data pre-processing and algorithmic techniques proposed in previous literature to overcome the class imbalanced problem for SVMs. These techniques are also tested for various imbalances to analyse their performances and limitations.

4.2 Soft Margin Optimization with Class Imbalance

Table. 4.1 summarises the number of positive and negative examples found in the dataset for each affordance type. It is clear from this data that all of the affordance types found in the dataset have a higher number of negative examples than positive examples. The sitting-working affordance has recorded the highest class imbalance. This is justified as the sitting surface of an office chair is very small compared with the rest of the room. Any affordance detection algorithm should be able to effectively handle these types of class imbalances in order to effectively build an affordance map in a large room. However, a few researchers have reported that an SVM classifier tend to produce sub-optimum results when trained on highly imbalanced datasets [35, 36].

To understand the impact of the class imbalance on affordance detection, a few experiments are carried out and the experimental results are shown below. In these experiments, K-fold cross validation is done for different class imbalances. Then the classifiers are tested on unseen 3D images and F1-scores are recorded for each setting. The average F1-scores for each affordance type in different class imbalances are shown in Fig. 4.1.

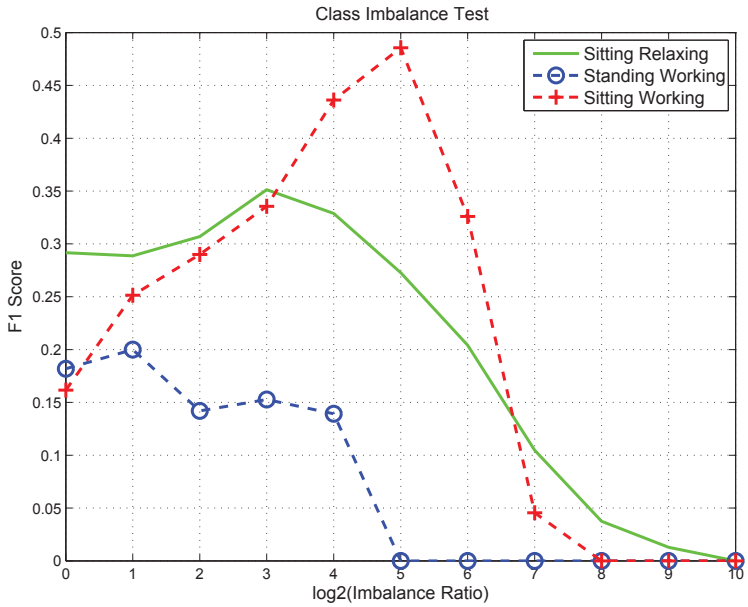
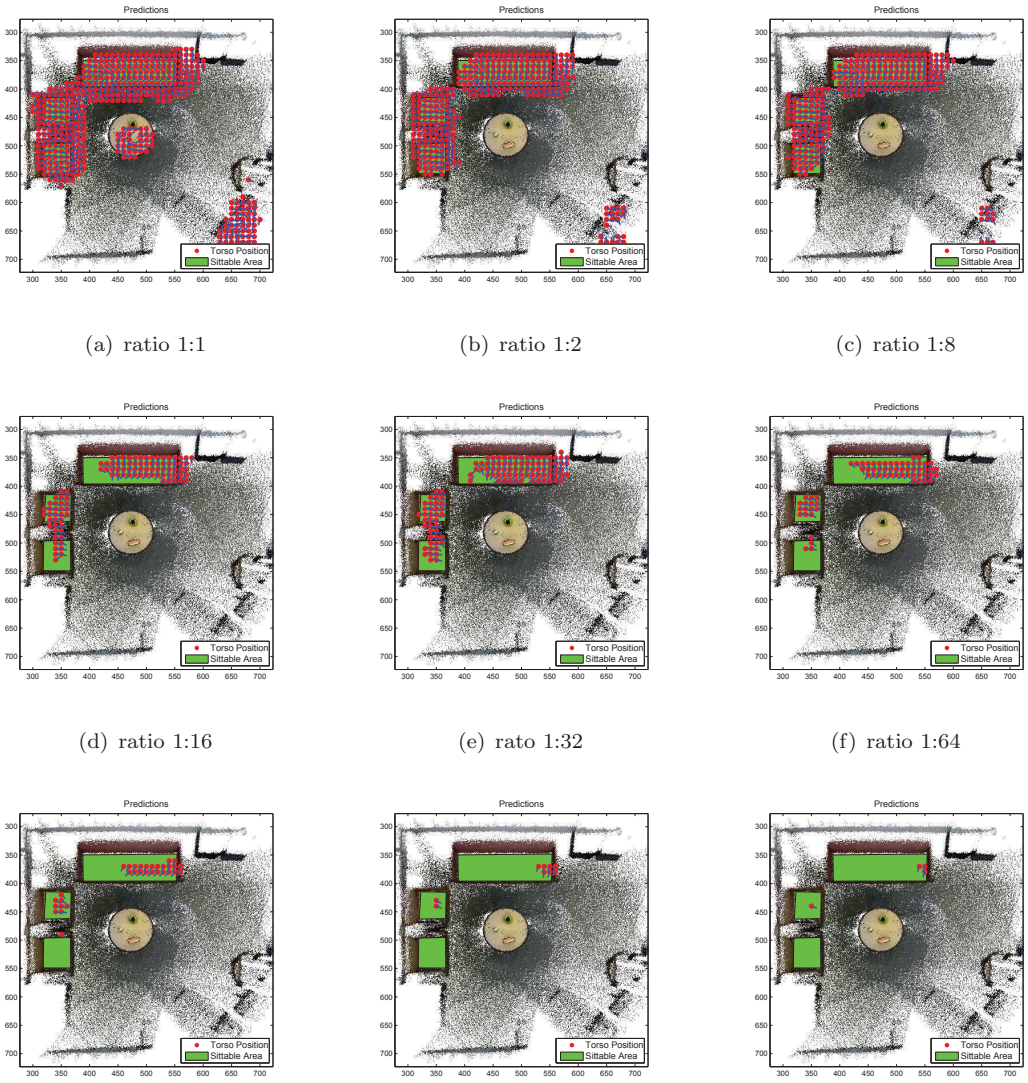


FIGURE 4.1: The effect of class imbalance on the SVM classifier. K-fold cross validation results with average F1-score vs different class imbalances

According to the test results, it is clear that all three affordances have reported low F1-scores initially (when the positive and negative classes are equally balanced) and these have increased slightly before gradually dropping down again. This behaviour is clearly visible with the sitting-working affordace and less visible with the standing-working affordance. When more negative examples are added for training, the performances of the SVM classifiers have gradually increased as can be seen with increasing F1-scores. This is predicted because more negative examples mean the SVM classifier can discriminate the difference between the positive examples and negative examples more effectively. However, the performances of the SVM classifiers have degraded when the class imbalances become extreme. When the class imbalances are too high, all F1-scores have fallen to zero. That means the classifiers have predicted all locations of the room with negative labels. Overall, standing-working affordance has recorded low F1-scores.

The same phenomena can be seen in Fig.4.2. It shows the prediction results of the sitting-relaxing affordance on the plan view of the test room. When the positive and negative examples are balanced, the classifier has produced many false positives. As more and more negative examples are added to the training data, the number of false positives has decreased gradually, and the highest classification result is recorded at the class imbalance



(a) ratio 1:1 (b) ratio 1:2 (c) ratio 1:8

(d) ratio 1:16 (e) rato 1:32 (f) ratio 1:64

(g) ratio 1:128 (h) ratio 1:256 (i) ratio 1:512

FIGURE 4.2: Prediction results for the Sitting-Relaxing Affordance with different positive :negative class imbalance ratios

ratio of 1:32. The further increase of class imbalance ratio causes the classification results to be degraded, resulting in more false-negative predictions, and finally the classifier has labelled the entire room with negative labels.

This unexpected behavior is due to the movement of the separating hyperplane of the SVM classifier toward the majority class and it can be understood by analysing the SVM soft-margin optimization problem.

$$\begin{aligned} \arg\min_{(\mathbf{w},b)} \{\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i\} \\ y_i(\mathbf{w}\cdot\Phi(\mathbf{x_i}) - b) \geq 1 - \xi_i \\ s.t \quad \forall i = 1, \ldots, n \end{aligned} \tag{4.1}$$

The objective function of the standard SVM classifier shown in (4.1) has two terms. The first part tries to maximize the margin while the second part attempts to minimize the penalty associated with the misclassification. The regularization parameter C, which is a constant, balances the trade-off between these two terms and can also be considered as the defining factor of the misclassification cost. As both the positive and negative examples are assigned with the same misclassification cost (C), the penalty term is minimized when the misclassification cost of the training samples become minimized. However, for an imbalanced dataset with a higher number of negative examples there could be more negative examples even near the class boundary where the ideal hyperplane is passing through. This influences the optimization problem, because in order to minimize the total misclassification cost, the separating hyperplane could shift towards the minority class. This undesirable shift can cause more false negative examples by impacting the performances of the minority class. However, the total misclassification cost and the total error would remain low due to the higher number of true negative examples. In cases like affordance detection, where class imbalance is extreme, the SVM could easily produce a skewed hyperplane which would classify all examples as negative.

4.3 Imbalance Leaning Methods for SVMs

As discussed in the previous section behaviour of the SVM classifier with imbalanced data is undesirable. However in many domains training data sets are highly skewed and therefore researchers have tried a variety of techniques to learn SVM models with imbalanced data. This section discusses a few widely used methods and their applicability for learning affordances.

The SVM classifier with imbalanced data can be categorized into two main classes as data pre-processing methods and algorithmic modification methods.

4.3.1 Data pre-processing methods

In data pre-processing methods classifiers are trained on subsets of the original dataset and try to minimize the effects of class imbalance by balancing the training example in the subset. Most of these methods can be used in many classification algorithms that are affected by class imbalance and are not limited only to the SVM algorithm. This section introduces two such popular data preprocessing methods called re-sampling method and ensemble learning method.

4.3.1.1 Re-sampling methods

Re-sampling methods attempt to correct the data imbalance by balancing the number of training samples belonging to each class and training the SVM on the modified balanced dataset. Many researchers have applied re-sampling methods to train SVMs with imbalanced datasets in different domains [36, 38–40]. These methods can be categorized as random under/oversampling, focus under/oversampling and data generation methods like SMOTE [39].

The basic SVM classifier presented in the previous chapter is based on random under sampling. In random under sampling, examples from the majority class are randomly selected and removed from the original dataset until the number of examples from both classes are reasonably balanced. Consequently, under sampling readily gives a simple method for adjusting the balance of the original dataset. However, as shown in section 4.2 under sampling throws away potentially informative examples from the majority class, and it could affect the decision boundary of the SVM adversely resulting a higher number of false positives.

While under sampling removes examples from the original dataset, random oversampling appends examples of the minority class to the dataset by creating multiple copies of examples from the minority class. Although this method balances the class distributions,

it also introduces its own set of problematic consequences. Firstly, adding more examples increases the size of the dataset and learning with an extremely imbalanced dataset would not be feasible. On the other hand, randomly added new examples would not even affect the decision boundary if most of the majority class examples reside near the decision boundary. More specifically, random oversampling could lead to over-fitting since oversampling simply appends replicated data to the original dataset. This will cause the decision rule to become too specific and could worsen the classification performances on unseen testing data even for a classifier with high training accuracy [41].

The data generation methods on the other hand create synthetic data from the minority class in order to balance an imbalanced dataset. The synthetic minority oversampling technique (SMOTE) is one of the data generation methods that has shown a great deal of success in various applications [39]. The SMOTE algorithm creates artificial data around the K-nearest neighbors of each data point of the minority class. However, experiments in [42] have shown that SMOTE could adversely affect the decision boundary when applied to SVM classification. This is because SMOTE makes the strong assumption that the instance between a positive class instance and its nearest neighbors is also positive. However, along the decision boundary of the SVM, the nearest neighbor of a minority class data point could be a data point from the majority class. When SMOTE is applied to a such data point, it could replicate with many majority class data points and class imbalance could be further increased.

In contrast to other sampling methods, focused sampling methods try to select the most informative data points around the decision boundary and use them to balance the two classes. In [43] researchers present an efficient focused sampling method specifically for SVMs. First, the separating hyperplane is found by training an SVM model on the original imbalanced dataset and using it to select the most informative examples lying around the decision boundary. Then these selected examples are used to balance the dataset and a new SVM model is trained on this new dataset. This method is more suited for the affordance learning because all of the negative examples are not required for the SVM learner and therefore the final classifier can be trained on more informative examples of the original dataset.

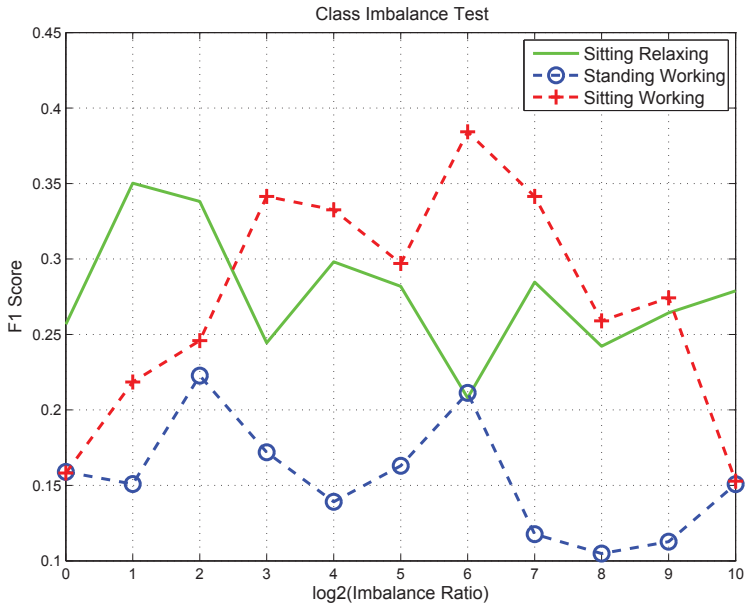


FIGURE 4.3: SVM learning with focus re-sampling

Fig. 4.3 shows the performance of the SVM learner with focus re-sampling for different class imbalances. Initially, F1-score slightly increases as more and more negative class examples are added to the SVM learner. This is clearly visible with the sitting-working affordance. However, when the class imbalance increases, then the performances of the SVM learners gradually degrade in all three affordances. This could be due to the two step learning process used in focus re-sampling. When the dataset is extremely imbalanced, the SVM classifier learned at the first step of the focus re-sampling method could be extremely biased towards the majority class. These examples could lie far away from the optimum separating hyperplane, and when the second classifier is learned, it still could have a bias towards the majority class. However, focus re-sampling has shown some resistance to class imbalance problem when the class imbalances are moderate when compared to the random re-sampling method. Therefore, the focus re-sampling method can handle the problems associated with SVM learners in imbalanced datasets to some extent.

4.3.2 Algorithmic Modifications

The methods discussed in the previous section modify the original imbalanced datasets and apply them into the general SVM framework without modifying the SVM algorithm.

However, SVM algorithm can be modified in order to handle class imbalance internally. This section introduces a few such popular methods that were successfully applied to handle the class imbalance problem associated with SVMs.

4.3.2.1 zSVM Algorithm

zVM is a another SVM imbalance training method proposed in [44]. In this method first the standard SVM classifier is learnt on the original dataset. Then the bias of the separating hyperplane towards the minority class (positive class) is removed by artificially shifting the decision boundary toward the minority class. Theory behind this modification can be understood by looking at the SVM decision function. The decision function of the standard SVM classifier can be rewritten as shown below,

$$\begin{aligned} f(x) &= sign(\sum_{i}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_i) + b) \\ &= sign(\sum_{i}^{n_1} \alpha_i^+ y_i K(\mathbf{x}_i, \mathbf{x}_i) + \sum_{j}^{n_2} \alpha_i^- y_j K(\mathbf{x}_i, \mathbf{x}_i) + b) \end{aligned} \tag{4.2}$$

Where n_1 is the number of positive examples, n_2 is the number of negative examples and α_i^+, α_i^- are positive and negative coefficients of the support vectors. In the zSVM method positive coefficients α_i^+ of the support vectors are multiplied by a small positive value z as shown below.

$$f(x) = sign(z * \sum_{i}^{n_1} \alpha_i^+ y_i K(\mathbf{x}_i, \mathbf{x}_i) + \sum_{j}^{n_2} \alpha_i^- y_j K(\mathbf{x}_i, \mathbf{x}_i) + b) \tag{4.3}$$

This modification increases the weight of the positive support vectors by artificially modifying the bias of the original hyperplane. The optimum value for the z can be obtained by testing on a validation set.

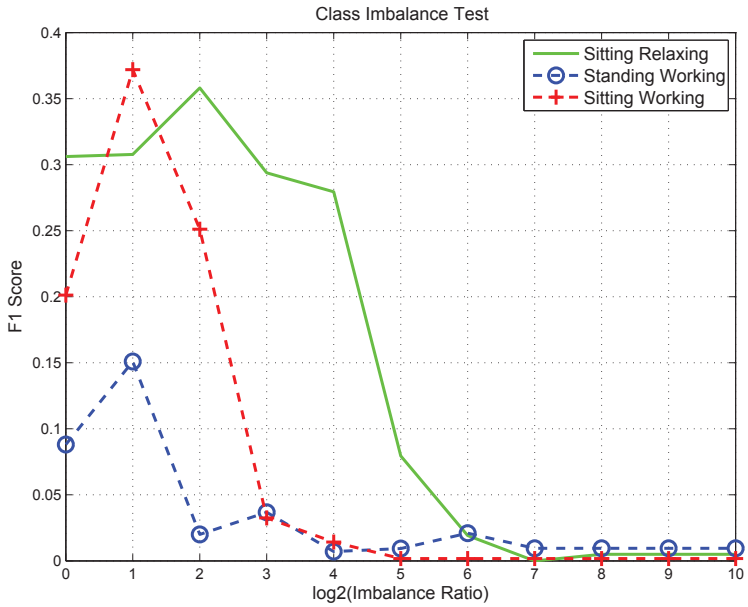


FIGURE 4.4: SVM learning with zSVM method

Fig. 4.4 shows the performance of the zSVM method in different class imbalances. Although it has shown a slight increase, in small class imbalances its performance has rapidly dropped as the class imabalance increases. This could be due to selecting an incorrect value for z parameter of the Eq. 4.3. In particular, the separating hyperplane of the zSVM classifier is very sensitive to the z value and could easily get biased to one of the classes. On the other hand, z value is chosen by testing on a validating set. Therefore, the value of the z could easily over-fit to the validation set by degrading the classifier's performances on the testing set.

4.3.2.2 Different Cost Model

As explained in the previous section, the main reason for the inability of the soft margin SVM to learn the separating hyperplane accurately when the dataset is imbalanced would be that it assigns equal cost values for both the minority and majority class misclassification in the penalty term. This would create a biased model with a hyperplane that is skewed towards the minority class. The different cost model proposed in [45] has been designed to mitigate this issue by assigning different penalty terms for two classes. By assuming that the positive class is the minority class, misclassification cost of C^+ can be

assigned for the positive class and C^- cost can be assigned for the negative class in the objective function of the soft margin SVM optimization equation as shown in (4.4).

$$\begin{aligned} \arg\min_{(\mathbf{w},b)}\{\frac{1}{2}\|\mathbf{w}\|^2 + C^+ \sum_{i|y_i=+1}^{n} \xi_i + C^- \sum_{i|y_i=-1}^{n} \xi_i\} \\ y_i(\mathbf{w}\cdot\Phi(\mathbf{x_i}) - b) \geq 1 - \xi_i \\ s.t \quad \forall i = 1, \ldots, n \end{aligned} \tag{4.4}$$

The effect of the imbalanced data is minimized by assigning higher misclassification cost for the positive class (i.e.,$C^+ > C^-$) which eventually would fix the bias of the separating hyperplane. Therefore the optimization would try to balance the total positive and total negative cost values and the separating hyperplane would not skew towards the positive class examples. The dual Lagrangian form of this modified objective function can be represented as follows:

$$\begin{aligned} \max_{\alpha_i}\{\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i\cdot\mathbf{x}_j)\} \\ s.t. \quad \sum_{i=1}^{n} y_i\alpha_i = 0, \quad 0 \leq \alpha_i^+ \leq C^+, \quad 0 \leq \alpha_i^- \leq C^- \quad i = 1, \ldots, n \end{aligned} \tag{4.5}$$

where α_i^+ and α_i^- represent the Lagrangian multipliers of positive and negative examples, respectively. The cost values C^+ and C^- can be chosen by the rule of thumb method proposed in [45].

$$\frac{C^+}{C^-} = \frac{Number \quad of \quad negative \quad examples}{Number \quad of \quad positive \quad examples} \tag{4.6}$$

The behavior of the SVM learner in different class imbalances with different cost models is shown in Fig. 4.5. Initially F1-scores of all three affordances have gradually increased as the class imbalance ratio increases. A steep increase is visible in the sitting-working

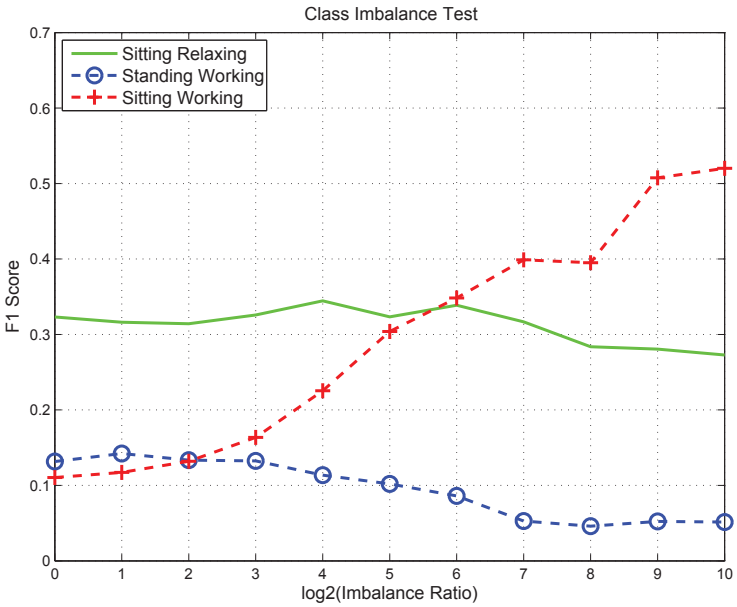


FIGURE 4.5: SVM learning with different cost models. K-fold cross validation results

affordance and has resisted extreme class imbalances well. There is a marginal drop of F1-score in the sitting-relaxing affordance when the class imbalance increases but overall it has shown a steady performance. However, F1-score of the standing-working affordance has declined as the class imbalances become extreme but has not fallen to zero. Overall, the different cost model has shown some consistent performances in different class imbalances.

4.4 Discussion and Limitations

Table 4.2 summarizes the results of the imbalance test. Focus re-sampling method, DCM-SVM method and zSVM method have recorded better F1-scores than the random sampling methods in extreme class imbalances. However, it is hard to predict the behaviour of these algorithms in different class imbalances. The observations obtained in the class imbalance test have also revealed the following implementation difficulties.

The first difficulty is, how to select the best SVM algorithm for inferring affordances in a new room. One option is to select the SVM algorithm that outputs the best F1-score on a validation set. According to the results of the imbalance test, the three affordances have recorded their highest F1-score in three different algorithms. If such a method was chosen

TABLE 4.2: Imbalance Test Result Summary

	Affordance Type	Highest F1-Score	Imabalance Ratio
Random Sampling	Sitting-Relaxing	0.35	1:8
	Standing-Working	0.19	1:2
	Sitting-Working	0.48	1:32
Focus Sampling	Sitting-Relaxing	0.35	1:2
	Standing-Working	**0.23**	1:4
	Sitting-Working	0.38	1:64
zSVM method	Sitting-Relaxing	**0.36**	1:8
	Standing- Working	0.15	1:2
	Sitting-Working	0.37	1:2
Different Cost Model (DCM-SVM)	Sitting-Relaxing	0.34	1:4
	Standing-Working	0.14	1:2
	Sitting-Working	**0.52**	1:1024

then three different SVM models that are trained on three different algorithms would be selected for inferring affordances. However, it is difficult to generalize such an approach to a large set of affordance types.

The second difficulty is to incorporate all training data into the training set. This is found to be difficult because of the following reasons. Firstly, all the algorithms tested in the imbalance test have recorded low performances when the class imbalances become extreme although some of the algorithms have shown some resistance to moderate class imbalances. Secondly, training becomes computationally difficult with a large set of examples as all examples are considered for optimization. This was evident in the imbalanced test as the training beyond 1:2048 imbalanced ratio was difficult due to computer memory limitation (The test was done on a High Performance Cluster Computer with 16 GB memory).

On the other hand, except in the DCM-SVM method, class imbalance is not considered in the SVM optimization problem. All these algorithms are first trained on the original imabalanced dataset and then the separating hyperplane is shifted artificially to reduce the effect of the class imbalance. In the case of DCM-SVM, the method of selecting the cost model is rather arbitrary and therefore the calculated hyperplane could be sub-optimum.

Because of these reasons SVM based algorithms discussed in this chapter are not satisfactory enough for learning affordance. Particularly, their performances on the 'standing-working' accordance is comparably low. The next chapter addresses these shortcomings and formulates the affordance learning as structured output learning which can be solved using the Structured output SVM (S-SVM) algorithm.

Chapter 5

Structured SVM for Learning Affordance Map

5.1 Introduction

The previous chapter discussed SVM learning in imbalanced datasets and its inability to produce acceptable results when the class imbalance become severe. This is particularly true for affordance learning where even in a single image class imbalance is high. Therefore, when more images are added for training the class imbalance becomes extreme. As shown in the previous chapter, even some algorithms that showed good results in some other imbalanced datasets failed to provide acceptable results with affordance learning when the class imbalance become extreme. On the other hand, learning the conventional SVM in a large dataset with high dimensional data becomes practically infeasible.

However, similar class imbalances exist in some machine learning problems and the generalization of the conventional SVM for structured problems called Structured SVM (S-SVM) has been applied successfully to learn discriminate models in extreme class imbalances [35, 46]. Consider the example of binary text classification with the topic 'machine learning' and 'other topics'. As in many other text classification tasks, in the dataset of this problem there could be only less than 1% of documents with the topic 'machine learning'. In these situations a performance measure like error-rate becomes meaningless as

the classifier that classifies all documents as 'other' already has a great error-rate which is hard to beat in the optimization. To overcome this problem the information retrieval community has designed other performance measures like F1-score and precision/ recall which are meaningful even when the datasets are imbalanced.

What does this mean for learning affordances in an imbalanced dataset? Instead of optimizing the error rate during training, which is what conventional SVMs and most other learning algorithms do, it seems like a natural choice to have the learning algorithm directly optimize on performance measures like the F1-score. This is why the binary classification needs to be defined as a structured problem, since unlike error rate, F1-score is not a function of individual examples, but a function of a set of examples. For example, consider a 3D image with n number of grid locations with labels $\bar{\mathbf{y}}_k = (y_1, ..., y_n)$, $y_i \in \{-1, +1\}$ and a feature vector $\bar{\mathbf{x}}_k = (x_1, ..., x_n)$, $x_i \in \Re^n$. Then for each predicted set of labels $\bar{\mathbf{y}}'_k$ there exist a F1-score $F(\bar{\mathbf{y}}_k, \bar{\mathbf{y}}'_k)$ with respect to the true labeling $\bar{\mathbf{y}}_k$ which can be optimized directly in training. This structured problem can be learned efficiently using Structured SVM (S-SVM).

The S-SVMs inherit the attractive features of conventional SVM namely a convex training problem, flexibility in the choice of loss function, and the opportunity to learn nonlinear rules via kernels. On the other hand it has the ability to build upon the underlying structure of data like most generative models (e.g., Markov Random Field). However, unlike generative models S-SVM does not assume feature independence and is a discriminative training model. This features make the S-SVM a suitable candidate for affordance learning.

The purpose of this chapter is to introduce the S-SVM algorithm for affordance learning. It first formulates the affordance learning as a structured output problem by describing required algorithms. Then results of the quantitative and qualitative test are presented to evaluate its performances. As a contribution, this chapter formulates affordance mapping as a structured SVM problem and its performances are experimentally evaluated.

5.2 Affordance Learning with SVM Optimized for Performance measures

Given a set of 3D pointcloud images $\{i_1, ..., i_k\} \subset I$ and their associated affordance-map Y, the goal is to learn mapping $g : I \rightarrow Y$ which can be later used to automatically build an affordance-map in unseen pointclouds. The apparent solution to this problem is to define a hypotheses h as a function that matches a single feature vector x at skeleton location $X_L = \{x, y, z, \theta\}$ of each image to a single label $y \in \{-1, +1\}$,

$$h : X \rightarrow Y \tag{5.1}$$

Learning this mapping function is non-trivial due to the challenges discussed in the previous section. Therefore, the following SVM algorithm that optimized on non-linear performance measures are proposed to learn and infer the affordances. This can be done by defining this learning problem as a multivariate prediction problem.

Once feature vectors $\bar{\mathbf{x}}_\mathbf{k} = (\mathbf{x_1}, ..., \mathbf{x_n})$ for each skeleton location $\mathbf{g}_j = (x_j, y_j, z_j, \theta_j)$ of the 3D pointcloud k and its associated label $\bar{y}_k = (y_1, ..., y_n)$ are known the goal is to find a new function $\bar{h}$ that maps a tuple $\bar{\mathbf{x}}_\mathbf{k} \in \bar{X}$ of n feature vectors to a tuple of $\bar{y}_k \in \bar{Y}$ of n labels

$$\bar{h} : \bar{X} \rightarrow \bar{Y} \tag{5.2}$$

where $\bar{X} = X \times ... \times X$ and $\bar{Y} \subseteq \{-1, +1\}^n$ is the set of all possible label tuples. The following discriminant function is used to implement the proposed multivariate mapping

$$\bar{h}_w(\bar{\mathbf{x}}) = \arg\max_{\bar{y}' \in \bar{Y}} \{\mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}')\} \tag{5.3}$$

where $\mathbf{w}$ is the parameter vector and Ψ is the feature function which defines the relationship between features, $\mathbf{x}$ with the output $\bar{y}'$. The above function $\bar{h}_w(\bar{\mathbf{x}})$ returns the set of labels $\bar{y}' = (y_1', ..., y_n')$, which score high values according to the discriminant function,

$\bar{h}_w(\bar{\mathbf{x}})$. Whether the *argmax* in equation (5.3) can be efficiently calculated solely depends on the structure of the feature function, Ψ. If the feature function is defined as a linear combination of labels and features as shown in (5.4) then (5.3) can be calculated efficiently.

$$\Psi(\bar{\mathbf{x}}_k, \bar{y}_k) = \sum_{i=1}^{n} y_i \mathbf{x}_i \tag{5.4}$$

As the feature function given in (5.4) is linearly decomposable over $\bar{y}^{'}$, the solution can be calculated by maximizing $\bar{y}^{'}$ element wise. This makes the computation very fast and efficient.

The major advantage of multivariate rule $\hat{h}$ for SVM optimization is that it allows the inclusion of a loss function Δ that is based on a set of examples (F_1 score, Precision / Recall) rather than optimizing on a single example-based loss function like error rate.

To train the discriminant $\bar{h}_w$ function the following generalization of the support vector machine has been used.

Optimization Problem

$$\begin{aligned} \min_{w,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_k \\ \text{s.t.} \quad \xi_k \geq 0, \forall k \\ \mathbf{w}^T[\Psi(\bar{\mathbf{x}}_k, \bar{y}_k) - \Psi(\bar{\mathbf{x}}_k, \bar{y}'] \geq \triangle(\bar{y}', \bar{y}_k) - \xi_k, \quad \forall k, \quad \forall \bar{y}' \in \mathcal{Y} \setminus \bar{y}_k \end{aligned} \tag{5.5}$$

where $\triangle(\bar{y}', \bar{y}_k)$ is a loss function that decreases during training as the the predicted tuple of outputs $\bar{y}'$ approaches the ground truth lable tuple, $\bar{y}_k$. This optimization is convex but in contrast to the standard SVM the above optimization problem has an infeasibly large number of constrains (number of training examples into each possible tuple of $\bar{y}_k \in \mathcal{Y}$), which makes this optimization intractable. However, by using the sparse approximation algorithm proposed by [47] the above optimization problem can be solved in a polynomial time for many types of multivariate loss functions. This will be detailed in the next section. The objective function that needs to be minimised (5.5), is a trade-off between the model

complexity, $\| \mathbf{w} \|$, and the hinge loss relaxation of training loss, $\sum \xi_k$. Here $C > 0$ is a constant that controls this trade-off.

5.2.1 Efficient Learning Algorithm

The key question that needs to be solved at the learning phase is how can the optimization problem in equation (5.5)be solved despite the large number of constraints? Since there is a constraint for each possible tuple of $\bar{y} \in \mathcal{Y}$ in every single point cloud k in the dataset, simply enumerating all constraints and solving the quadratic problem in equation (5.5) is practically impossible.

Algorithm 1: for training structural SVMs (margin-rescaling)

1 Input: $S = ((\bar{\mathbf{x}}_1, \bar{y}_1), ..., (\bar{\mathbf{x}}_\mathbf{k}, \bar{y}_k)), C, \varepsilon$;
2 $\mathcal{W} \leftarrow \emptyset, \mathbf{w} = \mathbf{o}, \xi_k \leftarrow$ for all $k = 1, ..., n$;
3 **repeat**
4 **for** $k = 1, ..., n$ **do**
5 $\bar{y}^{'} \leftarrow \arg\max_{\bar{y}' \in \bar{Y}} \{\triangle(\bar{y}', \bar{y}_k) + \mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}^{'})\}$;
6 **if** $\mathbf{w}^T[\Psi(\bar{\mathbf{x}}_k, \bar{y}_k) - \Psi(\bar{\mathbf{x}}_k, \bar{y}] < \triangle(\bar{y}', \bar{y}_k) - \xi_k - \varepsilon$ **then**
7 $\mathcal{W} \leftarrow \mathcal{W} \cup \{\mathbf{w}^T[\Psi(\bar{\mathbf{x}}_k, \bar{y}_k) - \Psi(\bar{\mathbf{x}}_k, \bar{y}] \geq \triangle(\bar{y}', \bar{y}_k) - \xi_k\}$
8 $(\mathbf{w}, \xi) \leftarrow \arg\min 6_{w, \xi \geq 0} \frac{1}{2}\|w\|^2 + C \sum_{k=1}^{m} \xi_k \quad s.t \quad \mathcal{W}$
9 **end**
10 **end**
11 **until**;
12 $\mathcal{W}$ has not changed during iteration ;
13 return $(\mathbf{w}, \xi)$

Therefore the cutting plane algorithm proposed by [47] has been used to iteratively train parameters of the discriminant function (see algorithm 1). It iteratively builds a sub-set of working constrains, $\mathcal{W}$ that is equivalent to the full set of constrains specified in (5.5) up to the precision ε. The algorithm first starts with an empty set of $\mathcal{W}$ and $\mathbf{w} = 0$ iterates through the samples of the dataset. At each iteration, the algorithm finds the most violated constraint (Line 5), *i.e* the constraint related to the label $\bar{y}^{'}$ that maximizes $\triangle(\bar{y}', \bar{y}_k) + \mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}^{'})$. If this constraint is violated by more than ε (Line 6) then it is

added to the the current working constraint set $\mathcal{W}$ and new $\mathbf{w}$ is calculated by solving the quadratic program over the new working constraint set, $\mathcal{W}$ (Line 6). Finally the algorithm stops when no constraint of the optimization problem in (5.3) is violated more than ε. Therefore the solution obtained through the algorithm fulfills all constraints up to the precision ε, and the norm of $\mathbf{w}$ is not bigger than the norm of the exact solution. Interestingly the proposed cutting plane algorithm always terminates in a polynomial number of iterations as shown in [47]. It has also been shown that the refined version of the algorithm terminated after adding most of $O(C\varepsilon^{-1})$ constraints that are independent of the size of the output space $\bar{\mathcal{Y}}$ and the number of training examples. This makes the proposed algorithm an attractive training solution to learn the affordance-map.

5.2.2 Maximization Step at Inference

Since complexity of the proposed structured SVM method largely depends on efficient calculation of *argmax*s in inference equation 5.3 and training algorithm 1, it is important that these maximizations can be computed efficiently. At inference step we need to compute,

$$\bar{h}_w(\bar{\mathbf{x}_k}) = \arg\max_{\bar{y}' \in \bar{Y}} \{\mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}')\} \tag{5.6}$$

By substituting feature function, (5.4) in (5.6) the following form of the inference equation can be obtained.

$$\bar{h}_w(\bar{\mathbf{x}_k}) = \arg\max_{\bar{y}' \in \bar{Y}} \{\mathbf{w}^T \sum_{i=1}^{n} y_i' \mathbf{x}_i\} \tag{5.7}$$

Since the feature function is linearly decomposable over single label, y_i' the above inference equation can be converted to a simple form as shown below.

$$\bar{h}_w(\bar{\mathbf{x}_k}) = \arg\max_{\bar{y}' \in \bar{Y}} \{\sum_{i=1}^{n} \mathbf{w}^T y_i' \mathbf{x}_i\} \tag{5.8}$$

The *argmax* in equation (5.8) can be further decomposed into a single feature vector and single label as follows, which makes structured SVM equivalent to the conventional SVM prediction problem.

$$* y_i^{'} = \underset{y_i^{'} \in \bar{y}^{'}}{\arg\max} \{\mathbf{w}^T y_i^{'} \mathbf{x}_i\}, \quad \forall y_i^{'} \in \bar{y}^{'} \tag{5.9}$$

As each label is a binary value with $y_i^{'} = \{+1, -1\}$ computing prediction becomes very efficient with

$$\bar{h}_w(\bar{\mathbf{x}_k}) = \underset{\bar{y}^{'} \in \bar{Y}}{\arg\max} \{\mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}^{'})\} = (sign(\mathbf{w}.\mathbf{x}_1), ..., sign(\mathbf{w}.\mathbf{x}_n)) \tag{5.10}$$

Therefore, the linear feature function Ψ makes predictions very fast by considerably reducing the size of the output space ($\bar{Y}$) from 2^n to n, where n is the number of different grid locations in the map.

5.2.3 Maximization Step at Learning

However, computing the loss-augmented argmax (Line 6 , Algorithm 1) that is required for training is more complicated and depends on the type of loss function used. Therefore, maximization step at training needs a sophisticated algorithm that allows efficient training.

The step that calculates the most violated constraint in algorithm 1 needs to complete the following maximization step shown in equation (5.11).

$$\bar{y}^{'} \leftarrow \underset{\bar{y}^{'} \in \bar{Y}}{\arg\max} \{\triangle(\bar{y}', \bar{y}_k) + \mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}^{'})\} \tag{5.11}$$

If the loss function $\triangle(\bar{y}', \bar{y}_k)$ is linear in $\hat{y}^{'}$ (Eg : Hamming Loss) then the solution for (5.11) can be computed by maximizing $\bar{y}^{'}$ element wise. However, linear loss functions like, hamming loss cannot handle the label bias problem associated with highly unbalanced classes as discussed in Chapter 4. To overcome this problem, it seems like a natural choice

to directly optimize for performance measures like F_1 score which is the harmonic mean of precision and recall. This can be done in the structural SVM framework by incorporating a loss function based on the contingency table at the maximization step (Line 6) of algorithm 1.

5.2.3.1 Loss Function Based on Contingency Table

When the loss function becomes non-linear in $\bar{y}^{'}$, computation of the $argmax$ in Eq. (5.11) is difficult because an exhaustive search over all possible $\bar{(y)}^{'}$ is infeasible. However, the computation of the $argmax$ can be categorized over all possible contingency tables so each sub-problem can be solved efficiently. The contingency table for a binary classification is shown in Table 5.1, where a is the number of true positives, b is the number of false positives, c is the number of false negatives and d is the number of true negatives.

	y=1	y=-1
h(x)=1	a	b
h(x)=-1	c	d

TABLE 5.1: Contingency table for Binary Classification

It can be observed that there are only order $O(n^2)$ of different contingency tables for a binary classification problem with n samples. Therefore, if the loss function $\triangle(a, b, c, d)$ is computed from the Table 5.1 then the loss function can only have at most $O(n^2)$ different values.

With the Loss function $\triangle(a, b, c, d)$, $argmax$ for calculating the most violated constraint can be redefined as in (5.12) .

$$\bar{y}^{'} \leftarrow \arg\max_{\bar{y}^{'} \in \bar{\mathcal{Y}}} \{\triangle(a, b, c, d) + \mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}^{'})\} \tag{5.12}$$

Although computation of $argmax$ in (5.12) is exhaustive, it can be simplified by stratifying search space of $\bar{\mathcal{Y}}$ over different contingency tables. Algorithm 2 details this process. For each contingency table (a, b, c, d), it calculates $argmax$ over all $\bar{\mathcal{Y}}_{abcd}$ which is a subset of $\bar{y}^{'}$ that correspond to the selected contingency table.

$$\bar{y}_{abcd} = \underset{\bar{y}' \in \bar{\mathcal{Y}}_{abcd}}{\arg\max}\{\mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}')\} \tag{5.13}$$

$$\bar{y}_{abcd} = \underset{\bar{y}' \in \bar{\mathcal{Y}}_{abcd}}{\arg\max}\{\mathbf{w}^T \sum_{i=1}^{n} y_i' \mathbf{x}_i\} \tag{5.14}$$

As in the inference function, the equation in (5.14) is linear in $\bar{y}'$. Therefore the solution can be obtained by maximizing $\bar{y}'$ element wise. The maximum value of the object function for a particular contingency table is achieved when the a positive examples with the largest value of $(\mathbf{w}^T \mathbf{x}_i)$ are classified as positive and the d negative examples with the lowest value of $(\mathbf{w}^T \mathbf{x}_i)$ are classified as negative. Finally, overall $argmax$ can be found by maximizing over each of these maxima plus their loss function value.

Algorithm 2: for training structural SVMs (margin-rescaling)

Input: $\bar{\mathbf{x}}_k = (\mathbf{x}_1, ..., \mathbf{x}_n), \bar{y} = (y_1, ...y_n), \quad and \quad \bar{\mathcal{Y}}$;
$(i^p_1, ..., i^p_{\#pos}) \longleftarrow \text{sort}\{i : y_i = 1\} \quad \text{by } \mathbf{w}^T \mathbf{x}_i$;
$(i^p_1, ..., i^p_{\#neg}) \longleftarrow \text{sort}\{i : y_i = -1\} \quad \text{by } \mathbf{w}^T \mathbf{x}_i$;
for $a \in [0, ..., \#pos]$ **do**
    $c \longleftarrow \#pos - a$;
    set $y'_{i^p_1}, ..., y'_{i^p_a}$ to 1 AND set $y'_{i^p_{a+1}}, ..., y'_{i^p_{\#pos}}$ to -1 ;
    for $d \in [0, ..., \#neg]$ **do**
        $b \longleftarrow \#pos - d$;
        set $y'_{i^n_1}, ..., y'_{i^n_b}$ to 1 AND set $y'_{i^n_{b+1}}, ..., y'_{i^n_{\#neg}}$ to -1 ;
        $\upsilon \longleftarrow \triangle(a, b, c, d) + \mathbf{w}^T \sum_{i=1}^{n} y'_i \mathbf{x}_i$;
        if υ *is the largest so far* **then**
            $\bar{y}^* \longleftarrow (y'_1, ..., y'_n)$
        end
    end
end
return $\bar{y}^*$

5.2.4 Experiments

This section explains implementation details of the affordance map building process using the structured SVM. The algorithms explained in previous sections can be easily incorporated into the publicly available SVM^{struct} [21] implementation which is a common framework to predict multivariate or structured outputs. It needs the definitions of the following four functions.

The first is the feature function defined by (5.4).

$$\Psi(\bar{\mathbf{x}}_k, \bar{y}_k) \tag{5.15}$$

The second is the loss function,

$$\triangle\,(\bar{y}', \bar{y}_k) \tag{5.16}$$

which the contingency table based loss function described in section. 5.2.3.1.

The third is a function that calculates the most violated constraint in the algorithm 1

$$\arg\max_{\bar{y}' \in \bar{Y}} \{\triangle(\bar{y}', \bar{y}_k) + \mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}')\} \tag{5.17}$$

The algorithm 2 is proposed to obtain the solution for (5.17).

Finally the SVM^{struct} requires the inference function,

$$\arg\max_{\bar{y}' \in \bar{Y}} \{\mathbf{w}^T \Psi(\bar{\mathbf{x}_k}, \bar{y}')\} \tag{5.18}$$

to predict the affordance map for a new image.

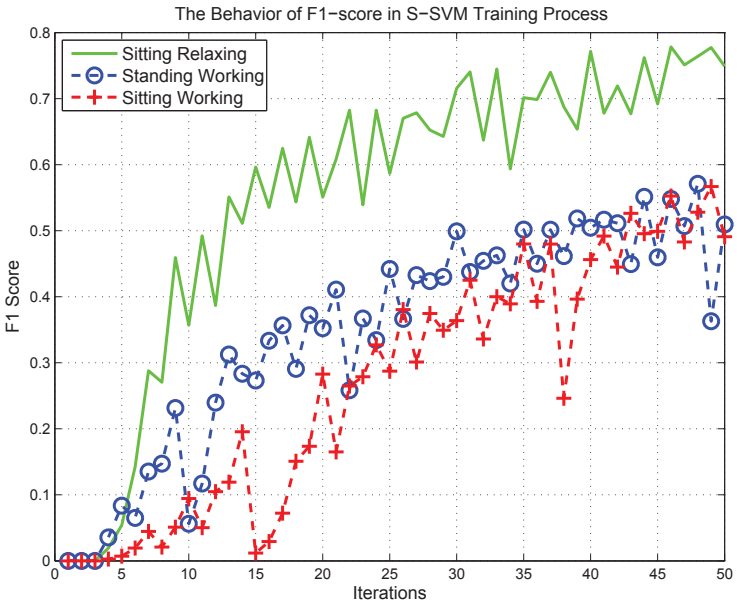


FIGURE 5.1: The F1-score improves at each iteration of the S-SVM training process

5.3 Training

Structured output SVM is a supervised training algorithm and therefore it needs to be trained before inferring the affordance-map in a new environment. First the feature set is calculated for each image of the training dataset, and algorithm 1 is used to calculate the weight vector, $\mathbf{w}$. Finally, the inference function given in Eq. 5.18 is used to predict the affordance map in a new 3D image.

The main advantage of using structured output SVM training for affordance detection is, parameter estimation can be directly optimized on performance measures like F1-score, which are not affected by class imbalance problem. Fig. 5.1 shows the behavior of the F1-score over each iteration of the training process. The F1-score of sitting-relaxing affordance quickly improves and the F1-scores of the other two affordances have relatively slow improvements. This indicates the complexity of the training process associated with each affordance type. Fig. 5.1 also suggests that Sitting-Relaxing affordance is easy to learn when compared to other two affordances as its F1-score improves quickly.

As each iteration of the proposed structured output SVM algorithm improves the F1 score, the trained parameters could easily over-fit to the training data. This could degrade the

performances on a previously unseen input image. However, over-fitting problem can be easily avoided by using a validation set. The optimum set of parameters for the structured SVM training is selected by recording the parameter set for each iteration of the training set and later testing them on a validation set. The parameter set that gives the best classification results on the validation set is used in inference.

5.4 Experimental Results

This section presents the experimental results of the affordance mapping process based on the structured SVM algorithm. The k-fold cross validation is carried out to test and report the results. The classification results of the Structured output SVM (S-SVM) are compared with the methods discussed in the Chapter 4, which previously reported good performances in the imbalanced dataset.

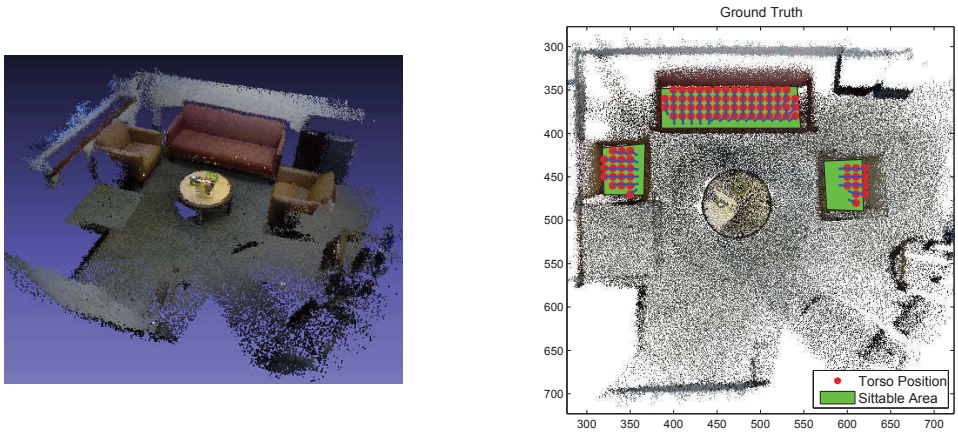
In the following sections, experimental results are presented using qualitative and quantitative methods. The same framework used to report the results of the Naive Bayes and SVM classifier in Chapter 3 is used for this purpose.

5.4.1 Qualitative Analysis

The same qualitative method described in Chapter 3 is used to compare the results between the S-SVM and the algorithms described in Chapter 4. It involves testing prediction results of the three affordances on a set of pre-selected 3D images. Both 2D affordance maps and 3D skeleton maps are used for this purpose.

5.4.1.1 Sitting-Relaxing Affordance

Fig. 5.2 shows the 3D view of a room and its ground truth skeleton locations of the room used to test the affordance 'Sitting-relaxing'. It is the same room used in experiments explained in Chapter 3. The predicted results of the S-SVM method are compared with the results of the zSVM method that previously recorded the highest F1-score for the Sitting-Relaxing affordance.



(a) 3D view of the living room

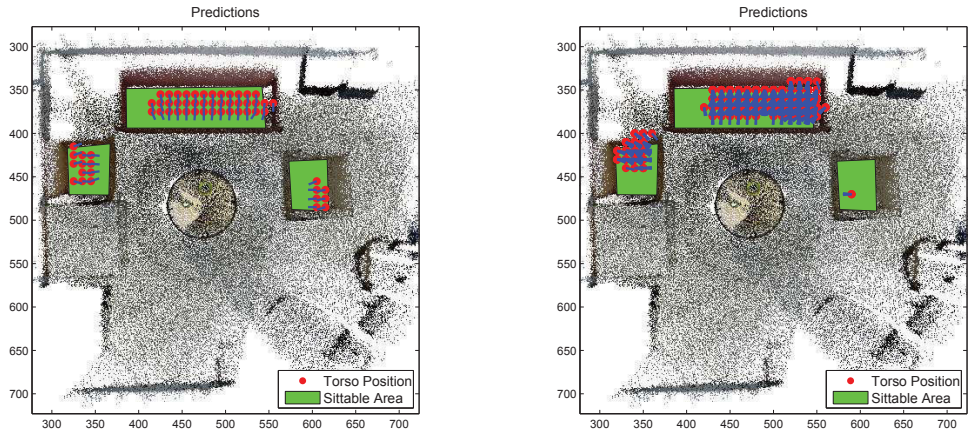
(b) Plan view with ground truth skeleton locations and their orientations

FIGURE 5.2: Test room for the Sitting-Relaxing Affordance. Best viewed in colour

Fig.5.3 shows the prediction results of the 'Sitting-Relaxing' affordance and Fig. 5.3(a) shows the prediction results of Structured Output SVM (S-SVM) and Fig. 5.3(b) shows the prediction results for the zSVM Model. Both these classifier have resulted in better predictions and the S-SVM has performed slightly better than the zSVM method. This is clearly visible from the prediction results on the armchair towards the right-side of the room. On this armchair zSVM has only predicted one skeleton location while S-SVM has predicted several skeletons with a slightly better representation of the ground-truth.

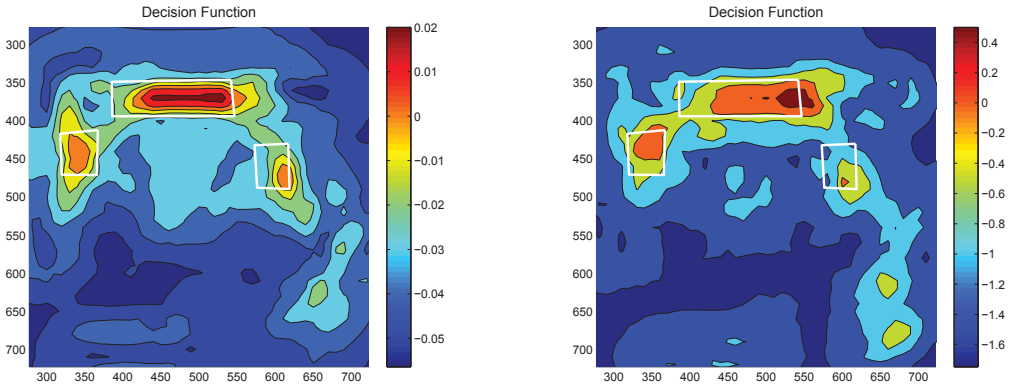
Fig. 5.4 shows the decision function for the S-SVM classifier and the zSVM classifier. It is clear from these results that the S-SVM decision function has better distribution of confidence values in positively predicted locations than the zSVM decision function. Specially, this can be observed near the long sofa towards the top-middle of the room. In this location, the S-SVM decision function has placed high confidence values throughout the sittable area of the sofa, whereas the zSVM decision function has shown a bias towards one of the corners of the sofa.

Fig. 5.5 shows the skeleton map for the affordance type Sitting-Relaxing. It is clear that the S-SVM classifier has predicted 3D skeletons more accurately than the z-SVM classifier. All the skeletons predicted by the S-SVM classifier are placed on the sofa set with correct orientation but the zSVM method has predicted few skeleton in locations other than the



(a) S-SVM Classifier (b) zSVM Classifier

FIGURE 5.3: Classification results of the Sitting-Relaxing Affordance



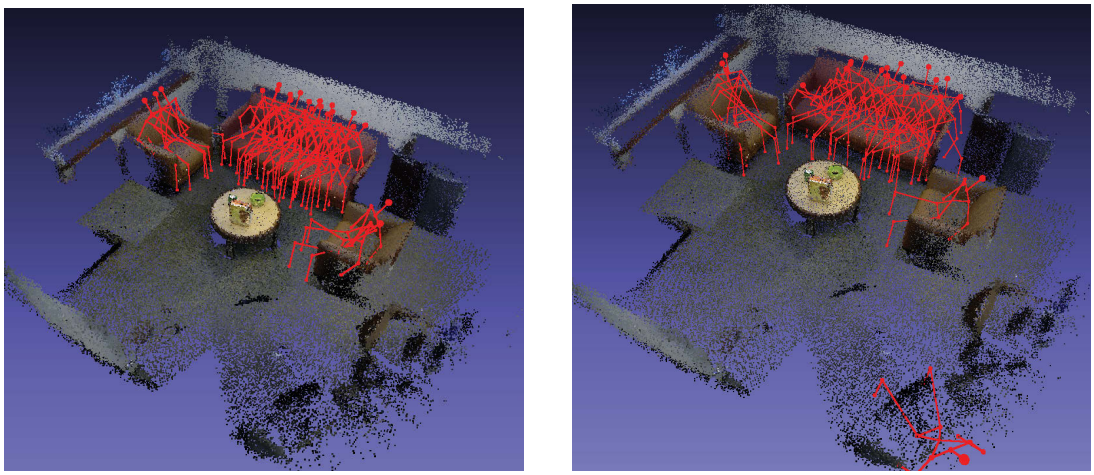
(a) S-SVM Classifier (b) zSVM Classifier

FIGURE 5.4: Decision Function values that indicate the confidence level of the predictions

sofa. Further few of the skeletons predicted by zSVM on the sofa do not seems to be correctly oriented.

5.4.1.2 Sitting-Working Affordance

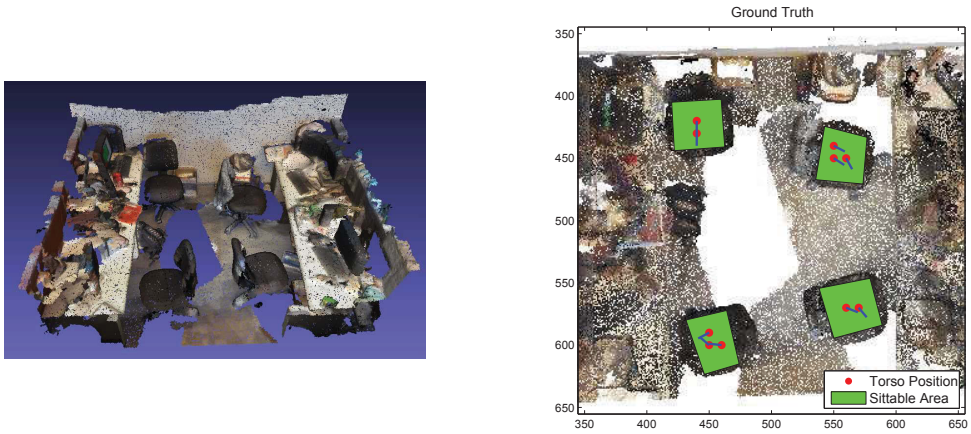
The 3D view and 2D plan view of the office space used to test the 'sitting-working' affordance is shown in Fig. 5.6. It has four office chairs as shown with green rectangle areas. For



(a) S-SVM Classifier (b) zSVM Classifier

FIGURE 5.5: 3D skeleton map for the Sitting-Relaxing affordance

the comparison of the results in the Sitting-Working affordance, the DCM-SVM method, which recorded the highest F1-score in the imbalance test, is used.

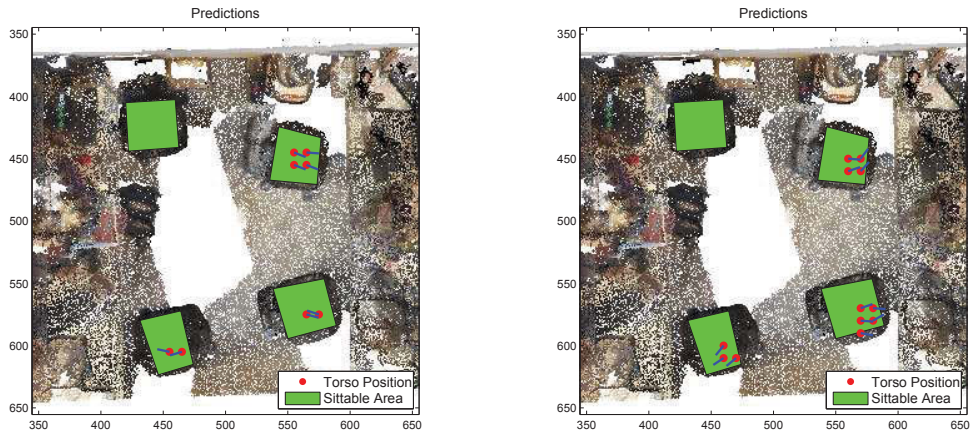


(a) 3D view of the office space (b) Plan view with ground truth skeleton locations and their orientations

FIGURE 5.6: Test room for the Sitting-Working Affordance. Best viewed in colour

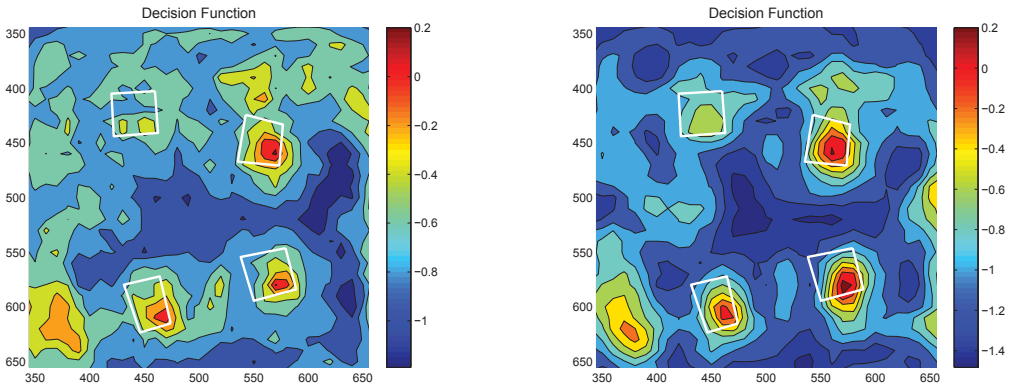
The prediction results of the 'sitting-working' affordance are shown in Fig. 5.7. It is clear that both the S-SVM classifier and D-SVM classifier have predicted similar classification results. The two classifier have accurately predicted 'sitting-working' affordance on three chairs out of four chairs in the room but have failed to identify 'sitting-working' affordance

on the chair towards the top-left of the room. Also, one of skeletons predicted by the S-SVM classifier on the chair at the bottom-right of the room has oriented towards the back side of the chair.



(a) S-SVM Classifier (b) DCM-SVM Classifier

FIGURE 5.7: Classification results of the Sitting-Working Affordance

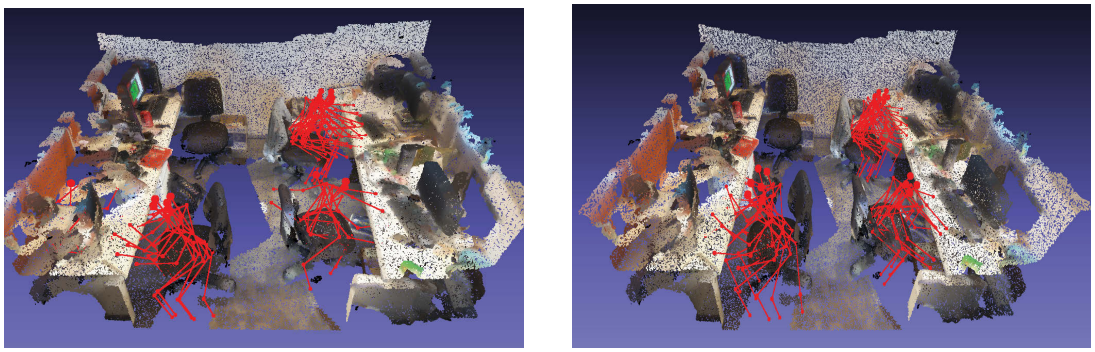


(a) S-SVM Classifier (b) DCM-SVM Classifier

FIGURE 5.8: Decision Function values that indicate the confidence level of the predictions

The Fig. 5.8 shows the decision function values for the 'sitting-working' affordance. Both S-SVM and DCM-SVM classifiers have predicted similar decision function values and have predicted high confidence values on locations where most of the chairs are located. Relatively, high confidence values can be observed in both classifiers decision functions towards

the bottom-left corner of the room. A careful examination reveals that those locations consists of geometric features very similar to a chair (Right angled horizontal and vertical surfaces).



(a) S-SVM Classifier (b) DCM-SVM Classifier

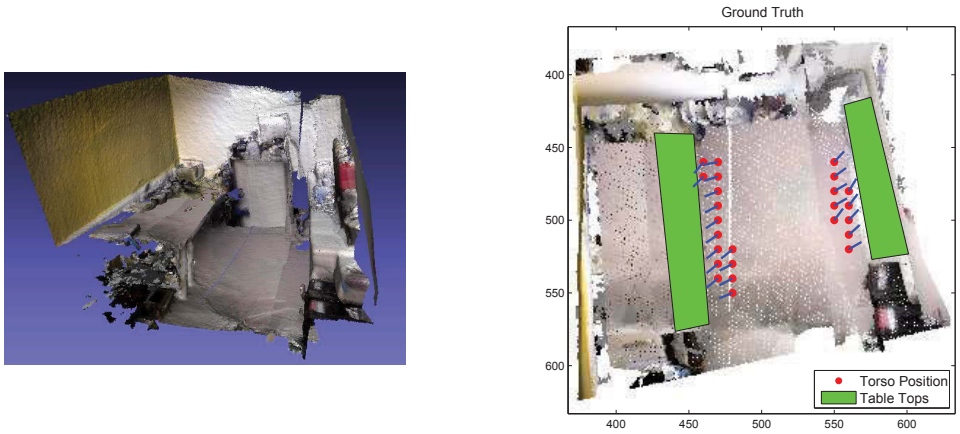
FIGURE 5.9: 3D skeleton map for the Sitting-Working affordance

Fig. 5.9 shows the skeleton map for the affordance type Sitting-Working. In this case the DCM-SVM classifier has marginally outperformed the S-SVM classifier. In particular S-SVM classifier has predicted a skeleton on the working desk towards the left side of the room and none of the classifiers have predicted any skeleton on the chair at the top-left corner of the room. This may be because the chair is oriented away from the nearby desk, and therefore it could be argued, that chair is not supporting the Sitting-working affordance type. Overall, both classifiers have predicted a very similar skeleton map.

5.4.1.3 Standing-Working Affordance

Fig. 5.10 shows the room used to analyze the performances of the 'standing-working affordance'. It is the same room used in Chapter 3 experiments and green color rectangle areas indicate workbenches of the lab space. According to the imbalance test results discussed in Chapter 4, the focused resampling SVM (Focused-SVM) method recorded the highest F1-score in the Standing-Working affordance. Therefore the results of the S-SVM method are compared with the Focus-SVM method.

The prediction results of the S-SVM classifier and Focused-SVM classifiers are shown in Fig. 5.11. It is clear from this figure that the S-SVM classifier has predicted better



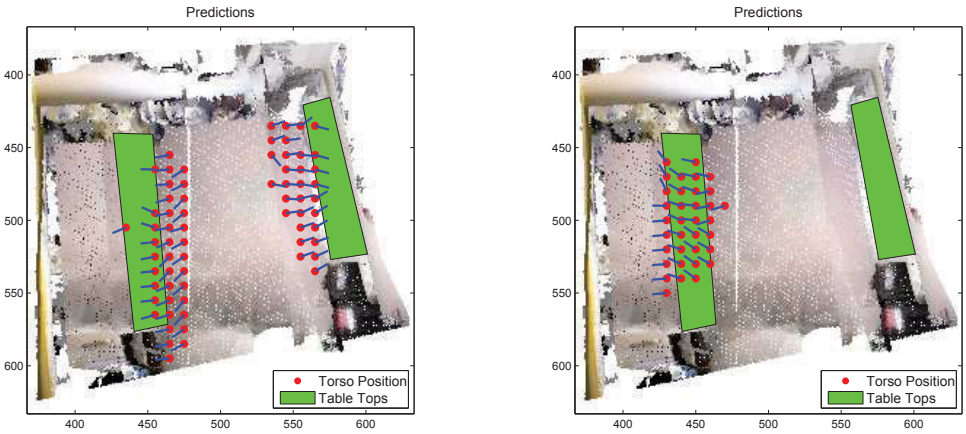
4

(a) 3D view of the work shop

(b) Plan view with ground truth skeleton locations and their orientations

FIGURE 5.10: Test room for the Standing-Working Affordance. Best viewed in colour

results than the Focused-SVM classifier. The Focused-SVM classifier has failed to predict any skeleton model at the workbench towards the right side of room. Although it has predicted some skeletons at the other workbench towards the right side of the room, most of them are located inside the workbench. In contrast, the S-SVM classifier has predicted skeletons at both workbenches and they are nicely oriented towards the working areas of the benches.

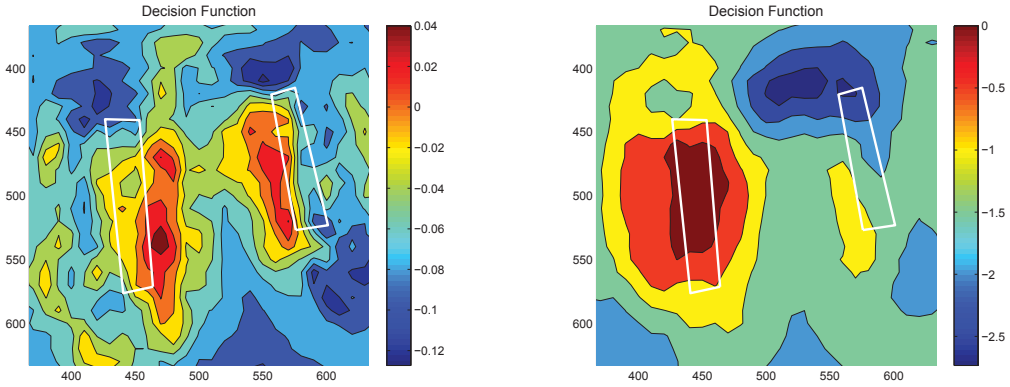


(a) S-SVM Classifier

(b) Focused-SVM Classifier

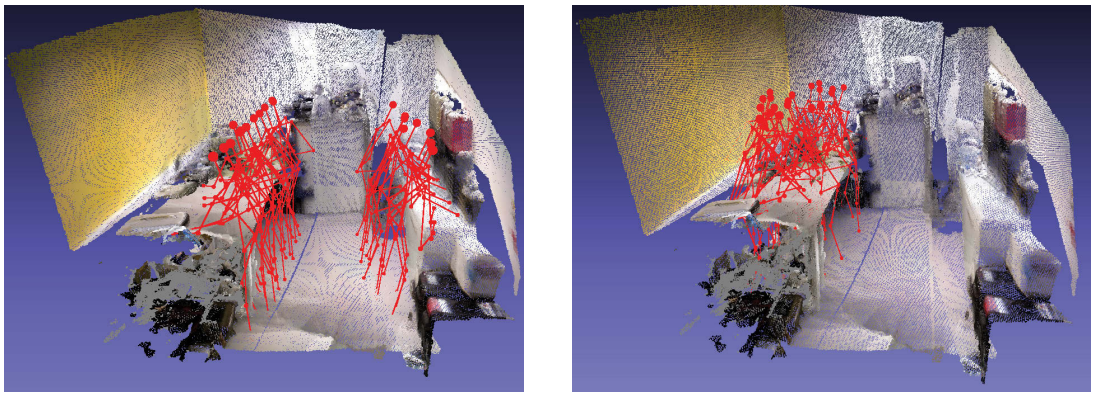
FIGURE 5.11: Classification results of the Standing-Working Affordance

Fig. 5.12 shows the values of the decision function for the affordance type Standing-Working. As indicated by the prediction results, the S-SVM classifier has placed high confidence values near the two workbenches whereas the Focused-SVM classifier has placed a number of high confidence values inside the workbench towards the left of the room. Therefore, the S-SVM classifier has predicted much better localized results than the Focused-SVM classifier.



(a) S-SVM Classifier (b) Focused-SVM Classifier

FIGURE 5.12: Decision Function values that indicate the confidence level of the predictions



(a) S-SVM Classifier (b) Focused-SVM Classifier

FIGURE 5.13: 3D skeleton map for the Standing-Working affordance

Fig. 5.13 shows the skeleton map for the affordance type Standing-Working. In this example, the S-SVM method has predicted a very accurate skeleton map with skeletons

located near the working benches with correct orientations. However, the Focused-SVM method has only predicted skeletons around one of the workbenches and some of the skeletons are located inside the workbench. In this case, S-SVM method has outperformed the Focus-SVM method.

5.4.2 Quantitative Analysis

Table 5.2 shows the comparison of average performance measures of k-fold cross validation between S-SVM and other methods discussed in Chapter 4. For the methods discussed in Chapter 4, Table 5.2 shows the average precision and recall values of the best F1-score recorded in class imbalance test for each affordance type. It is clear from this data the S-SVM method has outperformed other methods. The S-SVM method has recoded the highest F1-scores in the affordance types Sitting-Relaxing and Standing-Working by a considerable margin to the next best F1-score of each category. It has recorded the third highest F1-score in Sitting-Working affordance with only -0.04 % difference to the highest recorded F1-score of that category. Overall, the S-SVM method has shown consistent performance in all three affordance types. This is because, unlike other methods, S-SVM has been trained directly optimizing on F1-score. Therefore it is minimally affected by the class imbalance problem.

5.5 Discussions and Limitations

As discussed in previous chapters affordance learning in a highly imbalanced dataset could be challenging. As a solution to this probelm, this chapter showed how affordance learning can be formulated as a structured output learning problem. The quantitative and qualitative results of the experiments showed that the S-SVM method outperformed other SVM learning methods in Sitting-Relaxing and Standing-Working affordance types. In the Sitting-Working affordance type, DCM-SVM and Random sampling methods outperformed the S-SVM method by a small margin. However, overall S-SVM has outperformed the other four classifiers.

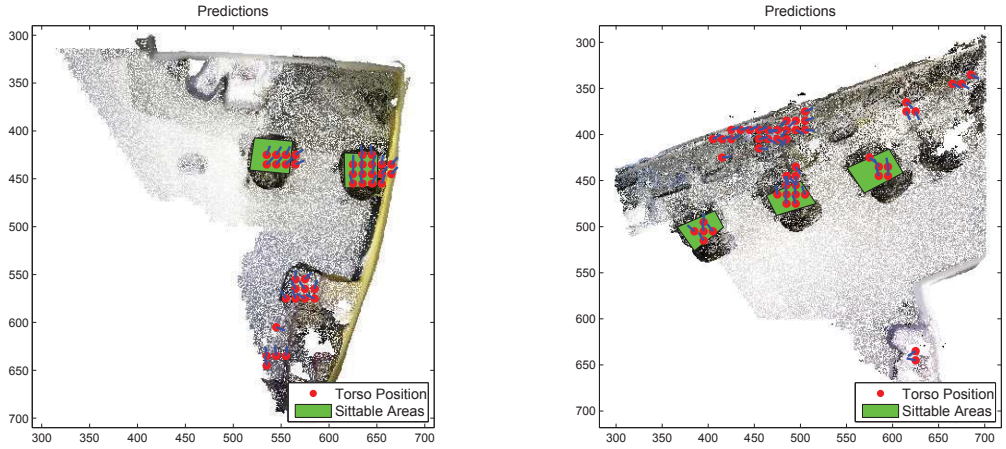
TABLE 5.2: Comparison of performance measures in the K-fold cross validation test. Structured SVM and Other Methods.

Affordance Type	**SVM Algorithm**	**Precision**	**Recall**	**F1-Score**
Sitting- Relaxing	Random Sampling	0.36	0.33	0.35
	Focused Sampling	0.32	0.37	0.35
	Z- SVM	0.51	0.35	0.35
	Different Cost Model	0.28	0.44	0.34
	Structured SVM	0.65	0.80	**0.72**
Standing-Working	Random Sampling	0.09	0.49	0.13
	Focused Sampling	0.17	0.30	0.22
	Z- SVM	0.12	0.31	0.15
	Different Cost Model	0.10	0.42	0.14
	Structured SVM	0.28	0.81	**0.42**
Sitting-Working	Random Sampling	0.55	0.52	0.50
	Focused Sampling	0.34	0.51	0.38
	Z- SVM	0.27	0.84	0.37
	Different Cost Model	0.50	0.55	**0.52**
	Structured SVM	0.40	0.77	0.48

The success of the S-SVM method is mainly due to its ability to learn separating hyper plane of two classes by minimizing F1-score of the training data set. Therefore, unlike the standard SVM method which is prone to generate sub-optimum results when trained in imbalanced datasets, the S-SVM method can be efficiently trained in all class imbalances. On the other hand, the formulation of the S-SVM algorithm is well defined and its performances do not depend on any rule of thumb parameters like the z in the zSVM method or the class cost values in DCM-SVM.

In addition, the S-SVM method provides additional benefits when used for affordance learning. First, S-SVM does not use the kernel method for feature mapping but outperforms the standard SVM and its variants that uses kernel methods. In fact, S-SVM only requires the same number of calculation as the standard SVM method that doesn't use kernels in the inference stage. Therefore, inferring the affordance-map in a new room could be done efficiently with a lesser time than the SVM method that uses kernels. This makes S-SVM more appropriate for real-time robotics applications. On the other hand, the formulation of S-SVM allows the input of each 3D scene sequentially into its optimization equations. Therefore, the S-SVM algorithm can be implemented on a standard

computer as a batch processing method thus considerably reducing the amount of data memory required for learning. This makes S-SVM learning in a large dataset with many 3D images very efficient and practically viable.



(a) room 5 (b) room 6

FIGURE 5.14: False positive prediction of the affordance type Sitting-Working

Despite these advantages, the proposed S-SVM method tends to generate a few false positive predictions in some 3D scenes. This is predominantly observed in the Sitting-Working affordance and it is also evident by high precision values recorded in Table 5.2. A few such examples of false positive predictions are shown in the Fig. 5.14. In this figure, all positive predictions outside sittable areas are false positive. Careful observation of locations of these false positive predictions revealed that most of those locations consist of geometric features (vertical and horizontal surfaces) which are similar to a chair. Therefore, one way to reduce the number of false positives would be to add more features that could discriminate against these wrongly predicted locations. However, this approach was not pursued any further as advanced feature selection is beyond the scope of this thesis.

Chapter 6

Active Visual Object Search using Affordance-map

6.1 Introduction

This chapter introduces an application of the affordance-map. There is a strong relationship between the objects and humans as humans generally arrange objects around them to support their activities. These human-object relationships could be used to build an efficient object search strategy. Therefore, this chapter explains how human context embedded in an affordance-map could be used to develop an efficient object search strategy.

6.2 Human Context for Object Search

Ability to recognize objects plays a major role in a robot's understanding of its environment. To execute various tasks such as pick and carry or object manipulation requires effective interaction with objects. Therefore, a service robot that operates in an indoor environment is required to localize and map objects in its operating environment. This is challenging due to several reasons. First, the operating environment of an indoor robot could extend beyond a robot's sensory range. Therefore the robot should have an understanding of where to look for objects when they are located beyond its perception range.

Secondly, real time object recognition is still largely an open problem. This makes object search and recognition a challenging task even when objects are visible to the robot's sensors.

Active object search involves executing a series of sensing actions in order to bring the object to the field of view of the sensor. When vision sensors like cameras are involved, this is called an 'active visual object search'. In order to increase object detection accuracy, the robot should move in a path that maximizes the object detection probability. Solving this problem is far from trivial because factors such as occlusions and poor illumination affect the object recognition capabilities. On the other hand, 3D objects can be viewed from a number of different view points. Therefore, often object recognition techniques involve training models for multiple view points. However, even such a greedy approach may often fail for most of the unsymmetrical objects, if the object is viewed from a previously untrained view point.

However, complexities associated with object search can be minimised by understanding the relationships between objects and humans. This is because most of the objects play a major role in humans' perception of their environment [48]. Further, humans arrange objects in their environment in a specific order to cater for their requirements and activities [6]. For example a 'computer monitor' is placed on a tabletop so it can be easily seen, and a 'keyboard' is placed within arms reach of the human sitting beside the table. A 'Mouse' is placed near the keyboard in a place that is convenient to handle. Therefore, these strong relationships between objects and humans can be used to localize probable locations of a given object.

The proposed approach for active object search is based on the concept of affordance-map. This brings the human context to the active object search which has not previously explored in robotic object search research. The bulk of the previous works have relied on the fact that the object is already located within the fields of view of the robot's sensors [49]. These aproaches have used object features such as color to guide the robot towards the object. However, it is unclear how the same approach can be used, if the object is located outside the sensory range of the robot.

Some other researchers have used object-object relations to search objects in large search areas [50, 51]. Common approach is to model object-object relationships using probabilistic graphical models such as Markov Random Fields (MRF). However for this to work, at least a few objects need to be recognized before the object search step or strong prior assumptions have to be made about the location of objects. On the other hand, these approaches do not model the human context in the environment, which has a strong relationship with objects that are being used in the environment.

In order to model human context, humans need to be observed in the environment. However, observing real humans could be time consuming, and in most scenarios, robots are required to operate in environments where humans are not present. This is a situation where an affordance-map becomes beneficial, because an affordance-map embodies probable affordances in an environment.

Although many affordances are possible in a given environment, very few are practically probable. For example, in an office environment most frequently observed affordance is sitting and working beside an office desk. An office desk supports this action and other objects such as the monitor, keyboard, mouse and the telephone are arranged around the sitting human. Therefore, by identifying the locations of the 3D map that support the affordance 'sitting and working', most probable locations of any object can be easily modelled. Each grid cell of the affordance-map consists of orientation information of the human skeleton model. This information is used to model the relationship between the human, object and the robot's position which gives the best view of the object that is being searched. Therefore the proposed method for active object search only requires a small number of training samples that are taken from a limited number of view points. In other terms, our method looks at the objects only from similar viewpoints that were used to train the object detector.

6.2.1 Affordance-map for Object search

The main challenges that need to be solved in any robotic object search algorithm is the estimation of the camera pose that gives the best view of the searched object. This problem can be effectively addressed by using human skeleton information embedded in

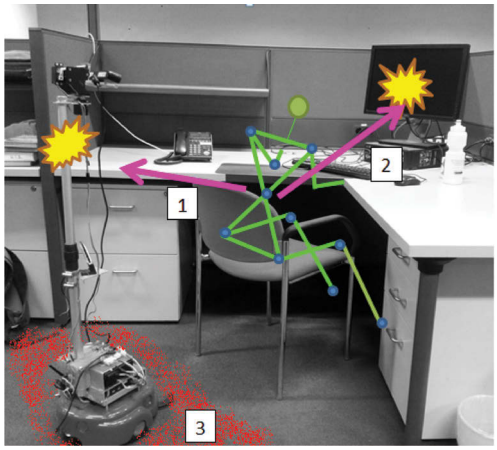


FIGURE 6.1: Affordance-Map models the relationship between virtual human skeleton model and the 3D environment. 1.Robot- Human relationship 2. Object -Human relationship 3. Probable robot's locations that give the best view of the object

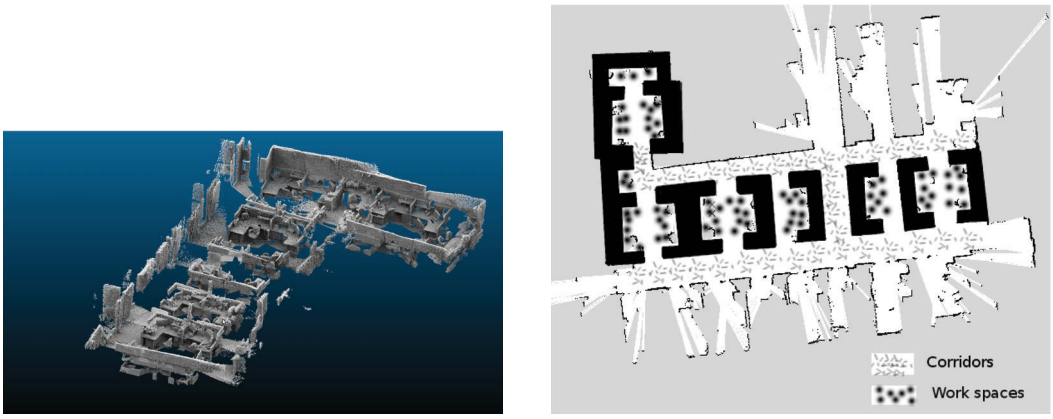
the affordance-map. For example, consider the scenario shown in Fig. 6.1. Suppose that the task is to search for computer monitors, and the affordance-map is already available for the scene. The affordance-map has predicted a human skeleton in-front of the monitor and therefore the human-object relationship is already modeled implicitly in the affordance-map. Also let us assume that the camera is fixed to the robot.

Now if, in human-robot relationships, the relative poses of the robot from the sitting human skeleton that give the best views of the object are known, then the skeleton-map from the affordance-map can be used to map all robot positions in the global map that give the best views of the object. Therefore, human-object relationships and human-robot relationships simplify the complexities associated with robotic active object search.

6.2.1.1 Human-Object Relationship

The objective of this research is to find monitors in a large office space using a mobile robot. Therefore, the first step is to obtain the affordance-map of the environment that models the human-object relationships. The sitting-working affordance type is chosen to build the affordance-map. Most of the computer chairs found in the office space had rotating bases and most of them were not oriented towards the work benches. Therefore, in order to minimize any adverse effects, the ground truth training examples included chairs as well as locations near the work benches with skeletons oriented towards the workbenches.

The 3D point cloud map of the environment was built using a state of the art RGDB SLAM (Simultaneous Localization and Mapping) algorithm [30] and a depth camera mounted on a mobile robot. Fig. 6.2(a) shows the 3D point cloud and Fig. 6.2(b) shows a map generated by a horizontally mounted 2D laser range finder data. Work bench locations are manually marked on the map.



(a) 3D view of the office space (b) 2D Laser Map

FIGURE 6.2: Test room for active object search experiments

The obtained 2D and 3D representations of the affordance map is shown in Fig. 6.4 and Fig. 6.3. High 'workable' affordances are recorded near work benches as can be seen from the Fig. 6.3. A 3D representation of the affordance map with skeletons is shown in Fig. 6.4. The proposed algorithm has placed most of the skeletons near the workbenches. More importantly, the learned affordances are more or less realistic to real human behaviours in the environment, as can be seen from the 3D affordance map in Fig. 6.4.

6.2.1.2 Human-robot Relationship

The affordance map which is built for the office environment, models the relationship between the environment and the 'working' human pose. Therefore, the position of the robot can be inferred by modelling the relationships between the object, human and the robot. More specifically once the human pose position is given, we can infer the best position and the viewing angle of the robot that give a clear view of the object to be recognised.

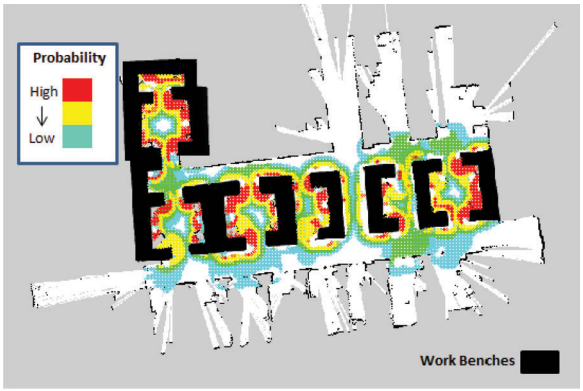


FIGURE 6.3: 2D Affordance Likelihood Map

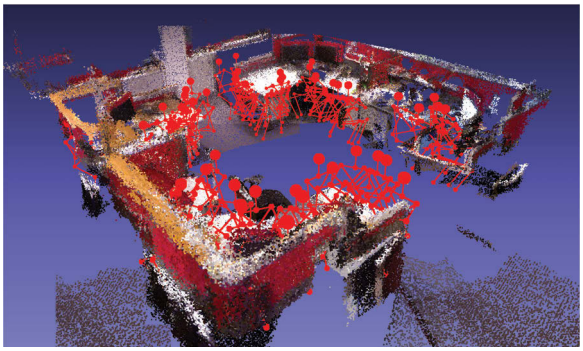


FIGURE 6.4: 3D Skeleton Map

The robot position is modelled as a multivariate normal distribution relative to the human skeleton model using (6.1). Here μ_s and Σ_s are the mean and covariances of position and orientation of robot in human skeleton co-ordinate system.

$$R_s \sim \mathcal{N}(\mu_s, \Sigma_s) \tag{6.1}$$

This probability distribution can be converted to world coordinate system for a skeleton at $\mathbf{x}_k = (x_k, y_k, z_k, \theta_k)$ by,

$$R_k \sim \mathcal{N}(\mathbf{x}_k + B\mu_s, B\Sigma_s B^T) \tag{6.2}$$

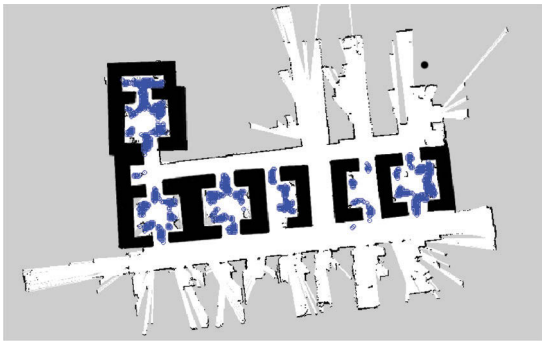


FIGURE 6.5: Positions of the Robot that give the best view of the object

where B is the rotation matrix given by (6.3)

$$B = \begin{vmatrix} cos(\theta_k) & -sin(\theta_k) & 0 & 0 \\ sin(\theta_k) & cos(\theta_k) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{6.3}$$

Finally, the likelihood of a robot at location $\mathbf{y}_j = (x_j, y_j, z_j, \theta_j)$ which can see an object is given by

$$P(z_j|\mathbf{y}_j) \propto \sum_{k=1}^{n} P(a_k|\mathbf{x}_k).P(\mathbf{y}_j|\eta_k) \tag{6.4}$$

where η_k are the parameters of the probability distribution given by (6.2), n is the number of skeletons in the map and $P(a_k|\mathbf{x}_k)$ is the affordance likelihood of the location $\mathbf{x}_k$, which is calculated by the affordance-map. Note that, only closeby skeletons with high affordance likelihood contribute to (6.4). By using (6.4), the likelihood of finding an object by a robot at location Y_j can be calculated for all grid locations of the map.

6.2.2 Experiments- Active Object Search

The affordance-map that was built previously for the office space is then used to experimentally verify the effectiveness of the active object search algorithm. A 'computer monitor' was selected as the object to be searched in this experiment as it could be observed in several locations in the office space. Fig. 6.5 and Fig. 6.6 show position information of

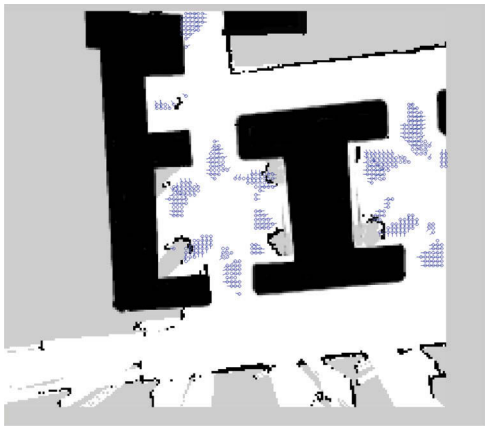


FIGURE 6.6: Scaled view of the robot's position and orientation for object search

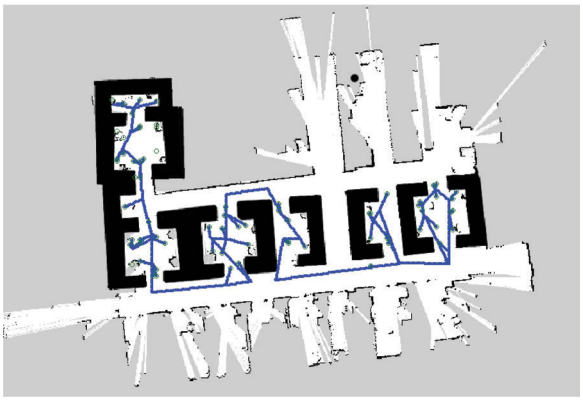


FIGURE 6.7: Path planning for active object search

the robot that gives the best view of the object 'computer monitor'. These positions are calculated using object, human and robot relationships explained in Section 6.2.1.2 and locations with highest likelihood values calculated using (6.4) are selected as locations that give the best views of the searched object. It is clear from these results, in most of the occasions the robot has positioned itself near work benches pointing the camera towards benches so it can obtain clear views of "computer monitors". Although the robot never observed a real human in the environment, the affordance-map has allowed the robot to learn human context as well as the probable locations for the searched object.

Fig. 6.7 shows the path planning results of the robot for object search. It uses the Probabilistic Road Map (PRM) based path planner [52]. To move the robot across all possible search locations efficiently, we formulated the problem as a 'Travelling Salesman problem'. Since finding an exact solution for the Travelling salesman problem is NP hard,

TABLE 6.1: Object detection results summary

Performance Measure	Value
Number of Monitors in the environment	33
Total Number of Snaps taken	49
Number of Snaps with Monitors in the image	43
True Positive Detections	37
False positive Detections	14
True Negative Detections	6
False Negative Detections	8
Precision	0.72
Recall	0.87

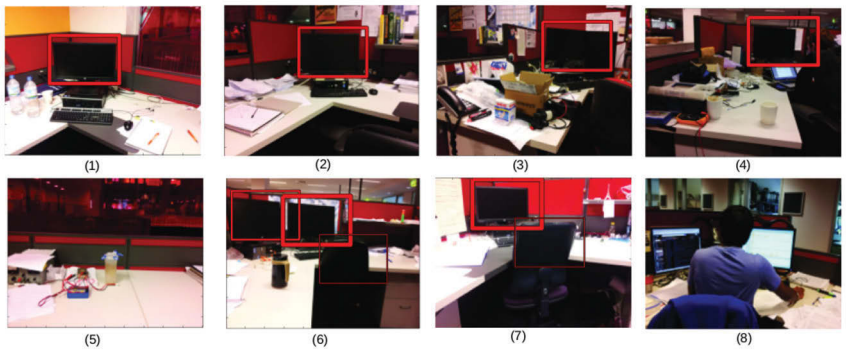


FIGURE 6.8: Samples from object search and detection experiments. (1-4) True Positive, (5) True Negative, (6-7) False positive, (8) False Negative

Nearest Neighbour Algorithm [53] is used to estimate the possible path of the robot. As depicted in Fig. 6.7, calculated path of the robot covers all possible locations where "computer monitors" can be found.

Once the robot reaches the predicted location, it captures RGB images of the possible areas that the computer monitors can be found. Then a simple object classifier based on boosting [54] is used to recognize monitors in the scene. Samples from the object detection experiments are shown in Fig. 6.8.

The object recognition results are summarized in Table 6.1. According to the test results, only 49 snaps are taken across the entire office environment and 43 of those images contain one or more monitors. This shows the proposed approach can effectively infer the possible locations of monitors in a large environment. On the other hand, object detector performed well giving acceptable recall and precision values (recall of 0.8 and precision of 0.72). Note here that none of the monitors present in the environment are used to train the object

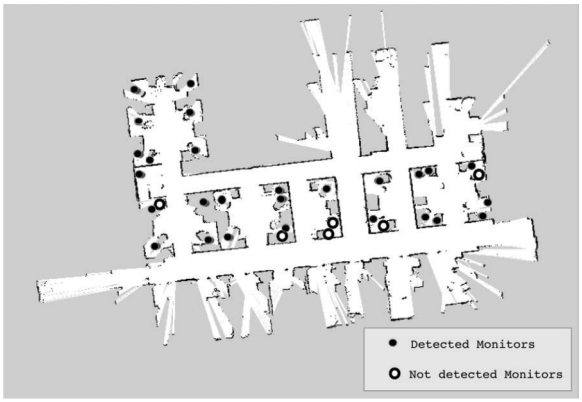


FIGURE 6.9: Object detection results. The figure shows detected and undetected monitors

detector. Although the object detector is neither scale invariant nor view point invariant, the algorithm was able to detect 27 monitors out of 33 monitors present in the given search area. This shows the robot is able to position its camera correctly towards the object.

It is clear from these results that the proposed algorithm can solve the object search problem efficiently by modeling the human context in the environment through the affordance-map.

6.3 Conclusions

This chapter introduced a novel active object search approach centered around human context in an indoor environment. It showed how the human context information embedded in the affordance-map can be effectively used for active visual object search. The object search is carried out by a naive algorithm but leads to high detection accuracies. This is due to the correct pose estimation based on the object-human-robot relationship model. Experiments carried out in a large office environment demonstrated that the proposed approach can efficiently search for given objects using the human context information provided by the affordance-map.

Chapter 7

Affordance-map for context aware path planning

7.1 Introduction

This chapter introduces another application of the affordance-map, i.e., how hidden human context obtained using the affordance-map can be used effectively for robotic context aware path planning. The affordance-map provides vital information for a robot that works alongside humans. Firstly, it provides possible locations of humans with their affordance likelihood. A robot can use this information to detect and track humans. More importantly, the robot can use this human context information to plan global paths that minimize distractions to humans. Secondly affordance-likelihood values can be used to to assign social cost values for each grid location of the map. These social cost values together with A* algorithm [55] are used in this chapter to develop a global path planning strategy that can be used by a mobile robot to avoid human work spaces while executing its task. It also presents some experimental results done on a real office environment to show the applicability of the proposed system.

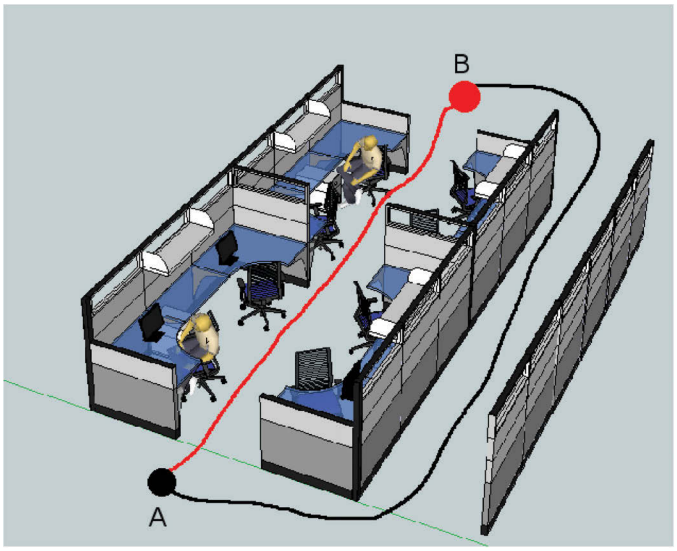


FIGURE 7.1: A robot with proper context-awareness will select a path along corridors rather than the shortest path which is through the working area

7.2 Human Context in Path Planning

In robotics, context awareness refers to the ability of a robot to sense and react according to the environment. Particularly, understanding human context is paramount for a robot that works alongside humans. Therefore, human context often involves recognizing probable human activities in a given environment. However, even recognizing a simple human activity is considered to be non-trivial due to large intra-activity variations associated with human activities [4].

In order to react according to the context of the environment, a mobile robot is often required to alter its path.Therefore, context-aware path planning is a fundamental ability that every robot should acquire in order to navigate in a human work space. However, most traditional path planning algorithms are based on two major factors, obstacle avoidance and minimizing cost associated with robot's movements. In other words, these robots try to move from the start position to the goal location through the shortest possible path while avoiding obstacles on its way. Although this path planning strategy is more suitable for an industrial setting, such an approach is often perceived as inappropriate for a robot that works alongside humans, as human factors are not considered in path planning.

The need for context-aware path planning can be explained by the scenario shown in Fig. 7.1. In this example, a robot is required to move from point A to B. A robot that uses a traditional path planning strategy would select the shortest path between A and B, which is going along the human work space. Therefore in this case, a robot that navigates on the shortest path may distract human workers. However, a robot with context-aware path planning capability would choose an alternative path along the corridor minimizing obstruction to human activities.

For this research, context-aware path planning is defined as the path that minimizes the interferences caused to people working in an office environment. The Context-aware path planning strategy involves avoiding work spaces of humans and planning paths through the human work spaces if and only if the human presence is not detected. The robot can use an affordance map for global path planning as it provides location information of human work spaces. In this research this path is defined as 'Global Path with Minimal Distractions', (GPMD). However, if this path is too long then it is desirable for a robot to seek an alternative path that goes through the nearest unoccupied workspace.

7.3 Related Works

There are many publications that explored various robotic path planning strategies. However, only a few researchers have previously considered global path planning with appropriate context awareness.

In [3] researchers developed social aware path planning strategy using a robot that learns human motion patterns based on sampled Hidden Markov Models. Then they utilized these models for path planning based on a Probabilistic Roadmap. However their approach needs to continuously observe real humans in the environment before it can be used for social aware path planning. Therefore, such a robot would require a considerable amount of time to learn human context in a new environment. On the other hand, human motion patterns could be affected by the presence of the robot itself.

In an attempt to build a context-aware path planning strategy, some other researchers have used an off-line learning procedures to obtain motion models of individual people in

an office using HMM [56]. Once a person is identified, a model is used to predict that persons future position in order to perform the 'give way' action. Thus, the model is used to improve local replanning and does not influence global planning. However, such a behaviour could lead to more complex behaviours and a robot might interfere with humans when deployed in crowded environments.

In [57] researchers tried to build a socially aware navigation strategy using objective criteria such as travel time or path length as well as subjective criteria such as social comfort. As opposed to model-based approaches they posed the problem as an unsupervised learning problem. They learnt a set of dynamic motion models by observing relative motion behaviour of humans found in publicly available surveillance data sets. Again this strategy requires the human to be present and detected in the environment. Further, they have only focused on human-like collision avoidance rather than global path planning.

7.4 Affordance-Map for Context-Aware Path Planning

Most of the shortcomings associated with current context-aware robotic path planning algorithms could be avoided by using the human context information embedded in the affordance-map. Firstly, an affordance-map does not require the observation of real humans in the environment, yet it can infer human behaviors of the environment by looking at geometric features in the environment. Therefore, robots would not be required to detect and observe real humans for a long time to learn the human context of the environment. Secondly, a robot can plan its paths globally rather than reacting to humans detected on its way to the destination. If needed the robot can predict possible locations of humans in the environment and purposely avoid them to minimize distractions . Thirdly, if human detection is needed, the robot can look only at places where the affordance likelihood is high thus reducing the total search space considerably.

The following sections explain how an affordance map can be used for context-aware path planning in an office environment. For this research, context-aware path planning is defined as the path that minimizes the interferences caused to people working in an office cell. The Context-aware path planning strategy involves avoiding work spaces of humans

and planning paths through the humans work spaces if and only if the human presence is not detected. The robot can use the affordance map for global path planning as it provides location information of human work spaces. This path is denoted as 'Global Path with Minimal Distractions', (GPMD). However, if this path is too long then the robot needs to seek an alternative path that goes through the nearest unoccupied workspace.

The first step of the context-aware path planning strategy involves obtaining the affordance-map of the robot's operating environment. The same office space used in Chapter 5 for active object search is used for the context aware path planning experiments. Sitting-working affordance type is used to build the affordance map as it is observed much more frequently in office spaces than other affordances.

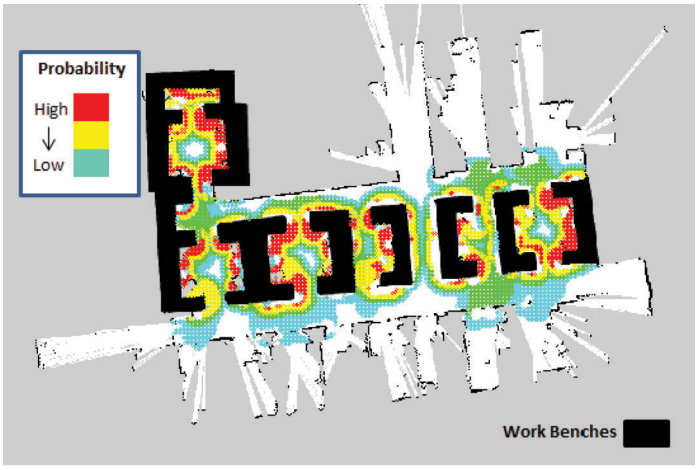
The obtained 2D and 3D representations of the affordance map are shown in Fig. 7.2(a) and Fig. 7.2(b). High 'workable' affordances are recorded near work benches as can be seen from Fig. 7.2(a). A 3D representation of the affordance map with skeletons is shown in Fig. 7.2(b). The proposed algorithm has placed most of the skeletons near the workbenches. In the next section, affordance likehoods obtained through this affordance-map are used to assign social cost values to plan paths that minimize distraction to working humans.

7.4.1 A* for Path Planning

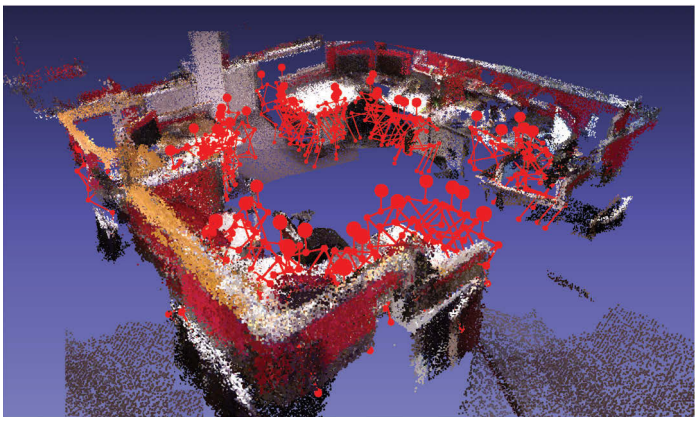
The A* algorithm has been in use for robotic path planning for many years [3, 55]. It utilizes best-first search strategy to search for the least cost path that lies in the D-dimensional collision free space of the robot.

The algorithm searches for a path using a priority queue, where each node x in the path is sorted according to the cost function $f(x)$. Therefore highest priority is given to the nodes with least cost. Generally $f(x)$ is the sum of two main components, $g(x)$ which is the shortest collision free path from start to goal by Euclidean distance and $h(x)$, which is used as a heuristic estimate of the length of the path.

In order to plan a minimum distraction path, a social cost value $g_S(x)$ can be incorporated to the total cost function such that A* algorithm will seek a collision free path while minimizing interference to humans. Depending on the affordance model selected social cost



(a) 2D Affordance Likelihood Map



(b) 3D Skeleton Map

FIGURE 7.2: The learned affordance map for the office space. High affordance likelihoods can be observed near workbenches

$g_S(x)$ for each grid cell can be assigned according to the affordance likelihood. For example, for an office environment 'sitting and working' affordance can be selected as the affordance model and high social cost values can be assigned to the areas where affordance likelihood is high. In such cases social cost is given by (7.1) where $P(A_k|\mathbf{x}_k)$ is the affordance likelihood of the grid location $\mathbf{x}_k$.

$$g_S(x) \propto P(A_k|\mathbf{x}_k) \tag{7.1}$$

Consequently, f(x) in the standard algorithm can be replaced by the cost function $F(x)$. In some scenarios it is appropriate to have additional weight term w for social cost function

$g_S(x)$ in (7.1) to enable the robot to choose between shortest path and alternative minimal distraction path. This can be incorporated as,

$$F(x) = g(x) + w * g_S(x) + h(x) \tag{7.2}$$

If w is set to 1, the robot prefers the shortest path whilst any number between 0 and 1 denotes a combination of the shortest path and a context-aware path.

7.4.2 Detecting Human Presence

Detecting human presence is vital for a robot that does context-aware path planning because the human context defines the role of the robot in most scenarios. Although a number of human detecting algorithms are available in the literature, detecting humans while the robot is in motion is considered to be non-trivial [3]. One reason for this challenge is the absence of a priori knowledge about the presence of human poses and their best viewing directions. On the other hand, how to react to the human presence depends on the context of the location. For instance, if a human is detected in a corridor, the robots can simply treat him/her as an obstacle and plan to avoid the obstacle (could be a dynamic obstacle). However, if the human presence is detected in an office space, then the robot

needs to plan a path to minimize interference to humans.

Algorithm 3: Context-Aware Path Planning

Data: Affordance map

Result: Context-Aware Path

1 $PS \leftarrow$ Calculate the shortest Path;
2 $PA \leftarrow$ Calculate the Global Path with Minimal Distractions (GPMD) using Afordance-map;
3 $T_p \leftarrow$ Path Threshold;
4 **if** $(len(PA) - len(PS))/len(PS) \leq T_p$ **then**
5 Follow PA
6 **else**
7 **while** *Detect people* ***OR*** *Goal* **do**
8 Follow PS
9 **end**
10 Assign Affordance cost values to all locations within the sensor range and update the cost map;
11 $PS_{new} \leftarrow$ Calculate the new shortest path ;
12 $PS \leftarrow PS_{new}$;
13 **GoTo** Line 7;
14 **end**

However, these problems can be effectively addressed by using the information embedded in the affordance-map. Firstly, it contains information about human activities like 'sitting and working'. Secondly, the affordance-map contains information about the locations and poses of possible human workers. This can be used for an active human search, which will eventually lead to a higher human detection rate.

7.4.3 Context-aware Path Planning Algorithm

Algorithm 3 summarises the major steps involved with context aware path planning. First A* algorithm is used with affordance map to calculate shortest path, PS and GPMD path PA as explained in the Section 4.2. Path threshold T_p defines when to use GPMD. The

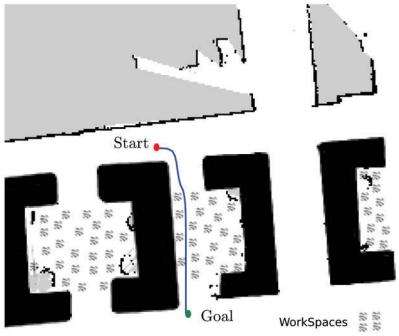
path threshold can be set according to the urgency of the task that is being carried out by the robot using a high level planner. If the difference of travel distance between these two alternative paths is small, then the robot chooses the GPMD path which is reasonable. If the difference of travel distance is large, the robot starts to move along the shortest path while looking for humans. If a human is detected, then the robot alters its path by recalculating a new path to minimize the interference caused to the human.

7.5 Experiments

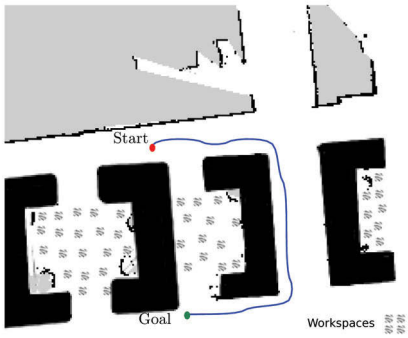
This section explains how affordance map is used for context-aware path planing. As the robot is required to avoid 'working spaces' of humans in the office environment, we used A* algorithm with the cost function given by (7.2). The values for the affordance cost, $g_S(x)$ are assigned in the range of 0-10 being proportional to the affordance likelihood 'sitting and working' for each grid cell. The value of w in (7.2) is set to '1' if the minimum distraction path is required and set to '0' if just the shortest path is required.

In experiment 1, the robot is placed in the corridor near a work station and requested to plan a path to a goal location in the other side of the work station as shown in Fig. 7.3. As depicted in Fig 7.3(a), A* planner has calculated the shortest path through a human workspace whereas the GPMD based planner has chosen a path that avoids workstations as shown in Fig. 7.3(b). In this case the difference of travel distances between these two paths is less than the chosen Path Threshold T_p in algorithm 3. Therefore, the robot selects an affordance-map based path as the context-aware path which eventually causes lesser distractions to humans.

In experiment 2, robot is placed in the right corner of the map and instructed to plan a path to a goal location in the opposite corner of the map as shown in Fig. 7.4. There is only one possible path along corridors and all other paths contain at least one working space. According to the results, it can be seen that the shortest path calculated by the A* planner goes through a human work space as shown in Fig. 7.4(a). In contrast the affordance-map based planner has selected the only path that always lies in corridors.



(a) Shortest path with A*

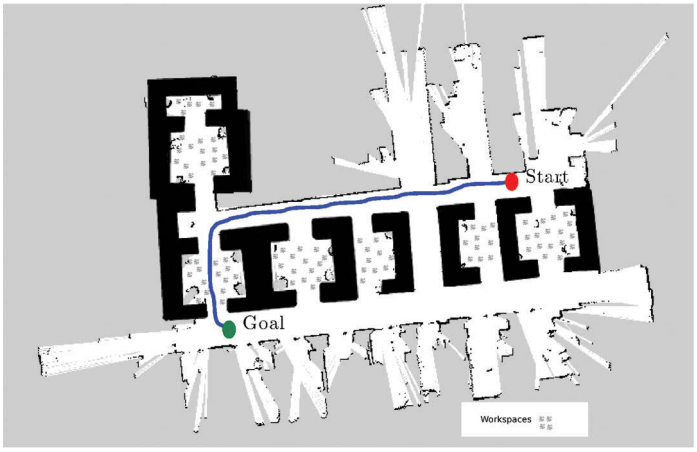


(b) Global Path with Minimal Distractions, GP

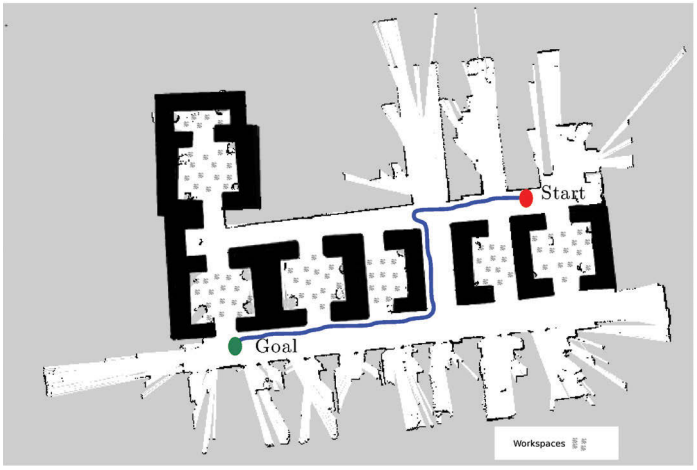
FIGURE 7.3: Experiment 1. Context-aware path planer selects a longer path to minimize the distractions to humans work spaces.

Since both of these paths have very similar travel distances the robot chooses the path based on the affordance-map which causes minimal distractions to humans.

In experiment 3, the robot is placed at the far left corner of the map near a workstation and is requested to plan a path to a location in the other end of the workstation as shown in Fig. 7.5(a). The shortest path is shown in Fig. 7.5(a) and minimum distraction path based on affordance-map is shown in Fig. 7.5(b). Note that there is a considerable difference in travel distances between the shortest path and the minimum distraction path that lies along corridors. Therefore, according to the algorithm 3, robot starts to move along the shortest path while looking for human presence as shown in Fig. 7.6. To make the human detection more efficient, the robot only looks at locations where 'sitting and working' affordance likelihood is high. This leads to a huge computational savings as it does not look for humans in each and every location. At location 'A' of the shortest path the robot detects its first human. Then the robot assigns affordance cost values to the



(a) Shortest path with A*

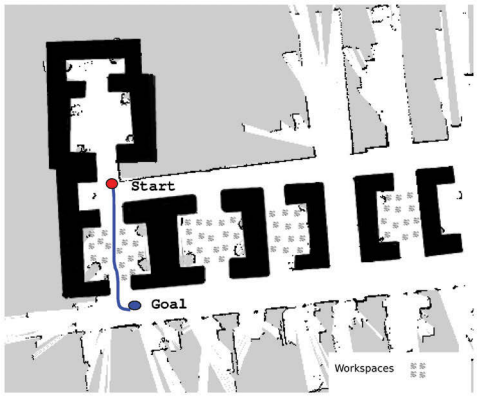


(b) Global Path with Minimal Distractions, GP

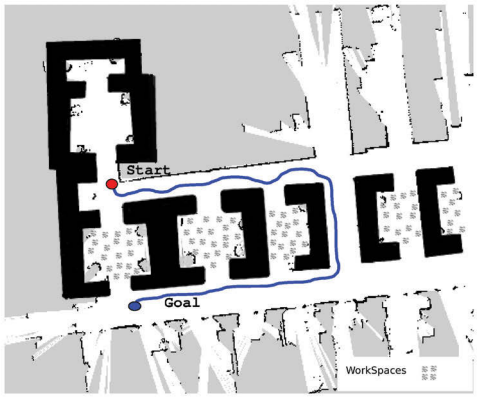
FIGURE 7.4: Experiment 2. Affordance-map based 'Global Path with Minimal Distractions' (GPMD) selects the only path that always lies along corridors.

locations which are within sensory range of the human detector. With this new cost map the robot plans a new path and follows it until it reaches its goal location as shown in Fig. 7.6. Although this new path is going through a human work space the robot does not alter its path as humans are not detected in this region.

Finally, these experiments showed that context-aware path planning can be done in an office environment efficiently using the proposed affordance-map. This map provides much additional information which a traditional grid-based map fails to provide for a successful context-aware path.



(a) Shortest path with A*



(b) Global Path with Minimal Distractions, GPMD

FIGURE 7.5: Experiment 3. Minimum distraction path along corridors is too long and shortest path goes through human works space.

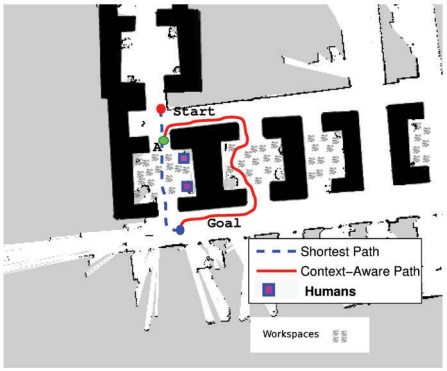


FIGURE 7.6: Experiment 3. Minimum distraction path along corridors is too long. Hence robot tries to find a path through an empty working space.

7.6 Conclusions

This chapter introduced a context-aware path planning method for robots, centered around human context in an indoor environment. First the affordance-map is built for the environment and then human context information embedded in it is used for path planning in an office environment. The experiments showed that the proposed context-aware path planner is capable of avoiding human work spaces and hence contributing less distractions to humans. Therefore, human context information embedded in the affordance-map can be used effectively to reduce complexities associated with context-aware path planning such as human detection, avoiding humans and planing global paths that minimize distractions to humans.

Chapter 8

Conclusions

This thesis addressed the problem of learning human context in indoor environments by analyzing geometrics features of the environment. To solve this problem, the concept of affordance-map was proposed which predicts possible affordances in the environment as human skeleton models with their affordance likelihoods. The problem of affordance mapping was formulated as a multi label classification problem with a binary classifier for each affordance type. In the first phase of this process, virtual human models were moved across the grid locations of the 3D environment to capture features for classification. These features were used to learn parameters of the classification model. Finally, learned classifiers were used to infer the affordance-map in new unseen 3D environments.

To select a binary clasifier for the affordance-map, two popular classifiers namely Support Vector Machine (SVM) and Flexible Naive Bayes (FNB) classifiers were experimentally evaluated. The quantitative and qualitative analyses of the experimental results of the two classifiers demonstrated that the SVM classifier is better suited for affordance mapping. However, the SVM classifier recorded sub-optimum results due to the exsistence of class imbalance in training data. In the search for a solution to this problem, three existing SVM learners namely: focused re-sampling , zSVM and Diffrent Cost model SVM were tested for learning affordances . These algorithms showed some tolerance to moderate class imbalances but failed to record acceptable results particularly in standing-working affordance. Therefore, this thesis proposed to use Structured SVM (S-SVM) optimized

for F1-score to further improve affordance mapping results. This approach defined the affordance-map building process as a structured problem which could be learned efficiently even with extremely imbalanced data. The S-SVM method was experimentally analyzed both quantitatively and qualitatively and outperformed previously used classifiers for affordance mapping.

This thesis also presented two applications of the affordance-map. In the first application, the affordance-map was used by a mobile robot to actively search for computer monitors in an office environment. The pose and location information of humans models inferred by the affordance-map was used to predict probable location of computer monitors. The experimental results in a large office environment showed that the affordance-map concept simplified the search strategy of the robot. In the second application the affordance-map is used for context aware path planning where affordance-map is used by a service robot to plan paths with minimum distractions to office workers. The experimental results proved that a robot could use information from the affordance-map to accurately predict paths with minimal distraction to humans.

8.1 Summary of Contributions

8.1.1 Affordance-map for Learning Hidden Human Context

In this thesis affordance-map is presented as a novel way to represent the human context of the environment. The affordance learning process is formulated as a multi label discriminative classification problem with a binary classifier for each affordance types. These binary classifiers predict labels for each grid location of the map using geometric features obtained by moving virtual human models across the environment. The learned affordance-map consists of two sub maps: 3D skeleton map and 2D affordance likelihood map. The 3D skeleton map holds 3D virtual skeleton models and their pose information. The 2D affordance likelihood map contains confidence values of predictions. These maps represent the human context of the environment which is learned without observing any real humans in the environment. This approach altogether bypasses complexities associated with detecting and tracking real humans. Experiments done in indoor environments

showed that the affordance-map can model real human behaviours by mapping affordance through virtual human models.

8.1.2 Learning Affordances with Extreme Class Imbalances

The 3D scenes used for learning binary classifiers are extremely imbalanced with a higher number of negative examples than the number of positive examples. The initial testing showed that the SVM classifier tends to provide sub-optimum results when trained on this highly imbalanced dataset. As solutions to the problem a number of popular SVM learners namely: random sampling, focus re-sampling, zSVM and different cost model SVM are tested for learning affordances. The k-fold cross validation on different class imbalances showed that these algorithms can handle class imbalances to some extent but resulted in only moderate results in some affordance-types. Therefore, to further improve the results the affordance learning process based on structured SVM is proposed which is directly optimized on the F1-score. As F1-score only considers positive predictions, it is not affected by the class imbalance problem. K-fold cross validation results showed that structured SVM is capable of learning affordances effectively and overall outperformed other methods. Therefore, the structured SVM algorithm is proposed as the best classifier to learn the affordance-map.

8.1.3 Affordance-map for Active Object Search

As an application of affordance-map, this thesis proposes a novel active object search method that uses human context information embedded in an affordance-map to reduce the search space of the searched object. The greatest challenge in active visual object search is estimating the most optimum camera pose that gives the best view of the searched object. As humans arrange objects around them to support their activities, in general there are strong relationships between humans and objects. These humans-object relationships are already available in the affordance-map and are used in this novel method to infer optimum camera poses that give the best views of the searched object. To test the viability of the proposed object search method, experiments are carried out in a large office environment by selecting computer monitors as the object to be searched. The experimental results showed

that the proposed algorithm is capable of using human context information embedded in an affordance-map effectively to reduce the search space of the object.

8.1.4 Affordance-map for Context-aware Path Planning

As another application of affordance-map, this thesis proposes to use affordance-map for context-aware path planning. In this method, affordance-map is used to assign social cost values, and A* algorithm is used subsequently to calculate global paths that minimize distractions to working humans. Experimental results done on a real office environment showed that the proposed method can utilise human context information embedded in an affordance-map to bypass the challenge of detecting humans in context-aware path planning.

8.2 Discussion of Limitations

There are a few limitations of the affordance-map concept presented in this thesis. First, prediction results of the binary classifier depend on the quality of the 3D scene. The proposed algorithms require 3D stitched point clouds so that the geometric features can be calculated. Although there are a number of RGBD SLAM algorithms which are available that can build 3D scenes in real time, care must be taken when moving the camera across the room so that the complete 3D map of the environment could be built. Therefore, most of these map building processes require human supervision during the data capturing phase in order to cover occluded areas of the environment. However, to yield the true potential of the affordance-map concept, mobile robots should have the ability to build the complete 3D map of the environment autonomously without any human supervision. Still this is largely an open problem, and perhaps needs to be addressed further by the 3D map exploration research community.

The second limitation of the affordance-map concept is human context information related to human movements can not be learned from it. Specifically, affordance-map cannot be used to learn human motion patterns, which could be beneficial in an application like human-aware path planning. In such scenarios, detecting and tracking humans could be

much more beneficial. However, affordance-map could be used as a prior map to estimate best locations where more humans could be observed.

The third limitation of the affordance-map is related to the inference stage. In this thesis, 4D grid based approach is used to infer virtual human skeleton models. Therefore, geometric features need to be calculated for each of these grid locations increasing the inference time considerably in large environments. This could become problematic for a robotic applications which require near real-time performance. One solution would be to divide large rooms into small areas and build sub affordance-maps which can be combined together to build the full map. These types of problems are applications specific and need to be addressed individually.

8.3 Future Work

This thesis demonstrated how an affordance-map can be built to model hidden human context in an environment. However, there are a number of improvements and applications that can be explored to further increase usability of the affordance-map concept. Although these suggestions are beyond the scope of this thesis, they are presented here to inspire future research work.

In the current approach for mapping affordances, the dependencies between the output labels are not considered. There could be dependencies between output labels which can be used to improve the accuracy of predictions of the affordance-map. For example, if the current location is supporting the affordance sitting-relaxing there is a high probability that adjacent locations will be supporting the same affordance. These structural dependencies can be incorporated with structured output SVM [21] to further increase the accuracy of the afforance-map. However, the computation burden of learning and inference could be high and needs to be explored further if affordance-map is going to be used for a real time application.

Another interesting application would be to use virtual skeleton models output by the affordance-map as features for 3D scene labeling. For example, if there are many locations in a 3D scene that support sitting-relaxing affordance, then that room could be a living

room. Unlike 3D features used in current approaches for scene labeling, affordance-based features would not be affected by the intra-class variations in furniture types. Therefore, this approach might improve the scene labeling accuracy in complex 3D scenes.

Also affordance-map could be used to improve existing human activity recognition algorithms. Most of the current activity recognition algorithms require a full body view of humans to recognize an activity correctly. However, this could become challenging in a cluttered environment as body parts of humans could easily become occluded by objects in the environments. In these scenarios, affordance-map could be used to predict locations that give full views of humans so the robots can track human body parts more accurately. On the other hand, information from an affordance-map could be used to improve human detection and tracking algorithms. A similar approach used in this thesis for active object search could be used to solve such problems.

Bibliography

[1] Helmut Grabner, Juergen Gall, and Luc Van Gool. What makes a chair a chair? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1529–1536. IEEE, 2011.

[2] Yun Jiang, Hema Koppula, and Ankur Saxena. Hallucinated humans as the hidden context for labeling 3d scenes. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2993–3000. IEEE, 2013.

[3] Stephan Sehestedt, Sarath Kodagoda, and Gamini Dissanayake. Robot path planning in a social context. In *Robotics Automation and Mechatronics (RAM), 2010 IEEE Conference on*, pages 206–211. IEEE, 2010.

[4] Lasitha Piyathilaka and Sarath Kodagoda. Human activity recognition for domestic robots. In *Field and Service Robotics Conference*, pages 567–572. Springer, 2013.

[5] Lasitha Piyathilaka and Sarath Kodagoda. Gaussian mixture based hmm for human daily activity recognition using 3d skeleton features. In *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*, pages 567–572. IEEE, 2013.

[6] Yun Jiang, Hema Koppula, and Ashutosh Saxena. Hallucinated humans as the hidden context for labeling 3d scenes. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2993–3000. IEEE, 2013.

[7] James G. Greeno. Gibson's affordances. 1994.

[8] Lasitha Piyathilaka and Sarath Kodagoda. Active visual object search using affordance-map in real world : A human-centric approach. In *ICARCV, The 13th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2014.

[9] Lasitha Piyathilaka and Sarath Kodagoda. Affordance-map: A map for context-aware path planning. In *ACRA, Australasian Conference on Robotics and Automation 2014*. ARAA, 2014.

[10] Lasitha Piyathilaka and Sarath Kodagoda. Learning hidden human conetext in 3d office scenes by mapping affordances through virtual humans. *Unmanned Systems*, 2015.

[11] Sunlee Bang, Minho Kim, Sa-Kwang Song, and Soo-Jun Park. Toward real time detection of the basic living activity in home using a wearable sensor and smart home sensors. *Conference proceedings Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2008:5200–5203, 2008. ISSN 1557170X. doi: 10.1109/IEMBS.2008.4650386.

[12] Emmanuel Munguia Tapia, Stephen S Intille, and Kent Larson. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. *Pervasive Computing*, 3001:158–175, 2004. doi: 10.1007/b96922.

[13] Duy Tâm Gilles Huynh. *Human Activity Recognition with Wearable Sensors*. PhD thesis, TU Darmstadt, September 2008. URL `http://tuprints.ulb.tu-darmstadt.de/1132/`.

[14] Abhinav Gupta, Praveen Srinivasan, Jianbo Shi, and Larry S Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 21(4): 2012–2019, 2009. ISSN 10636919. doi: 10.1109/CVPR.2009.5206492.

[15] Johannes Stork, Luciano Spinello, Jens Silva, Kai O Arras, et al. Audio-based human activity recognition using non-markovian ensemble voting. In *RO-MAN, 2012 IEEE*, pages 509–514. IEEE, 2012.

[16] Peter H. Kahn, Hiroshi Ishiguro, Batya Friedman, and Takayuki K. What is a human? toward psychological benchmarks in the field of human robot interaction. In *In Proceedings of the IEEE international*, pages 364–371, 2006.

[17] Victoria Bellotti, H. Distributed, Systems Architecture, and Csmil Technical. Design for privacy in ubiquitous computing environments, 1993.

[18] James J Gibson. *The Ecological Approach to Visual Perception: Classic Edition*. Psychology Press, 2014.

[19] Yun Jiang, Marcus Lim, and Ashutosh Saxena. Learning object arrangements in 3d scenes using human context. *arXiv preprint arXiv:1206.6462*, 2012.

[20] I Ben-Gal, F Ruggeri, F Faltin, and R Kenett. Bayesian networks, encyclopedia of statistics in quality and reliability, 2007.

[21] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[22] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[23] Gustavo EAPA Batista, Ronaldo C Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6(1):20–29, 2004.

[24] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[25] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[26] Simon J Sheather et al. Density estimation. *Statistical Science*, 19(4):588–597, 2004.

[27] M Chris Jones, James S Marron, and Simon J Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91 (433):401–407, 1996.

[28] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[29] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured human activity detection from rgbd images. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 842–849. IEEE, 2012.

[30] Ivan Dryanovski, Roberto G Valenti, and Jizhong Xiao. Fast visual odometry and mapping from rgb-d data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 2305–2310. IEEE, 2013.

[31] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *The International Journal of Robotics Research*, page 0278364912461538, 2012.

[32] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, 2012.

[33] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663, 2012.

[34] Alexander Hinneburg and Daniel A Keim. A general approach to clustering in large databases with noise. *Knowledge and Information Systems*, 5(4):387–415, 2003.

[35] Yuchun Tang, Yan-Qing Zhang, Nitesh V Chawla, and Sven Krasser. Svms modeling for highly imbalanced classification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):281–288, 2009.

[36] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine Learning: ECML 2004*, pages 39–50. Springer, 2004.

[37] Bernard W Silverman. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

[38] Gang Wu and Edward Y Chang. Adaptive feature-space conformal transformation for imbalanced-data learning. In *ICML*, pages 816–823, 2003.

[39] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pages 321–357, 2002.

[40] Jamshid Dehmeshki, Jun Chen, Manlio Valdivieso Casique, and Mustafa Karakoy. Classification of lung data by sampling and support vector machine. In *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, volume 2, pages 3194–3197. IEEE, 2004.

[41] Haibo He, Edwardo Garcia, et al. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284, 2009.

[42] Yang Liu, Aijun An, and Xiangji Huang. Boosting prediction accuracy on imbalanced datasets with svm ensembles. In *Advances in Knowledge Discovery and Data Mining*, pages 107–118. Springer, 2006.

[43] Rukshan Batuwita and Vasile Palade. Efficient resampling methods for training support vector machines with imbalanced datasets. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010.

[44] Tasadduq Imam, Kai Ming Ting, and Joarder Kamruzzaman. z-svm: an svm for improved classification of imbalanced data. In *AI 2006: Advances in Artificial Intelligence*, pages 264–273. Springer, 2006.

[45] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

[46] Yunpeng Li, David J Crandall, and Daniel P Huttenlocher. Landmark classification in large-scale image collections. In *Computer vision, 2009 IEEE 12th international conference on*, pages 1957–1964. IEEE, 2009.

[47] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

[48] Shrihari Vasudevan, Stefan Gchter, and Roland Yves Siegwart. *Cognitive Spatial Representations for Mobile Robots: Perspectives from a User Study.*

[49] Jeremy Ma and Joel W. Burdick. A probabilistic framework for stereo-vision based 3d object search with 6d pose estimation. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2036–2042. IEEE, 2010.

[50] Alper Aydemir, K. Sjoo, John Folkesson, Andrzej Pronobis, and Patric Jensfelt. Search in the real world active visual object search based on spatial relations. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2818–2824. IEEE, 2011.

[51] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for three-dimensional point clouds. *The International Journal of Robotics Research*, 32(1):19–34, 2013.

[52] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[53] David S Johnson and Lyle A McGeoch. The traveling salesman problem: A case study in local optimization. *Local search in combinatorial optimization*, 1:215–310, 1997.

[54] A. Torralba, K.P. Murphy, and W.T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004.Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–762–II–769 Vol.2, June 2004. doi: 10.1109/CVPR.2004.1315241.

[55] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a. *J. ACM*, 32(3):505–536, July 1985. ISSN 0004-5411. doi: 10.1145/3828.3830. URL `http://doi.acm.org/10.1145/3828.3830`.

[56] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Adapting navigation strategies using motions patterns of people. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2000–2005. IEEE, 2003.

[57] Matthias Luber, Luciano Spinello, Jens Silva, and Kai Oliver Arras. Socially-aware robot navigation: A learning approach. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 902–907. IEEE, 2012.