ABSTRACT

Title of dissertation: ACTIVE ATTENTION FOR TARGET DETECTION AND RECOGNITION IN ROBOT VISION
Wentao Luan, Doctor of Philosophy, 2017

Dissertation directed by: Professor John S. Baras
Department of Electrical and Computer Engineering

In this thesis, we address problems in building an efficient and reliable target detection and recognition system for robot applications, where the vision module is only one component of the overall system executing the task. The different modules interact with each other to achieve the goal. In this interaction, the role of vision is not only to recognize but also to select what and where to process. In other words, attention is an essential process for efficient task execution. We introduce attention mechanisms into the recognition system that serve the overall system at different levels of the integration and formulate four problems as below.

At the most basic level of integration, attention interacts with vision only. We consider the problem of detecting a target in an input image using a trained binary classifier of the target and formulate the target detection problem as a sampling process. The goal is to localize the windows containing targets in the image, and attention controls which part of the image to process next. We observe that detectors' response scores of sampling windows fade gradually from the peak response window in the detection area and approximate this scoring pattern with an exponential decay function. Exploiting this property, we propose an active sampling procedure to

efficiently detect the target while avoiding an exhaustive and expensive search of all the possible window locations.

With more knowledge about the target, we describe the target as template graphs over segmented surfaces. Constraint functions are also defined to find the node and edge's matching between an input scene graph and target's template graph. We propose to introduce the recognition early into the traditional candidate proposal process to achieve fast and reliable detection performance. The target detection thence becomes finding subgraphs from the segmented input scene graph that match the template graphs. In this problem, attention provides the order of constraints in checking the graph matching, and a reasonable sequence can help filter out negatives early, thus reducing computational time. We put forward a sub-optimal checking order, and prove that it has bounded time cost compared to the optimal checking sequence, which is not obtainable in polynomial time. Experiments on rigid and non-rigid object detection validate our pipeline.

With more freedom in control, we allow the robot to actively choose another viewpoint if the current view cannot deliver a reliable detection and recognition result. We develop a practical viewpoint control system and apply it to two human-robot interaction applications, where the detection task becomes more challenging with the additional randomness from the human. Attention represents an active process of deciding the location of the camera. Our viewpoint selection module not only considers the viewing condition constraints for vision algorithms but also incorporates the low-level robot kinematics to guarantee the reachability of the desired viewpoint. By selecting viewpoints fast using a linear time cost score function, the

system can deliver smooth user interaction experience. Additionally, we provide a learning from human demonstration method to obtain the score function parameters that better serves the task's preference.

Finally, when recognition results from multiple sources under different environmental factor are available, attention means how to fuse the observations to get reliable output. We consider the problem of object recognition in 3D using an ensemble of attribute-based classifiers. We propose two new concepts to improve classification in practical situations, and show their implementation in an approach implemented for recognition from point-cloud data. First, we study the impact of the distance between the camera and the object and propose an approach to classifier's accuracy performance, which incorporates distance into the decision making. Second, to avoid the difficulties arising from lack of representative training examples in learning the optimal threshold, we set in our attribute classifier two threshold values to distinguish a positive, a negative and an uncertainty class, instead of just one threshold value. We prove the theoretical correctness of this approach for an active agent who can observe the object multiple times.

Active Attention for Target Detection and Recognition in Robot Vision

by

Wentao Luan

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2017

Advisory Committee:
Professor John S. Baras, Chair/Advisor
Professor Yiannis Aloimonos, Dean's Representative
Professor Cornelia Fermüller
Professor Behtash Babadi
Professor Yu Chen

Acknowledgments

I owe my gratitude to all the people who supervised, supported, encouraged and inspired me during my Ph.D. life. I will cherish and keep in mind the achievements and lessons I learned in my graduate study.

First, I would like to express my sincere thanks to my advisor, Dr. John S. Baras for giving me the great opportunity to explore an emerging and promising research topic. His deep insight into a broad area and sharp sense of future calibrated me on the right path of research. His rigorousness in problem formulation phrased my research attitude and helped me search for solutions on a wide view. Also, his enthusiasm and focus in handling challenges motivated me to move ahead in the past four years and will continue guiding me in my future career.

I would also like to thank Dr. Cornelia Fermuller and Dr. Yiannis Aloimonos for the selfless support, precious energy, and exquisite guidance in academic communication and leading me to the professional level of research. Also thanks to Dr. Yezhou Yang for advising my research.

I also would like express my appreciation to Dr. Behtash Babadi and Dr. Yu Chen for serving on my thesis committee and for the influential discussions and feedback.

Sincere gratitudes to Ren Mao, Xiangyang Liu, Xiangnan Weng, Peixin Gao, Yuchen Zhou, and Yi Zhang who worked with me in fighting difficulties and with whom I grew together. Also, I would like to thank Mrs. Kim Edwards for her excellent administrative work.

Last but not least, I want to express my deepest gratitude to my parents, Yi Ding and other family members for the constant support and acknowledgment.

I would like to acknowledge the support offered by DARPA (through ARO) grant W911NF1410384, by NSF grant CNS-1544787, and by US Air Force Office of Scientific Research grant FA9550-10-1-0573.

Table of Contents

List of Figures

Chapter 1: Introduction

1.1 Target Detection in Robotics Vision

With the increasing demand for automation from both industrial manufacturing and daily life, the robot has become a hot topic that would contribute to a significant part of the production and the service market in future. In contrast to traditional robot applications, where a pre-programmed robot blindly performs routine tasks, nowadays people are expecting more intelligent robotic services that can be customized and can handle complicated environment interferences reliably.

Efficient target detection and reliable object recognition are among the essential tasks that make robots adaptive in the working scene and general in dealing with task specifics. During the job execution, the vision module usually answers fundamental questions like, where the target is, what is the status of the target. So an accurate result is vital for the successful completion of the tasks. Moreover, the time and computational efficiency is also an issue to be considered since it is directly related to the quality of the robotic service.

Object detection and recognition is an important task that has attracted a lot of attention in the computer vision community. Researchers have put forward various methods on the vision pipeline, such as feature abstraction, classification

and multiple sources fusion, to increase the detection and recognition performance.

However, the characteristics of robot tasks pose different requirements compared to the tasks of conventional computer vision. First, the detection or recognition results have to be more accurate, because a reliable execution pipeline cannot be built on a random correct vision input. An accuracy competition on a dataset would not help solve a robot's vision demand. Furthermore, for many tasks like human-robot interaction, a fast response from the vision module is required for the benefits of user experience. Also, limited resources in computation, power, and communication put additional constraints on the vision algorithms.

Fortunately, in robotics, engineers have extra degrees of freedom to deal with the challenging vision requirement. One important advantage is, at the task level, robots could have good knowledge of the target. For example, when to heat an object, a robot would know it should look for a microwave, so it can prepare before the task happens. Additionally, since robotics is a complete system consisting of vision, control, etc. the control module can be introduced to enhance the vision if necessary. e.g. the robot can change its observing viewpoint if not satisfactory with the current results. Besides, the number of observations can also become a controlled parameter while in conventional Computer Vision methods, algorithms have to give an answer based on the static images.

1.2 Main Contributions and Thesis Organization

In this dissertation, we aim to achieve efficient target detection and object recognition for robotics tasks. Specifically, we provide solutions to four situations: (1) For single image processing, how to detect the target efficiently when only a detector(binary classifier) is available; (2) Still take a single image as input, but can learn the target before, how to locate the target efficiently using the knowledge about the object; (3) On the control level, what is the next viewpoint if the current observing quality would not satisfy the task; (4) For the attribute-based classifier with multi-sources input, how to fuse the results when the number of observations can be controlled and inpput information may be unreliable.

Each chapter of this thesis answers one question above. In general, our solution can be interpreted as introducing attention mechanism into different levels of target detection pipeline. i.e. Focusing fast and on the right thing is the key to the efficiency and reliability of target detection and recognition. The main contributions and thesis organizations are summarized below.

1.2.1 Active Sampling Exploiting Detector Response Pattern

In chapter two, we treat the detection process as a sampling problem when only a binary classifier of the target is available. Efficient target detection becomes how to find the window(bounding box) containing the target with a small number of trials. We observe that the classifier's response score would follow a "half-ellipsoid" shape in the detection area. Thus an exponential decay function is used to model this

response pattern in the positive area. Exploiting this property, we propose an active sampling approach which estimates the probability of windows containing the target based on responses of observed windows and then chooses the next window according to posterior sampling. Experiments on the human detection dataset show that our method achieves higher detection rate with the same sampling windows number, and also requires fewer windows under comparable performance when compared with the sliding windows and multi-scale particle window method.

1.2.2 Fast Task-Specific Target Detection via Graph Based Constraints Representation and Checking

Chapter three deals with the case where interaction and learning with the target object are allowed before the detection. How to depict an object and utilize the knowledge becomes another practical problem in robot vision.

We describe the target as a set of template graphs over the segmented target object surfaces and define constraints for matching template graphs to input images. To speed up the graph matching, we prove that a greedy strategy of organizing constraint filters has a bounded performance concerning the optimal checking sequence. We implement and apply our framework to two different scenarios: the detection of drawers with handles and the detection of hands and the arms. The experimental results show the feasibility and benefits of introducing target descriptions early into the segmentation and object candidate proposal procedure for robotic applications. And the time reduction performance of our constraint filtering strategy is validated.

1.2.3 Active View Point Control for Reliable Target Detection in Human-Robot Interaction

In chapter four, we tackle the next viewpoint selection problem for object detection when the camera is mounted on the robot's arm. Specifically, we focus on the scenarios of human-robot interaction which requires highly of system's timely response, and human's randomness would increase the detection difficulty.

We propose a practical active viewpoint control strategy considering factors of the joint discontinuity of moving to a new position, target object's viewing centeredness, occlusion and viewing angle. By using a linear score function and precompute time costly intermediate parameters, the viewpoint control module can make the next view decision in time. To guarantee the selected viewpoint better serve the task, we adopt coactive learning method to learn score function's weights. We build two human-robot interaction applications and apply our viewpoint control module, demonstrating the usability of our proposed system.

1.2.4 Reliable Attribute-Based Object Recognition Using High Predictive Value Classifiers

Chapter five attends to the problem of component classifiers' observation fusion in attribute based 3D object recognition. Because an active agent can observe multiple times in the testing time, we propose to use two thresholds, one aiming for high-precision prediction for the positive class and the other for high negative

predictive value prediction for the negative class. Thus each attribute classifier will output three possible values: positive, negative and uncertain. We also incorporate environment factor into the decision making considering its influence to each component classifier. A reliable working region is defined indicating a fair separation of the distributions of positive and negative classes. We prove our fusion framework's asymptotic correctness under certain assumptions on the attribute classifier and randomness of the input data. Experiments are also done to valid our theorems.

Chapter 2: Active Sampling Exploiting Detector Response Pattern

2.1 Introduction

With the emerging social demand of robotics automation in both industry and daily life, robotics and computer vision systems have become an important and popular research area. Efficient object detection and recognition are among the most fundamental robot's tasks, on which many subsequent actions such as assembly, fetching, obstacle avoidance rely. Here, the task of object detection can be understood as segmenting the target out from the input image or video and the result can be in the form of a window (i.e. bounding box) or a contour enclosing the target.

The cardinality of search space could be extremely large considering windows of different locations and sizes. Therefore an exhaustive search would be very expensive. Many works in computer vision try to reduce the search space utilizing additional features. For example, segmentation techniques [1, 2] use information such as color, edges and texture similarity to cluster image pixels into super-pixels to avoid a brute-force searching.

In our work, we focus on a typical situation where only a trained detector (binary classifier) is available and we desire to detect the target efficiently from the

given input image. The general pipeline of detection includes three steps: window selection, feature abstraction and classification, where the provided detector implements the last two stages. The window selection scheme will determine the detection system's efficiency and quality.

A traditional manner is to slide a window of various sizes over the input image, from left to right, top to bottom and feed image patches to a binary target detector indicating whether the target exists. However, this sliding window method would run the detector a lot of times considering the potentially large number of image windows and it gets even worse when the feature abstraction and classification are complicated. The scanning step size can be increased to speed up but the accuracy will be traded off because the target may be skipped or the windows may not be aligned with the target very well.

To improve this static scanning scheme, one practical way is to regard window selection as a sampling problem, which is to treat the provided vision detector as a black box and sample windows based on the detector's response characteristics. For example, assuming the detector's response score on adjacent windows are similar, multi-stage particle window method (MS-PW) [3] samples windows in stages and follows a "coarse-to-fine" principle.

With the same insight of using the detector's property but going deeper, in this work we propose an active sampling method considering response pattern for efficient target detection. The main contributions of this paper are: 1) We observe that the detector's response pattern of sampling windows in the image follows a "half-ellipsoid" shape in the detection area (i.e. positive classification area). Then

an exponential decay function is used to model the response pattern in the positive area. 2) We propose an active sampling approach by exploiting such pattern, which estimates the probability of windows containing the target based on responses of observed windows and then chooses the next window according to posterior sampling. 3) The proposed method is implemented in the application of human detection and experimental results show that our method achieves higher detection rate with the same sampling windows budget and also requires fewer windows with comparable performance when compared with the sliding windows and MS-PW method.

2.2 Related Work

Efficient target detection has gained much attention and there are many directions of the trial to cut the detection time while maintaining good detection performance. In general, the attempts in speeding up classification procedure tend to find an early rejection strategy on negative samples, while the work on candidate generation procedures can be summarized as reducing the search space using different sources of information. Also, there is not a clear boundary between classification and candidate proposal. Therefore, these methods can be combined.

An attentional cascade is a classical approach to boost average classification speed, in which the fundamental idea is that background and irrelevant image patches usually occupy the largest portion of all window space and they can be rejected early in the designed cascade classification pipeline. This mechanism achieves good results in applications such as face detection [4] and car detection [5].

Applying a similar idea to reduce the cost of the recognition pipeline, a deformable part model [6] firstly runs a root filter over a downsampled image to filter negative windows out. Andrea *et al.* [7] run an object detector with a linear kernel before using more discriminative but also more time-consuming non-linear kernel ones.

On the other hand, reducing the search space is an approach aiming to reduce the total times of running a target detector, instead of cutting the classification time for each time. Image segmentation is a classical method exploiting low-level information. A common process is to over-segment the image into small boxes, or superpixels, then use graph algorithms, such as minimal spanning tree and graph cut minimization, to build meaningful candidate regions [1, 8, 9]. Selective search [2] generates candidates by hierarchically grouping small regions in a bottom-up manner. Multiscale combinatorial grouping [9] segments image at different scales hierarchically and generate object candidates by grouping and ranking. Making use of the close contour property of daily objects, the torque operator [10] can provide a reliable source of object candidates and even in high clutter environments [11].

Similar with image segmentation, though more bio-inspired, saliency can be another scheme to speed up detection by imitating human recognition behavior, which always focuses objects standing out of their neighbors pre-attentively. Koch and Ullman [12] firstly put forward a computational attention architecture consisting of the Winner-Take-All network to determine the most salient region, and one of its most well-known derivatives is the Neuromorphic Vision Toolkit [13] proposed by Itti, which is a bottom-up computational attention framework based on the center-surround mechanism of color, intensity and orientations.

Another approach to reduce searching workload is to take advantage of context information. It has attracted more attention recently when incorporated with a sequential decision strategy to optimize the observation path. Gonzalez-Garcia *et al.* [14] adopts context knowledge (a spatial distribution of target) into the windows selection procedure achieving the same detection accuracy with the original region feature convolutional neural network pipeline [15], while using a reduced number of sampling windows. In the indoor environment Nagaraja *et al.* [16] studies structure information such as objects' relative positions to choose the next candidate to observe for target detection. Mnih *et al.* [17] presents a recurrent neural network framework that can decide the next observation region and recognize a target with the same state configuration.

The last category mentioned is window sampling, which our work falls into. It seeks to learn the distribution of the target via sampling the input image. One advantage is avoiding the preprocessing such as edge detection, context analysis, which makes its application general. Multi-stage particle window (MS-PW) [3] samples images iteratively and updates the distribution of the target by a mixture of Gaussians. Pang *et al.* [18] advances MS-PW by classifying observed regions as rejection, ambiguity and acceptance regions based on classifier's response scores. Compared to those attempts, our work applies a distinct way in learning the target distribution which focuses on the detector's response pattern on the positive classification area.

2.3 Problem Definitions

Given a sensor image I as input, the goal is to find out sampling windows within the image that contain target objects. We describe the center point of sampling window w_i as pixel coordinate (x_i, y_i). As we fix the ratio between length and width of the sampling window according to the property of target detectors, the size of window w_i could be represented as an integer scale level ($s_i = 1, 2, \dots$) given base size and scale factor. Examples are shown in Fig. 2.1. Therefore, the complete set of possible sampling windows is defined as $\mathbb{W} = \{w_i | w_i = (x_i, y_i, s_i)\}$.

Figure 2.1: Windows of different sizes with the same center point. Base size is 64×128 and scale factor between two scale levels is 1.05. From left to right, the scale levels are 1, 5, 9 and 13.

After selecting the sampling window w_i for the current iteration, the target detector, a binary classifier, takes the corresponding patch from the image I as input and returns a detection score $f(w_i)$ as output, where $f(\cdot)$ depends on the classification algorithms in the detector. The range of such response scores may be different in different detection applications. For instance, the human detection system [19] uses the histogram of oriented gradients (HOG) feature and the SVM

classifier, while the detector's response score is a real value which mostly falls into $(-10, 10)$. Score higher than a specified threshold indicates a detection of a target. While in the case of face detection using Haar-like features and the cascade AdaBoost classifier [4], the detector's response score can be defined as $f(w_i) = l_{w_i}/L$ where l_{w_i} is the largest index of stage returning positive results for input window patch w_i and L is the total number of stages in the cascade classifier. Then, the range of such response is $[0, 1]$.

To efficiently detect the target, we aim to sample as small number of windows as possible to reduce the usage of the target detector while maintaining good detection performance, especially when the feature abstraction and classification processes are time-consuming.

Here we set our goal as maximizing the number of windows containing the target sampled when the total number of windows allowed to sampled is limited.

Next we are going to discuss the property of detector's response score and formulate a sequential sampling problem solved using this property.

2.4 Active Sampling with Response Pattern

2.4.1 Detector's Response Pattern

One of the key ideas of our work is to make use of detector's response property to evaluate the possibility of an unobserved window containing the target.

To start with, let us look at detector's response pattern with an example in Fig. 2.2, where (a) is an input image and (b) is the heat map for the response score

of human detector [19]. Each point in the heat map represents the center point of a sampling window and all sampling windows are of the same size. Fig. 2.2(c) explains the pattern of regions that can return positive classification results in 3D. From the figure, we can observe:

- **"Continuity" of Detector's Response Score** The response score of the detector on two nearby windows (same size and close center points) will not change too significantly.

- **Half-ellipsoid Pattern of Detection Area** The red region in the heat map is the detection area that returns positive results if detection threshold is set as 0. By looking at it in 3D, we recognize the overall shape of the detection area is like a half-ellipsoid, which tells that the detector's response score decays gradually with the increment of a window's distance to the peak response window in the detection area.

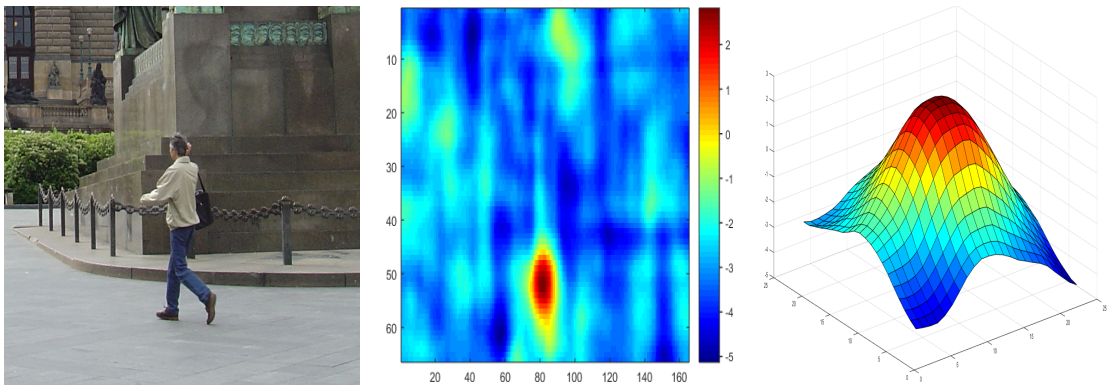


Figure 2.2: Illustration of response pattern. (a) Input image. (b) Heatmap of human detector's response score. (c) Positive classification region (red area in (b)) in 3D.

Although different detectors (binary classifiers) may have diverse ranges of

response score, many of them may still have similar response patterns when the target is not occluded severely. Also, this reaction pattern could be observed in some other target applications though we are focusing on visual object detectors here, thereby the sampling strategy exploiting such pattern can also be applied. Next, we approximate the response decay using an exponential function and utilize this pattern to estimate the probability of an unobserved window containing the target given observed results. Therefore, we can sample windows more efficiently.

2.4.2 Formulation

In general, we formulate this process of window sampling for target detection as a Markov Decision Process (MDP).

At iteration t, the fully-observable state consists of all sampled windows and their corresponding detector's response scores $s_t = \{(w_i, f(w_i)), w_i \in \mathbb{W}_e^t\}$, where $\mathbb{W}_e^t$ represents the set of all sampled windows at iteration t. Action a_{t+1}, which is the window to observe at time $t+1$, is selected among all the unexplored windows $\mathbb{W}/\mathbb{W}_e^t$. A binary reward is defined such that the reward is 1 for sampling a window that can return highest local response score in positive classification regions (i.e. $h(w) = 1$ defined in equation (2.2)) and 0 otherwise.

Our goal of efficient target detection is to minimize the number of total sampling windows while still achieving a certain number of windows containing the target. This could also be considered as maximizing the number of sampled windows that provide a local peak (highest) response in detection area given a constraint on

the total number of windows to be checked.

Formally, our objective function is:

$$\begin{aligned} &\text{maximize} \quad |\{w \in \mathbb{W}_e^t | h(w) = 1\}| \\ &\text{subject to} \quad t \leq M. \end{aligned} \tag{2.1}$$

where M is the bound on total iterations and also is the total number of windows to be sampled since only one window would be sampled in each iteration, $|\cdot|$ denotes the cardinality of the set and the function $h(\cdot)$ is an indicator of whether a window has a local maximum response in the detection area:

$$h(w) \triangleq \begin{cases} 1 & \text{If } f(w') \leq f(w) \; and \; f(w) > \tau \\ & \text{for } \forall w', d(w', w) < \delta \\ 0 & \text{o.w} \end{cases} \tag{2.2}$$

In (2.2), τ is a threshold related to the detector that is used to determine positive results. $d(\cdot)$ measures the distance between two windows and $\delta > 0$ is a threshold to determine local neighbors.

Since it is difficult to estimate directly the detector's response score of a selected window patch in each iteration based on observed windows and their scores, i.e., the transition probabilities are unknown, traditional MDP solutions cannot be adopted here. However, through sampling interaction between the input image and the detector's response, it is achievable to learn the distribution of the defined binary reward among unexplored windows. Accordingly, we could maximize our objective

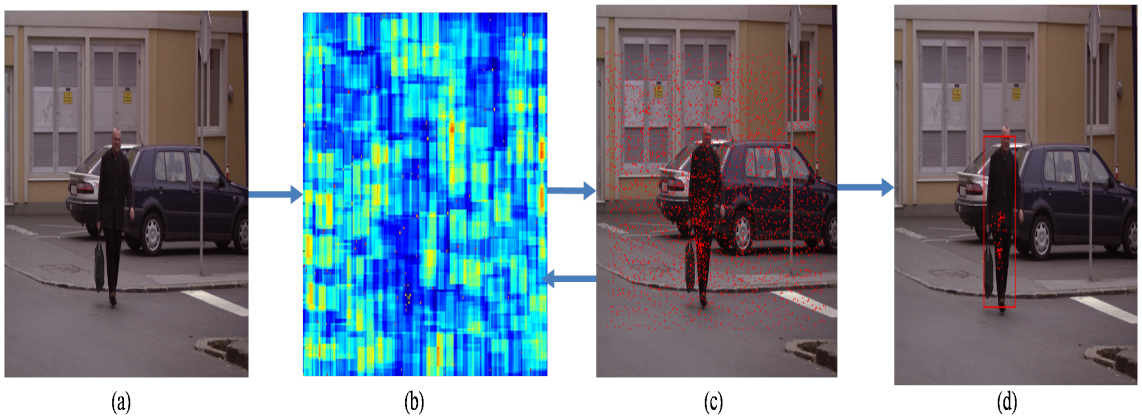


Figure 2.3: System procedure example. (a) Input image. (b) Estimation error heat map of all windows with same scale. (c) The center points of observed windows. (d) Output the positive classified windows.

rewards according to that estimated distribution.

The overall procedure is demonstrated with an example in Fig. 2.3. Given an input image Fig. 2.3(a), we calculate an estimation error (Fig. 2.3(b)) of each window having local peak response in the detection area based on all the sampled windows (Fig. 2.3(c)) and their corresponding detector's response score s_t. Then the next window to be tested is chosen according to the posterior sampling on the distribution of the binary reward derived from the estimation error obtained above. There is a loop between (b) and (c) because with the newly sampled window and its detector's response added, the reward distribution is reevaluated, and a new window will be selected to be sampled until it achieves the limited total number. Finally, outputs are the positive classified windows (Fig. 2.3(d)).

In the following sections, we detail on how to evaluate the distribution of the binary reward and how to choose the next action given current observations.

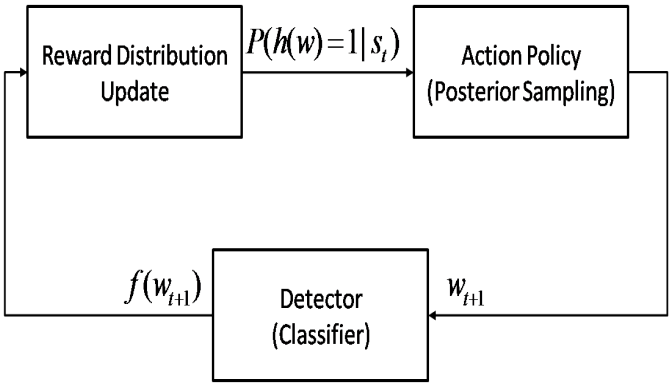


Figure 2.4: System block diagram

2.4.3 Reward Distribution Evaluation

In this section we will elaborate our reward distribution evaluation method. Based on the definition of the binary reward above, the probability of getting a reward 1 is the same as the probability of the selected window returning locally highest response score in the detection area given current observations, i.e. $P(h(w) = 1|s_t)$.

The procedures to calculate the probability $P(h(w) = 1|s_t)$ at window w are as follows. 1) We predict the detector's response score $\hat{f}(w')$ of windows w' that locally surround window w, assuming window w was the peak window in the detection area. This step applies the response pattern that the detector's score exponentially decayed with the increment of distance between a surrounding window w' and the peak response window w. 2) After we observe the response score $f(w')$ for each iteration, we compare it with the predicted one and obtain the prediction error. 3) The prediction errors of all surrounding windows of window w are entered in an energy function, and we evaluate the probability of window w being the local peak window in the detection area.

Formally, given current observation s_t, the probability of a window w being the peak window in the detection area is evaluated as:

$$P(h(w) = 1|s_t) = \frac{1}{Z} \exp(-\sum_{i=1}^{t} E(w_i, f(w_i)|w, \theta^*)) \tag{2.3}$$

where Z is the normalization factor and the energy function $E(\cdot)$ is defined regarding the error between the observed and predicted detector's response score. The error function is defined as:

$$E(w_i, f(w_i)|w, \theta) = \begin{cases} ||f(w_i) - \hat{f}(w_i|w, \theta)||^2 & \text{if } w_i \in R(w) \\ 0 & \text{o.w} \end{cases} \tag{2.4}$$

Here $\hat{f}(\cdot|w)$ is the predicted detector's response function assuming w was the peak response window, and $R(w)$ denotes the influence (cutoff) area for window w.

According to the detector's response pattern observed above, the predicted detector's response could be written as:

$$\hat{f}(w'|w, \theta) = C \exp(-(w' - w)^T \Sigma^{-1} (w' - w)) \tag{2.5}$$

and $\theta = (C, \Sigma^{-1})$ are parameters determining the peak response score and the decaying speed of scores surrounding the peak window.

Given a range for parameter θ, we need to estimate a value best fitting the current observation s_t. The estimation is done by minimizing the prediction error

of all observed window patches:

$$\theta^* = \arg \min_{\{\theta: C > \tau\}} \sum_{i=1}^{t} E(w_i, f(w_i)|w, \theta) \tag{2.6}$$

Finally the predicted detector's score is determined as $\hat{f}(w'|w, \theta^*)$ and the energy function will compare the truly observed response score $f(w_i)$ with the predicted score $\hat{f}(w_i|w, \theta^*)$ to update the probability of window w being the peak window in the detection area.

Even though we determine the maximum likelihood (minimum prediction error) parameter θ^* for all the unexplored windows, the update process can be fast if we restrict to a finite set of values for θ and use the kernel trick. A kernel function based on the observed windows can be defined: $q(w|w_i, f(w_i), \theta) \triangleq E(w_i, f(w_i)|w, \theta)$. The function's value under different θ and $f(w_i)$ can be pre-computed, where we can discretize $f(w_i)$ by binning if it takes a continuous value. As a result, the probability can be simply estimated through kernel functions:

$$\begin{aligned} -\log P(h(w) = 1|s_t) &\propto \min_{\theta} \sum_{i=1}^{t} q(w|w_i, f(w_i), \theta) \\ &= \min_{\theta} \sum_{w_i \in R(w)} q(w|w_i, f(w_i), \theta) \end{aligned} \tag{2.7}$$

When a new observation $(w_t, f(w_t))$ is made, only the probability of windows within the influence area of w_t: $w \in R(w_t)$ needs to be updated.

2.4.4 Active Sampling Action Policy

Given the reward distribution estimated based on the current observed state, we select an unexplored window to be sampled at the next iteration. In order to better balance exploration and exploitation during iterations, *Posterior Sampling* [20] is employed here as our action policy. The key idea of posterior sampling is to instantiate beliefs based on the posterior distribution given current observations in each iteration, then choose an action that can maximize the expected reward.

As the binary reward is gained only when the sampling window w is a peak response window in the detection area and the reward posterior distribution is estimated as described in the previous section, our action policy to select the next sampling window simply becomes:

$$P(A_{t+1} = w|s_t) \propto P(h(w) = 1|s_t) \tag{2.8}$$

where A_{t+1} denotes the action variable for iteration $t + 1$.

Algorithm 1 shows the overall active sampling algorithm.

2.5 Experiments

In this section, we evaluate our sampling method with Multi-Stage Particle Windows sampling (MS-PW) [3] to demonstrate that our proposed method obtains better efficiency while maintaining good detection performance through exploiting the detector's response pattern.

Algorithm 1: Active Sampling with Response Pattern

Parameters:

Total number of windows to be sampled: M;
Parameters set for prediction functions $\{\hat{f}_i\}$: $\{\theta_i\}$;
Influence region function $R(\cdot)$;
Detection threshold τ.

Input:

Image to be detected: I;
Target detector returning response $f(w)$ with input w.

Output:

Set of sampled windows with postive results: $\mathbb{W}_p$.

Pre-compute / load kernel functions $\{q_i(\cdot)\}$ for all $\{\theta_i\}$
Initialize the prediction error w.r.t each kernel function and minimum prediction error for all the window: $\{E_i(w) = 0\}$, $E^*(w) = 0$
Initialize the probability of each window being locally peak window in detection area: $p(w) = P(h(w) = 1|s_0) = \frac{1}{Z}\exp(-E(w)) = \frac{1}{Z}$
Set: $\mathbb{W}_p = \emptyset$
for $t = 1$ to M **do**
    Sample a window w_t proportionally to $p(w)$
    Observe detector's response $f(w_t)$
    for $\forall w \in R(w_t)$ **do**
        for each kernel function q_i **do**
            $E_i(w) = E_i(w) + q(w|w_t, f(w_t), \theta_i)$
        end for
        Update $E(w)$: $E(w) = \min_{i \in \{1,\ldots,M\}} E_i(w)$
        Update $p(w)$: $p(w) = \frac{1}{Z}\exp(E(w))$
    end for
    if $f(w_t) > \tau$ **then**
        $\mathbb{W}_p = \mathbb{W}_p \cup \{w_t\}$
    end if
end for

MS-PW is chosen as a comparison method because both methods detect targets only by sampling and using the detector's response without adopting other pre-processing techniques such as segmentation [2, 9].

It also samples windows in iterations and the number of windows sampled in each iteration decreases as the iteration goes on. From the observation that windows located nearby should have similar classification score, it estimates the distribution of positive classification windows based on the observed windows score using Gaussian kernel density estimation. Windows in the next iteration to sample are chosen according to the updated positive window distribution and distribution is updated again with new observations coming.

The flow of MS-PW is shown in Algorithm 2.

Algorithm 2: Multi-Stage Particle Windows Sampling

Input: :
The number of stages S;
The number of windows to sample in each stage N_i , $i = 1, ..., S$;
Total number of windows N to sample
Detection threshold τ;

Output: :
A set $\mathbb{W}_p$ of all the positive classification windows sampled.

1: Set: $\mathbb{W}_p = \emptyset$
2: Initialize the proposal distribution $g_0(w)$ for all the windows: $g_0(w) = \frac{1}{N}$
3: **for** $t = 1$ to S **do**
4: Sample a N_t window from $g_{t-1}(w)$: $W_t = \{w_1, ..., w_{N_t}\}$.
5: Transform the response score to positive if classifier's score can be negative. Then normalize classifier's response score: $f_N(w_i) = \frac{f(w_i)}{\sum_{j=1}^{N_t} f(w_j)}$
6: Update proposal distribution : $g_t(w) = (1 - \alpha)g_t(w) + \alpha \sum_{j=1}^{N_t} f_N(w_j)G(w_j, \Sigma_t)$ p
7: Update $\mathbb{W}_p = \mathbb{W}_p \cup \{w_i | w_i \in W_t \& f(w_i) > \tau\}$
8: **end for**

We test the algorithms' performance via several evaluation metrics including

detection rate, window usage efficiency, the average precision rate given the same budget and overall system detection performance using different sampling window budget.

2.5.1 Dataset and Settings

We assess our sampling method on the INRIA person dataset [19]. The training set contains 1208 cropped person patches for positive examples and 1218 non-person images where negative example patches can be sampled from. In the testing set, there are 453 images of scenery and buildings without people and 288 images containing one or more persons. Most people in testing images are standing, but they appear in different orientations and various backgrounds such as shops, statues and pillars. In this work we are addressing a detection problem, so that full images in the testing dataset are used to evaluate our algorithm's performance. A SVM classifier trained with HOG features is employed as the human detector, which takes input images of 64×128 pixels.

In all experiments, we set our influential region $R(w)$ as a cube of size $21 \times 31 \times 5$ pixels (width, height, scale) centered at observed window w. According to the observed detector's response pattern, we restrict the prediction function $\hat{f}(\cdot)$ to the set of parameters $\{\theta_1, \theta_2\}$: $(C_1, \Sigma_1^{-1}) = (1.2, diag(10, 20, 5))$ and $(C_2, \Sigma_2^{-1}) = (2.2, diag(25, 35, 5))$. Fig. 2.5 illustrates our prediction function using parameters θ_1, θ_2.

The ratio between width and height of each window is fixed as $\frac{1}{2}$ according

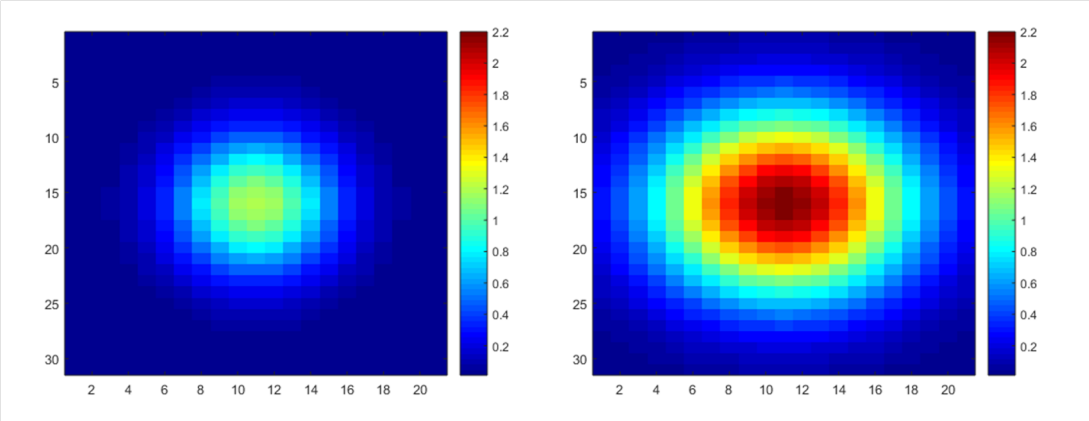


Figure 2.5: The heat map of predicted score with our two settings of parameters. Each point corresponds to a window in the same size with the selected observed window.

to the detector's input requirements. And the scaling factor for the window size of two adjacent levels is set as 1.05. Meanwhile, the total number of possible sampling windows varies with different sizes of input images. We denote N_{sw} as the total number of sliding windows when we scan images both vertically and horizontally with a stride of 8. Then we limit the total number of windows to be sampled in experiments proportional to N_{sw}.

2.5.2 Experimental Results

The first experiment compares the detection rate under the same false positive rate per image (FPPI = 1) between MS-PW and our method. Here the false positive rate is measured per image instead of per window because we allow multiple targets detected in one picture, even though the latter one is the standard metric for traditional classification problems. The outcome is shown in Table 2.1 top, from where we can notice that with the same budget number of windows to be sampled,

our method has higher detection rate and hits more windows with positive results than MS-PW.

	# of win.	Method	Detection Rate	# of Positive Windows
FPPI = 1	$1/7N_{sw}$	MS-PW	0.624	10.4
		Our	0.716	18.3
	$1/6N_{sw}$	MS-PW	0.650	11.4
		Our	0.718	23.7
	$1/5N_{sw}$	MS-PW	0.652	12.0
		Our	0.721	31.0
	$1/4N_{sw}$	MS-PW	0.667	14.1
		Our	0.725	42.3
	$1/3N_{sw}$	MS-PW	0.691	17.5
		Our	0.726	57.6
	$1/2N_{sw}$	MS-PW	0.708	24.0
		Our	0.728	75.2
$\tau = 0$	$1/7N_{sw}$	MS-PW	0.587	7.0
		Our	0.677	15.4
	$1/6N_{sw}$	MS-PW	0.596	8.1
		Our	0.681	20.6
	$1/5N_{sw}$	MS-PW	0.604	9.1
		Our	0.688	28.0
	$1/4N_{sw}$	MS-PW	0.652	11.8
		Our	0.713	38.6
	$1/3N_{sw}$	MS-PW	0.684	15.6
		Our	0.714	53.5
	$1/2N_{sw}$	MS-PW	0.708	23.7
		Our	0.719	72.5

Table 2.1: Detection performance with same sampling window budget. Top: Classification threshold customized to FPPI = 1.
Bottom: Classification threshold $\tau = 0$

The second experiment contrasts sampling efficiency between methods, i.e., the number of sampled windows with positive detection results per image using the same number of total sampling windows. The detector (binary classifier)'s threshold is identical ($\tau = 0$) for fair comparison. The consequence is displayed in Table 2.1

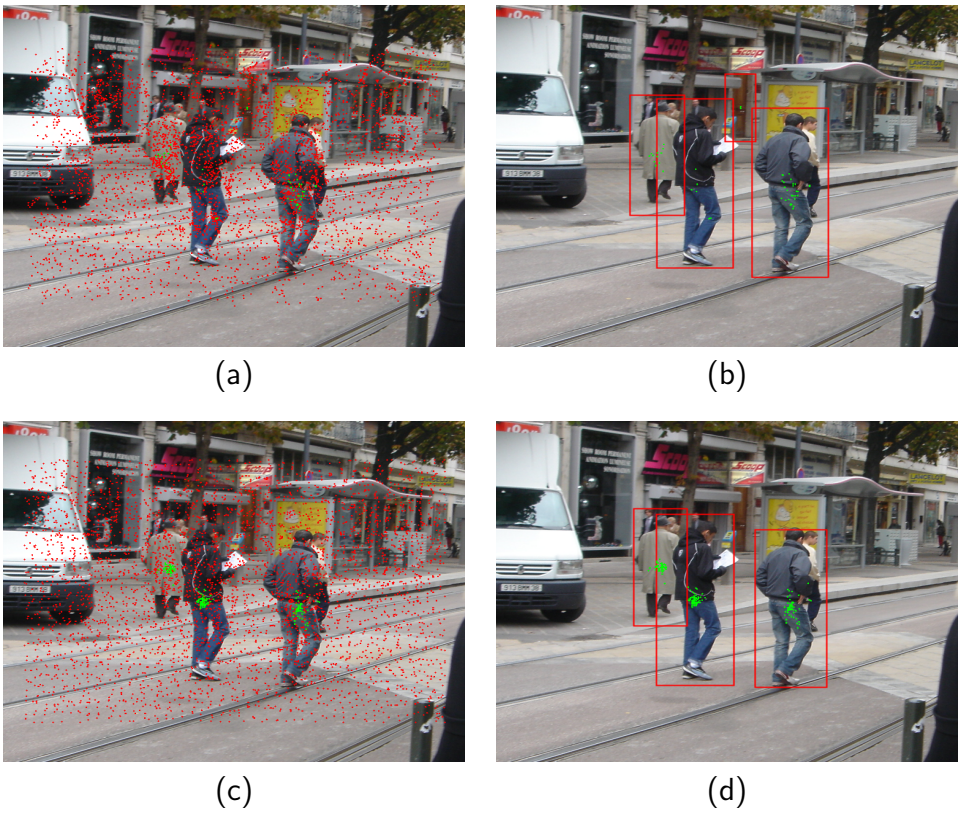


Figure 2.6: The qualitative result of sampling the same number of windows. Top row: MS-PW. Bottom row: Our method. Left column: The center points of the windows selected by each method (both red and green dots). Right column: The center points of positive classified windows sampled (green dots).

bottom. It is evident that our method can discover more positive windows and achieve higher detection rate than MS-PW.

An intuitive explanation of these results would come from the qualitative comparison in Fig. 2.6, where our method exhibits better performance in locating windows containing targets when sampling the same number of windows and classifying with the same threshold.

Meanwhile, Fig. 2.7 demonstrates the average precision rate of system's performance in retrieving targets from images under different sampling budget. Although the average precision of MS-PW method increases along with the sampling bud-

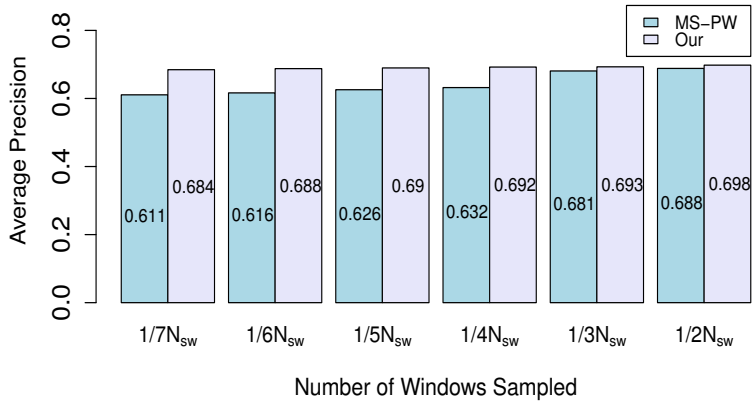


Figure 2.7: Average precision rate of two methods with same window budget

get, our method remains favorable because of better performance for all budgets. More interestingly, our method could hold a relatively high average precision rate when the budget number is small. This suggests our approach properly exploits the detector's response pattern and facilitates sample efficiency.

In the last experiment, we examine system's detection performance using *Detection Error Tradeoff (DET)* curves, which represent how missing rate (1 - detection rate) changes with the false positive rate per image (FPPI). Performance using sliding window method with N_{sw} budget windows (scanning step = 8) is also shown as a baseline. Results in Fig. 2.8 reveal similar DET curves when we set windows budgets for our method and MS-PW as $1/7N_{sw}$ and $1/3N_{sw}$. The results mean that to achieve the same detection performance with the sliding windows method, our method only uses 1/7 of the total windows which outperforms MS-PW that needs 1/3.

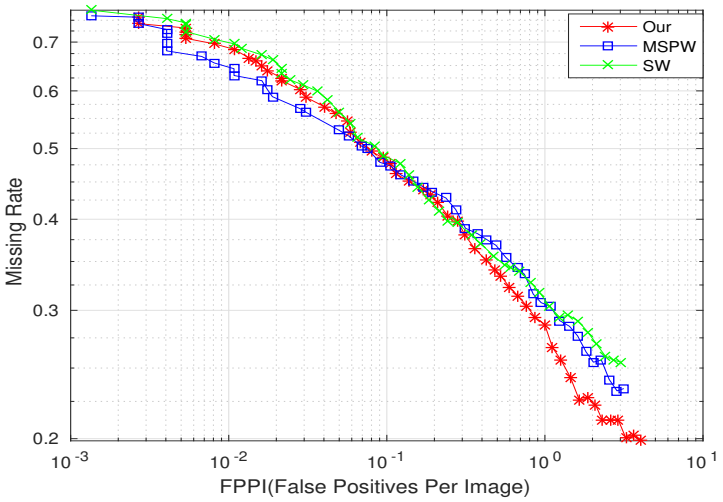


Figure 2.8: DET curve of MSPW, SW and our method

2.6 Conclusions And Future Directions

In this work, we present a method of active sampling with response pattern to detect targets efficiently in a visual image. The proposed method exploits the detector's response pattern to avoid an expensive, exhaustive searching for targets. An exponential decay function is used to model the pattern of detection score in the positive classification region. By comparing the predicted response score and the observed one, we estimate the probability of an unobserved window containing targets and having locally maximum response. Based on that, posterior sampling is applied to decide the next window to observe. Experimental results on human detection show that our approach can achieve higher detection rate than the MS-PW method using the same total windows budget, and also requires less number of windows to achieve similar detection performance compared to the sliding window and MS-PW methods.

In the future, we will consider integrating this sampling method with other search space reduction algorithms such as segmentation or saliency-based image processing techniques to achieve better target detection performance. Also, we may investigate other action policy strategies such as information-directed sampling [21], so that we can further incorporate the potential information gain of sampling each window into our reward evaluation to improve the balance between exploitation and exploration during detection iterations.

Chapter 3: Fast Task-Specific Target Detection via Graph Based Constraints Representation and Checking

3.1 abstract

In this work, we present a fast target detection framework for real-world robotics applications. Considering that an intelligent agent attends to a task-specific object target during execution, our goal is to detect the object efficiently. We propose the concept of early recognition, which influences the candidate proposal process to achieve fast and reliable detection performance. To check the target constraints efficiently, we put forward a novel policy to generate a sub-optimal checking order, and prove that it has bounded time cost compared to the optimal checking sequence, which is not achievable in polynomial time. Experiments on two different scenarios: 1) rigid object and 2) non-rigid body part detection validate our pipeline. To show that our method is widely applicable, we further present a human-robot interaction system based on our non-rigid body part detection.

3.2 abstract

3.2.1 Introduction

When robotics researchers address applications that require visual perception to allow for interaction with the environment, they usually adopt Computer Vision techniques. However, the state-of-the-art Computer Vision pipelines are not well suited for autonomous robotics. Take as an example the object recognition pipeline. Most recent approaches rely on a general object candidate proposal procedure to generate regions (both RGB or RGB-D), which likely contain objects. After this object proposal stage, pre-trained classifiers, such as pre-trained Convolutional Neural Nets (CNN), evaluate each candidate's region and determine whether the region contains one of the target objects [15, 22].

The above pipelines are considered effective and efficient for Multimedia applications, such as image tagging and retrieval. However, they are not directly applicable for Robotics applications. The reason is that during the execution of a task or a particular phase of the task, the robot needs to localize only the task-specific object in a fast and reliable fashion. For example, while programming a humanoid robot to open a microwave, only the microwave's exact pose and handle location are critical for successful execution, while other objects that happen to be in the scene can either be ignored or simply represented as generic geometric objects, such as boxes or cylinders, for collision check.

Thus, the general object recognition pipeline based on object candidate pro-

posals becomes redundant, due to two reasons: 1) before executing a task, the robot is aware of what object to focus on from task description; 2) the traditional object recognition pipeline, which considers the general situation without specific task, will hurt the system's overall detection performance.

Here, we present a novel strategy to tackle the object recognition problem in a robotic manipulation setting. We propose to consider the constraints from the target object already during the candidate proposal process in order to speed up the task-specific object detection during robotic execution. However, the main technical difficulty of the new pipeline is due to the vast amount of various constraints for real world objects. Let's consider the underlying distribution of the total real world target objects, each detection constraint shall contribute differently to target localization. In this work, we formulate the problem as a filtering problem and by achieving a sub-optimal order of constraints to check, our system is able to reject the negative instances early and thus significantly reduce the amount of time for target detection.

We summarize our contribution as follows:

1. We demonstrate the feasibility and benefits of introducing target descriptions early into the segmentation and object candidate proposal procedure for robotic applications.

2. The process of checking a target's constraints is formulated as a shared filter problem, and we prove that a greedy strategy of organizing constraint filters has a bounded performance with regard to the optimal checking sequence.

The optimized order can be interpreted intuitively as a task-specific attention mechanism under the current working conditions.

3. We implement and apply the presented framework to two different real world scenarios: the detection of drawers with handles and the detection of hands and the arms. The experimental results show that: 1) the optimized constraints checking order is time-efficient; 2) our detection framework is general enough to deal with both rigid objects and deformable objects.

3.3 Related Work

Object detection and recognition is a problem widely studied within the Computer Vision and Robotics communities. Various object detection pipelines have been proposed for different contexts and different applications.

Object candidate proposal followed by classification has become a dominant procedure for object detection. First, proto-objects or possible object areas are generated either by segmentation [23, 24] or searching [2, 10] using low-level visual cues. High-level knowledge such as context [14,25,26] and bio-inspired attention [13] can be added to help reduce the number of candidates and make the search more efficient. After pruning the search space, features [19] and attributes [27] can be extracted and classified by one or multiple statistical models [28]. Recently, deep neural network based approaches [15, 29] became popular due to their performance and their way of handling features and classification simultaneously. However, a general candidate proposal approach is not suited well to deliver a target-specific task

for a robot. The traditional detection pipeline would be computationally redundant, given the potentially large number of object candidates.

Another class of methods widely adopted in robotics applications employs keypoints [30] and model matching [31]. Especially when depth is available, 3D descriptors [32, 33] can encode the shape, and they perform well when considering them in conjunction with color [34]. However, though dealing with specific object instances, these methods spend a significant amount of resources on finding the key points. Also, by storing the complete 3D model and comprehensive views, the detection process is redundant and difficult to generalize.

Here, we propose the concept of early recognition, which influences the candidate proposal process to achieve a fast and reliable target detection performance. In our framework, the target object is described as a graph, and visual cues like attributes are treated as the constraints to be followed by the graph elements. In image processing, graph related models like Markov Random Fields have been used to recognize or segment the target [35]. Previous approach, however, focus on the recognition accuracy and thereby require a full list of attributes. In our framework, we present a novel way to speed up the detection process by optimizing the order of the visual constraints to check. Similar to algorithms in data mining [36, 37], we adopt a greedy algorithm in ordering the visual constraints. Moreover, we provide a theoretical foundation for our approach by proving its submodular property [38].

3.4 Our Approach

We illustrate the system's workflow in Figure 3.1. Given an input RGB-D image I (Fig. 3.1(a)), the system first generates a scene graph $G = \{V, E\}$ by segmenting I into surfaces V (Fig. 3.1(b)). At first, E contains all the possible connections, and G is a fully-connected graph. We then represent the knowledge about the target object as a set of template graphs $\mathbb{GT}$, with constraint functions associated with each vertex and edge. During the main detection procedure, our system checks sequentially the constraints provided by the description of the target object $\mathbb{GT}$ to remove negative matches between the scene and the template graph (Fig. 3.1(c)). In the end, the system returns the subgraphs satisfying all the target constraints, which provides the target object candidates (Fig. 3.1(d)).

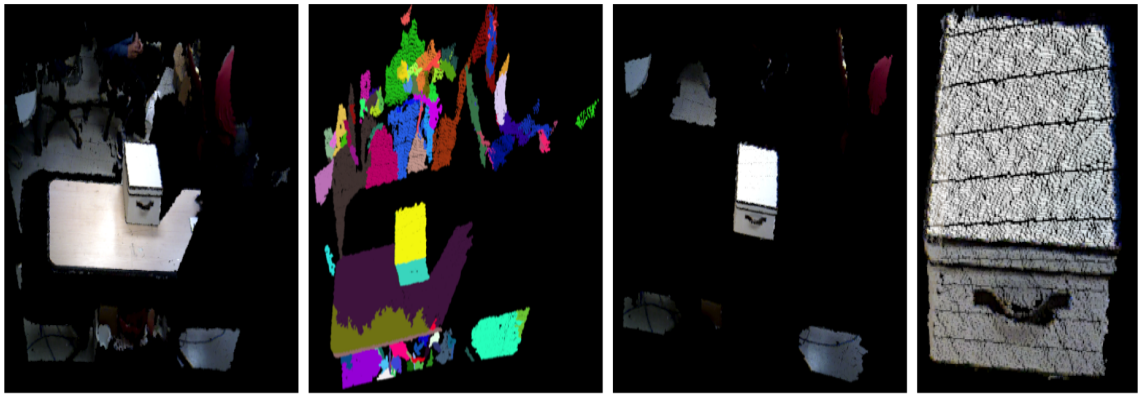


Figure 3.1: Detection pipeline. (a) Input image; (b) Scene graph after segmentation; (c) Remaining scene graph nodes with more than 1 active match with the template graph after constraints checking; (d) Detection output after matching template graphs.

Since our system considers the constraints from the target object early in the process, the procedure of finding candidate proposals becomes target-specific, and

the recognition phase becomes a part of the constraints checking. Here, the task of efficient target detection can be formulated as "how to find the target (a subgraph that matches the template graph) from the segmented input image (scene graph) efficiently."

3.5 Problem Formulation

The first stage of our pipeline segments input image I and generates a scene graph $G = \{V, E\}$ accordingly. Here, V is the set of segmented surfaces, and E represents the relationship between surfaces.

As mentioned before, we describe the target object as a set of template graphs $\mathbb{GT}$ along with a set of constraint functions. $\mathbb{GT} = \{G_l | l = 1, 2, ..., N\}$ and $G_l = \{V_l, F^{V_l}, E_l, F^{E_l}\}$, where V_l denotes the nodes (surfaces) in l-th template graph, $F^{V_l} = \cup_{v \in V_l}\{F^v\}$ represents the set of vertex constraints F^v for each node $v \in V_l$. The result of matching each constraint F with vertex v is a random variable F_v with values of $\{false, true\}$. Here, value $true$ means a constraint is satisfied. If F is evaluated to be $false$, the matching to the template vertex v will be rejected. In our case, one constraint can be used to match with different vertices from the template graphs, i.e. it is possible to have $F \in F^{v1}$ and the same $F \in F^{v2}$, where $v1 \neq v2$.

E_l is the set of edges in i-th template graph. $F^{E_l} = \cup_{e \in E_l}\{F^e\}$ are the edge constraints for all of them. It is worth noting that the nodes in different template graphs have the same index if both their nodes and edges constraints are identical

$V_T = \cup_l V_l$ and $E_T = \cup_l E_l$ denote all the node and edge labels in the template graph. $\mathbb{F} = \{\cup_{v \in V_T} F^v)\} \cup \{\cup_{e \in E_T} F^e\}$ represent all the constraints associated with nodes and edges.

Given a scene graph of an input image, $G = \{V, E\}$, we want to find all the subgraphs of G which match with one of the templates in $\mathbb{GT}$ efficiently:

$$\begin{aligned} &\text{minimize} \quad c_{avg}(I) \\ &\text{subject to} \quad I \in \text{Permutation}(1, 2, ..., |\mathbb{F}|), \end{aligned} \tag{3.1}$$

where I denotes one of the constraint checking sequence. $c_{avg}(I)$ denotes the expected cost of checking the constraints following the order of I. Here the cost originates from the temporal ordering, because the goal of our system is to detect the target object as fast as possible.

A naive approach of searching template graphs in G is to check all the constraints $\mathbb{F}$ in random order. The downside of such an approach is obvious. It neglects the cost for checking the constraints.

On the other hand, searching for the optimal order is computationally expensive because of the potentially exponential number of possible graph matches. The computation complexity to exhaustively test each of the constraint checking orders is non-polynomial. Thus, a computationally affordable strategy for determining the constraints checking sequence is desirable for efficient target search. In the following sections, we will introduce our take. Our system's output is a sub-optimal constraint checking order. The experimental results show that our approach is able to significantly reduce the time for target detection for robotic applications.

3.6 Constraints Order Searching Policy

In this section, we first clarify the constraints' checking procedure, then put forward our constraints order searching algorithm. Finally, we prove that the checking order determined by our policy holds theoretical performance guarantee under a set of assumptions.

For the sake of clarity, we introduce $A(s|\mathbb{F}')$ to be the set of possible template graph node labels that a scene node s matches, after checking the set of constraints in $\mathbb{F}' \subseteq \mathbb{F}$. Intuitively, s could match with template graph node v if all the node and edge constraints in $\mathbb{F}'$ associated with node v are satisfied.

We present in Algorithm 3 the procedure of matching a scene graph with template graphs. In a nutshell, the algorithm checks all the template graph constraints in $\mathbb{F}$ for each vertex or edge in the scene graph G. After filtering out the negative matches between the scene and template graph, our system returns all the subgraphs from the scene graph whose corresponding matched nodes and edges form one of the template graphs.

In our use case application, one legitimate assumption is that the number of possible matches for each scene vertex, after checking all the constraints in $\mathbb{F}$, is limited. Thus, returning the remaining subgraphs (Algorithm 3 line 16) is expected to take a reasonable amount of time. Traditional searching algorithms such as depth-first search and breath-first search are also expected to deliver decent performance. Here, we treat the constraints associated with the graph edges like the node constraints, Thus a matching is determined by whether the scene vertices pass through

Algorithm 3: The Procedure of Matching the Scene's Subgraphs to Template Graphs

Parameters:

A set of template graphs $\mathbb{GT}$;

An ordered list of constraints to check: I

Input:

Scene graph G.

Output:

A set of subgraphs of G matching to one of the template graphs.

1: Initialize the observation set $\mathbb{F}_{ob} = \emptyset$,
Every node can match to any template graph node at the beginning:
$A(s|\mathbb{F}_{ob}) = V_T \ \forall s \in V$,
2: **for** $i = 1, 2, ..., |I|$ **do**
3: Denote $R(F_{I_i}) = \{v \in V_T | F_{I_i} \in F^v\}$
4: **for all** $v_r \in R(F_{I_i})$ **do**
5: **for all** $s \in G$ **do**
6: **if** $v_r \in A(s|\mathbb{F}_{ob})$ **then**
7: Check constraint F_{I_i} on vertex s if F_{I_i} is a node constraints. Or check on s's edges if it is an edge constraint.
8: **if** false **then**
9: Update $A(s|\mathbb{F}_{ob}) = A(s|\mathbb{F}_{ob}) / \{v_r\}$
10: **end if**
11: **end if**
12: **end for**
13: **end for**
14: $\mathbb{F}_{ob} = \mathbb{F}_{ob} \cup \{F_{I_i}\}$
15: **end for**
16: Return all the subgraphs matching one of the template graphs.

the filter (or satisfy the constraints) of the template node.

It is not hard to notice that in Algorithm 3, the checking order of constraints (line 2 - 15 in Algorithm 3) influences the overall processing time, though it does not alter each node's final matching output $A(s|\mathbb{F})$. Intuitively, if a constraint can exclude a large portion of scene vertices from matching template graph vertices using low temporal cost, then the computational cost of the following constraints checking (with higher computational cost) is expected to be reduced significantly. In other words, the order of the constraints to check matters.

To formulate the ordering problem, let us denote c_i as the cost of checking constraint F_i on a scene vertex, and $P(F_{I_i}|I)$ as the probability that constraint F_{I_i} needs to be checked following the checking order of I. Intuitively, we aim to minimize the expected cost of checking a scene node s following constraints checking order I:

$$\begin{aligned} &\text{minimize} \quad \sum_{t=1}^{|\mathbb{F}|} c_{I_i} P(F_{I_i}|I) \\ &\text{subject to} \quad I \in \text{Permutation}(1, 2, ..., |\mathbb{F}|) \end{aligned} \tag{3.2}$$

The optimization formulation can also be interpreted as a dual problem: given a cost budget, minimize the possible matches between the vertices of the scene and the template graph. Here we assume that the cost of checking each constraint is static and independent from other constraints.

First consider a special case, where there is only one vertex in the template graph. Without loss of generality, let us denote $V_T = \{v\}$, and $\mathbb{F} = F^v$.

Lemma 3.6.1. *Define a greedy constraint checking order IG:*

$$IG(k) = argmax \tag{3.3}$$

$$\begin{cases} \frac{P(F_i=false)}{c_i} & if\ k=1, \\ \frac{P(F_i=false|F_{IG_1}=true,\ldots,F_{IG_{k-1}}=true)}{c_i} & o.w. \end{cases} \tag{3.4}$$

If the conditional filtering effect of each constraint in F^v is non-increasing, i.e. if index set $A_1 \subseteq A_2$ and $\forall j \notin A_2, P(F_j = false|F_i = true, i \in A_2) \leq P(F_j = false|F_i = true, i \in A_1)$, then the expected cost of checking if a scene graph vertex matches with v following the checking order of IG is the minimum among all the static sequences.

Proof. If a scene node can match to a template node v, then all the constraints in $\mathbb{F}$ are satisfied. Thereby all possible permutations have the same checking cost: $\sum_{i\in\mathbb{F}} c_i$, because the algorithm checks each scene vertex with all the constraints. So we want to find a sequence to minimize the expected cost of rejecting matching a scene node to v.

Define an objective function $g : 2^{|\mathbb{F}|} \times |2|^{|\mathbb{F}|} \rightarrow R^+$ as

$$g(A, O(A)) = \begin{cases} 1 & \text{if } \exists i \in A, F_i = false, \\ 0 & \text{o.w,} \end{cases} \tag{3.5}$$

where $O(A)$ is the constraints checking result of constraints indexed by A. We assume that at least one of the constraints in $\mathbb{F}$ returns false because we are dealing with a non-matching case. Here, since we are considering a special case of a single

node in the template graph, our goal is to minimize the expected cost of constraints checking when g reaches value 1. We have:

(1) g is strong adaptive non-decreasing: $\forall A \subseteq \mathbb{F}$ and for all possible corresponding observations $O(A)$, $g(A, O(A)) \leq g(A \cup \{j\}, O(A) \cup \{F_j = o\})$ $\forall o \in \{true, false\}, \forall j \notin A$. It means that the objective function value does not decrease with more observations coming in.

(2) g is adaptive submodular: $\forall A_1, A_2$, s.t. $A_1 \subseteq A_2, O(A_1) \subseteq O(A_2), \forall j \notin A_2$,

$$\begin{aligned} &\mathbb{E}[g(A_2 \cup \{j\})|O(A_2)] - g(A_2|O(A_2)) \leq \\ &\mathbb{E}[g(A_1 \cup \{j\})|O(A_1)] - g(A_1|O(A_1)) \end{aligned} \tag{3.6}$$

Intuitively, this means that the marginal gain of the objective function g is non-increasing.

Here is our proof. If one of the constraints in A_2 returns false, the left-hand side of Eq. (3.6) is 0 since the matching has been rejected, while the right side of the inequality can be 0 or 1. When none of the elements in A_2 returns false, based on the assumption in the lemma, that the conditional filtering effect is non-increasing, (3.6) still holds.

(3) g is self-certifying: we know immediately once g reaches value 1 based on the current observations. Because we are dealing with the case that a scene node will be rejected, our observation space does not contain non-zero possibility events of passing all the constraints checking. So, based on Proposition 9 in [38], function g is a self-certifying instance.

Based on Theorem 11 in [38], when g reaches 1, the average cost of greedy

sequence IG is smaller than $(1 + ln(\frac{Q}{\eta}))$ times optimal time cost. However, in our case, we have value $Q = 1$, and $\eta = 1$, so the cost of IG is equal to the optimal.

A theorem of adaptive strategy is adopted to prove Lemma 1 for our static sequence, because we are dealing with a special case of single label matching ($Q = 1$). If the adaptive strategy continues, it implies that all the observed constraints return true. Under such a scenario, both static and adaptive sequences are the same.

□

Theorem 3.6.2. *Assume the constraint's conditional filtering effect for the same template vertex is non-increasing (as defined in Lemma 3.6.1) and constraints belonging to different template vertices are independent. i.e. for $i \neq j$, if $\nexists v \in V_T$, s.t. $F_i \in F^v and F_j \in F^v$, then $F_i \perp\!\!\!\perp F_j$. Then the cost of a greedy constraint checking order will be upper bounded by μ times optimal cost, where μ is the maximum number of template vertices that have the same constraint.*

The proof is similar to the proof of Theorem 3.4 in [36]. The idea is that for any single template vertex, the expected cost of the optimal sequence should be at least as large as the one returned by a greedy policy, as proved in Lemma 3.6.1. Since one constraint can appear at most μ times for different template vertices, the cost of the greedy strategy can be at most μ times that of the optimal strategy.

Furthermore, under an arbitrary distribution of constraint responses, the cost of greedy sequence checking is still bounded.

Theorem 3.6.3. *For any distributions of constraints in $\mathbb{F}$, the average time cost of checking constraints with the greedy sequence method is bounded by 4μ times*

optimal average cost. μ is the maximum number of template vertices sharing the same constraint.

Proof. Similar to the proof of Theorem 3.6.2, we start with the expected cost of a single template vertex case, then extend it to the general case.

Based on Theorem 2.3 in [37], the average cost of checking constraints following the greedy policy is at most 4 times the optimal cost.

Thereby, when multiple template graph vertices exist and at most μ nodes share the same constraint in $\mathbb{F}$, the greedy sequence checking order is at most 4μ times the optimal expected cost.

□

Our constraints' order determination policy is listed in Algorithm 4. The Algorithm 4 has a time complexity of $O(n^2)$ where n is the cardinality of $\mathbb{F}$.

Algorithm 4: Determine the Constraints Checking Order

Input:
The set of constraints: $\mathbb{F}$.
The cost of checking constraint $F \in \mathbb{F}$ for one node $c(F)$
The distribution of constraints checking results

Output:
The sequence of the constraints to check: $I \in Permutation(1, ..., |\mathbb{F}|)$.

Initialize observed set of constraints: $\mathbb{F}_{ob} = \emptyset$,
ordered list I = empty queue.
while $\mathbb{F}$ is not empty **do**
  for all $F \in \mathbb{F}$ **do**
    Compute $h(F) \triangleq P(F = false | \forall F' \in \mathbb{F}, F' = true)/c(F)$
  end for
  Select $F^* \in argmax(h(F))$.
  $\mathbb{F}_{ob} = \mathbb{F}_{ob} \cup \{F^*\}$.
  I enqueue F^*.
  Remove F^* from set $\mathbb{F}$.
end while

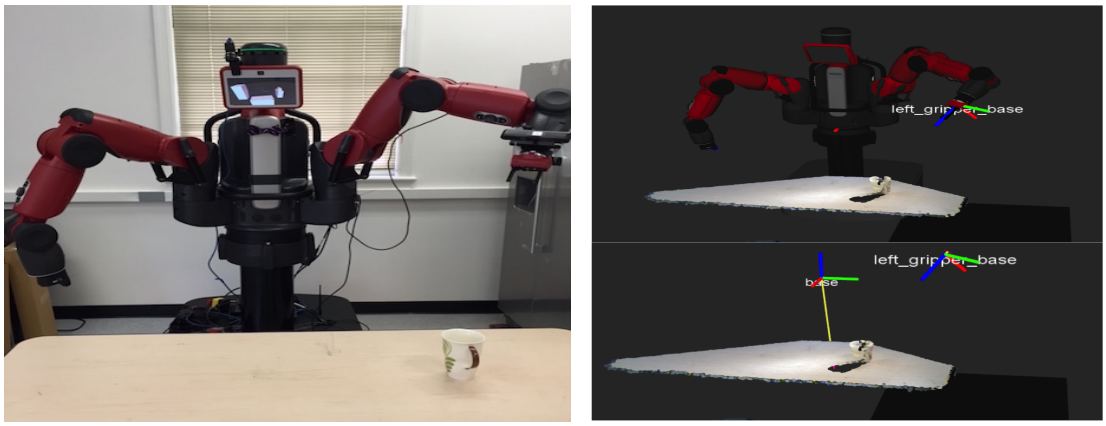
3.7 Experiments

We apply our target detection framework to two different real-world robotic tasks to validate its generality and effectiveness. The first scenario is to detect a drawer with a handle as shown in Fig 3.1. Its shape is a cuboid with a handle on the front surface. The second case is to detect a human hand in a pointing gesture. Hand localization is of great interest in the field of Human-Robot Interaction (HRI). In our scenario, we consider the hand together with the arm a single target object. This makes the task difficult, because the hand and the arm together no longer form a rigid object. Experimental results show that our framework can still work as long as the target object can be represented as a template graph.

In the phase of determining the constraint checking order, we use $\frac{n_F}{N}$ to approximate $h(F)$ in Algorithm 4, where N denotes the running times of constraint F, and n_F are the rejected matches between scene and template nodes.

3.7.1 Experimental Setup

As shown in Figure 3.2 (a), we mount an ASUS Xtion PRO camera to the left wrist of a Baxter humanoid robot. Our system maintains and provides transforms between the Baxter base frame and other joints. Since the camera's pose is fixed to the wrist, we calibrate the camera's coordinate to the "left_gripper_base" frame. By propagating the tf (transform) tree, the system projects the point cloud data from the ASUS camera into the robot base frame, which enforces the z-axis to point upwards and x-axis to face forwards (Figure 3.2 (b)).



(a) System outlook (b) Illustration of coordinate system

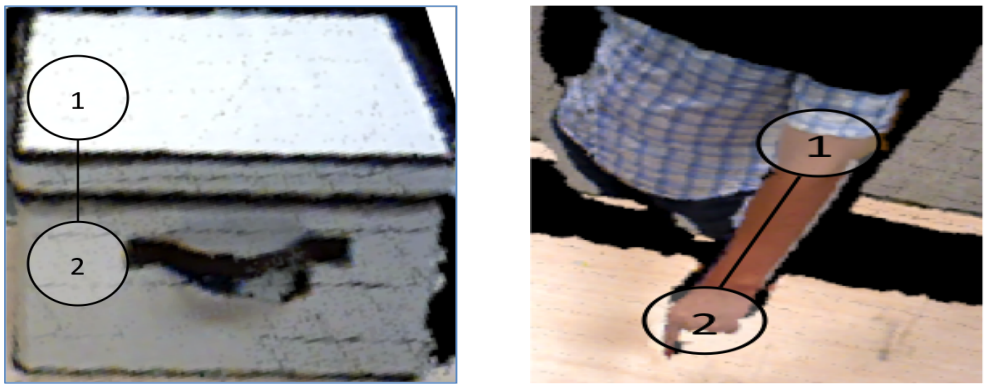
Figure 3.2: Robot and camera setup

3.7.2 Segmentation

An input RGB-D image from the ASUS camera is over-segmented into surfaces to generate a scene graph. In our implementation, we apply the plane-fitting algorithm from [23], which uses depth-adaptive normal calculations and plane fitting taking into account the noise from the depth measurements. Since the adaptive operations are based on the assumption that the z-axis value is the depth value, we calculate the surface normal and fit a plane in the camera's original frame ("camera_link") before transforming the measurements to Baxter's base frame.

3.7.3 Handle Drawer Detection

As shown in Figure 3.1, a handle drawer has the shape of a box with a horizontal handle on one of its surfaces. Depending on the viewpoint, two (top, front) or three (top, front, and side) surfaces of the box are visible. In our implementation, while maintaining a high success rate, we model the template graph of the drawer as



(a) Handle box (b) Hand with arm

Figure 3.3: Illustration of template graphs

two nodes. Figure 3.3 (a) shows the template graph. The constraints of the handle drawer template graph are listed in table 3.1.

Template component	Constraints
Node 1	Size($node_1$), Orientation($node_1$)
Node 2	Size($node_2$), Orientation($node_2$), Has_handle
Edge $(1, 2)$	Pairwise_vertical, Convex_box

Table 3.1: Constraints for handle box detection

We check the size constraint by comparing the first two principal components of the surfaces with target-specific thresholds. For example, because node 1 of the handle box is a rectangular surface, we restrict the first principal component (length) to be within $(0.3, 0.7)$ meters and the second dimension (width) to be in the range $(0.25, 0.6)$. For the orientation constraint we check whether a surface's normal (third principal component) is along a particular direction. Here, node 1 in the handle drawer graph is upward while node 2's direction aligns with the horizontal plane. The checking of the surface's handle constraint is done in two steps. First, we extract points within a 3D bounding box in front of the surface patch. Then we

validate if there is a connected component on that surface that has the shape of a handle, i.e. satisfying the size constraint of a handle.

For the two edge constraints, the pairwise vertical constraint returns true if the two surfaces are adjacent to each other, and their orientations are perpendicular to each other. The convex box constraint checking is done by checking if two surfaces not spanning the same plane form a convex shape, which means the two surfaces shall not segment each other, and the shape they form must form a convex box instead of a concave corner, as illustrated in Figure 3.4.

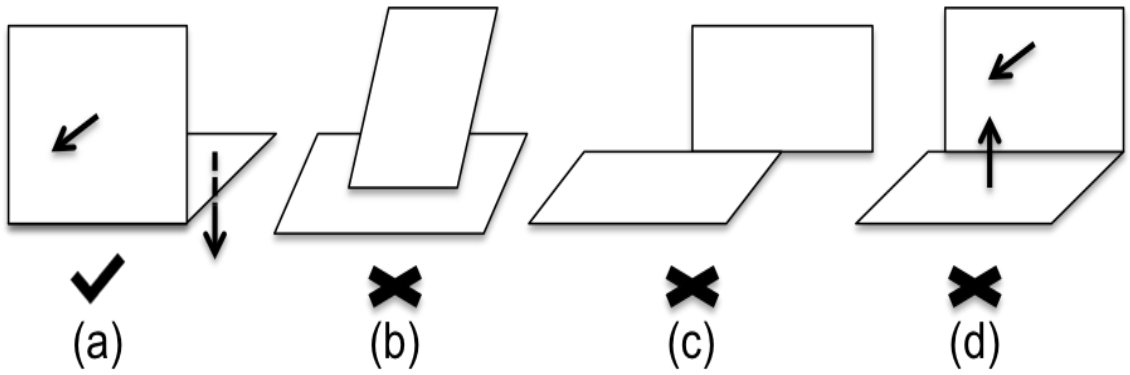


Figure 3.4: (a) Illustration of a convex box. Three failure cases: (b) top surface segments the bottom one; (c) not enough shared boundary; (d) two surfaces form a corner (faces towards)

3.7.4 Hand Pointing with Arm Detection

As second scenario we consider hand detection. Here we show an application of detecting the hand together with part of the part of arm visible in the scene. Figures 3.3 (b) and 3.7 show the experimental setup. Note that the target object is no longer rigid in this situation because the angle between the finger and the wrist is flexible during pointing.

As shown in Figure 3.3 (b), we use one node to denote the arm in the template graph and the other for the hand. The constraints for checking template graphs are listed in Table 3.2.

Template component	Constraints
Node 1	Size($node_1$), Location($node_1$)
Node 2	Location($node_2$), size($node_2$), Hand_shape
Edge $(1, 2)$	Hand_arm_relationship

Table 3.2: Constraints for pointing hand detection.

The size constraint is handled in the same way as the handle drawer detection except that the size thresholds need to be set for surfaces that belong to the hand and the arm. The location constraint returns true if the centroid of a scene surface is in a given cuboid area. Since we have already transformed the point cloud to the Baxter's base frame, which aligns well with human perception, it is not hard for people to manually annotate a 3d range of possible locations of the target. For example, because we do not expect to see the arm or the hand on the ground or flying high around the ceiling in this scenario, we can set the location threshold on the z-axis to reject surfaces heights that are too large or too small. The hand shape constraint checks if the contour of a surface patch has the shape of a pointing hand.

The hand-arm relationship is encoded as edge relationship between the hand and the arm. We check if there is a hand node close to the arm node and enforce the constraint that the the hand is along the direction of the arm.

3.7.5 Optimization of the constraints checking order

As discussed in Section 3.6, the order of checking the constraints influences the time for target detection, and we proved that the greedy constraint ordering algorithm (algorithm 4) has a bounded computational cost w.r.t the optimal computational cost.

To show the efficiency of the presented algorithm, we compare the average computational time for constraint checking (line 2-15 in algorithm 3) of four different checking order policies: random checking order, best order among sampling 300 sequences, the greedy order proposed, and the optimal order.

Random ordering does not need any training data and it represents the most naive policy. The time cost of random order is the mean of the running time of 300 random checking sequences. Order determined by sampling is the one with the minimum running time among 300 random checking sequences on the training data. The optimal order is obtained by exhaustively trying all possible sequences and then selecting the sequence with minimal cost as the optimal one. The handle drawer has $7! = 5040$ possible sequences and the hand detection has $6! = 720$. Note that an exhaustive search for the optimal sequences is not feasible because the number of permutations grows exponentially when the number of constraints increases.

For both handle box and hand detection, we collected 50 point clouds each for training and testing. Also, another 100 background point clouds were collected to serve as negative samples with half used in training and half in testing. The result is shown in table 3.3.

time (ms)	Random	Sampling	Algorithm 4	Optimal
Handle box	382.6	28.0	24.6	19.3
Pointing hand	43.2	8.1	8.8	8.1

Table 3.3: Running time of four constraint checking order on testing data.

The machine used in the experiments has an Intel i7-6700 CPU of 3.4GHZ and the memory is 16 GB. We did not use GPU or parallel computing.

From the result, we can see that a random ordering without any optimization would take the longest time to execute. Our proposed greedy algorithm is close to the optimal order. When the total number of possible sequences is not large, a sampling method would perform better than our method as is shown by the case of the pointing hand. But when the possible number of sequences is large so that sampling cannot cover a reasonable portion, our algorithm works better as is shown for the case of the handle box. Also note that to determine a checking order, algorithm 4 is much faster than the sampling method because algorithm 4 only needs 49 and 36 sequence checking runs while sampling needs 300 in our experiment.

3.8 A live HRI application

In this section we describe an application of human-robot interaction that is built on the pointing hand detection discussed in section 3.7.

Our system allows humans to interact with real world objects through pointing gesture, and it then generates a command for the robot accordingly. For instance, let us suppose that a user intends to heat an object using the microwave or put an object into the refrigerator. As shown in Figure 3.7, using our system, a user can

select the target object using a pointing gesture. After the selection is confirmed, the target object can be virtually dragged to its target location through human guidance.

After applying the constraint checking order optimization, our hand detection works faster (than a naive approach) thence our system provides a smoother interaction user experience. To complete the whole scenario, we introduce other components of the interaction system aside from hand detection in the following sections. We provide a video showing the interaction process in the supplementary material.

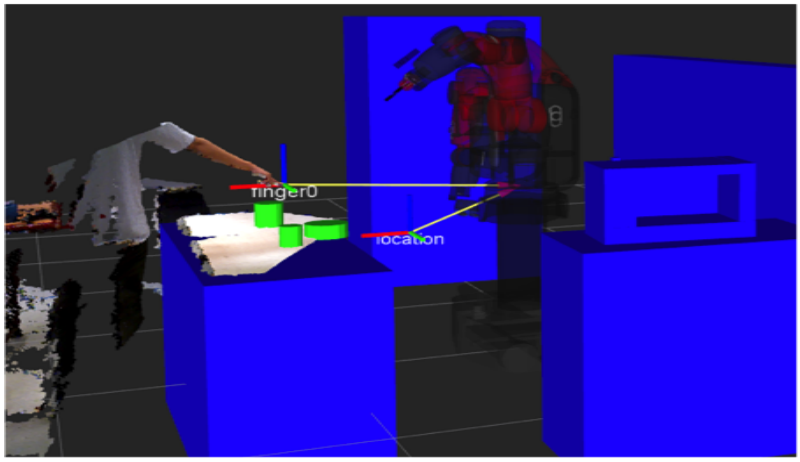


Figure 3.5: Visualization of the fingertip and its pointing location.

3.8.1 Surrounding objects recognition

Large surrounding objects, such as a fridge, a table and the shelf holding the microwave can be detected at the beginning of the process and are kept in a stored world model since they will be static for a long time. The table top objects can be detected and recognized online by a tabletop point cloud reconstruction,

segmentation [24] and recognition. The recognition outputs are stored in the same world model describing the real world configurations. It is worth mentioning that our pipeline is applicable for both the tabletop objects and detection of other objects. For example, the microwave in Figure 3.7 is detected using the same pipeline.

3.8.2 Obtaining the object that the hand is pointing at and its location

We use a straight line to represent the pointing direction after detecting the pointing hand and arm. The direction of the line is the detected arm's direction, which is calculated by taking the eigenvector corresponding to the first principal component of the point cloud belonging to the arm. The fingertip are at the starting point of the straight line. We detect the fingertips by searching the 3D points on the hand that are furthest along the pointing direction. Figure 3.5 shows a visualization of fingertip point.

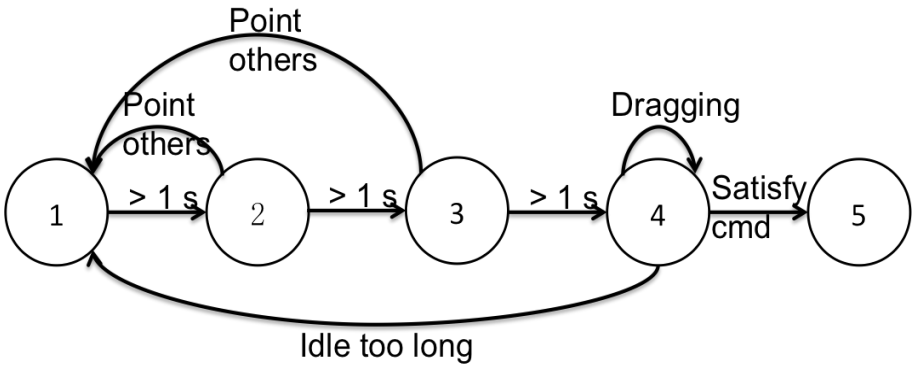


Figure 3.6: An illustration of the state machine for human-robot interaction. State 1: initial state; state 2,3: state for object pointing confirmation; state 4: object virtual moving state; state 5:ending state after parsing the command successfully.

During the object selection phase, we simply treat the object closest to the

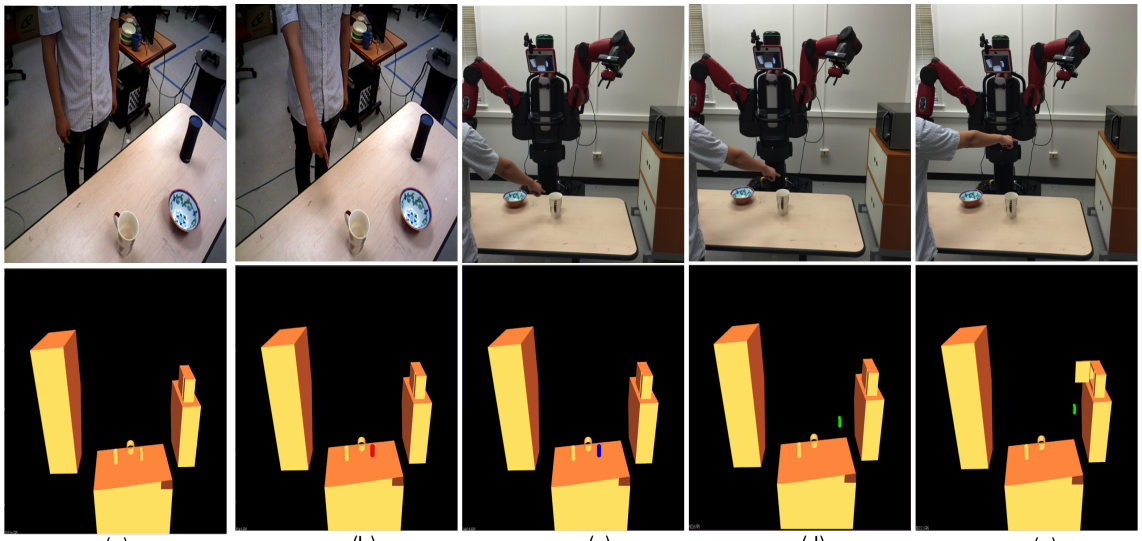


Figure 3.7: Sending a command: <heat the mug> via HRI system. The top row shows the human operation and the bottom row shows the corresponding status of the interaction process. (a) Initial stage with objects detected in the scene; (b)(c) select target object by pointing and then object confirmation; (d) drag the selected object to another functional place, i.e. microwave, virtually; (e) the system receives the command.

pointing line as the target object of the user. Also a threshold is used to limit the distance between the object and the pointing ray.

After an object is confirmed, the user can start moving it around by pointing to other locations in our virtual environment. At this time, we treat the intersection point between the pointing ray and a plane spanning table top as the target location (Figure 3.5). Thence, the selected object's position can be updated as the new location.

3.8.3 Robot feedback interface

A proper way to display the current configuration of the world model and the status of pointing is necessary for a smooth and accurate interaction between human

and robot.

In our system, we directly show the current status of the system through the Baxter robot's screen. Specifically, we visualize the virtual world using the PCL visualizer [39].

3.8.4 Interaction flow finite state machine

The underlying logic flow is implemented as a finite state machine as shown in Figure 3.6.

At the beginning, the system is in state 1 waiting for the human's command (Figure 3.7 (a)). The system will advance to the object selection state 2 once a pointing to the object happens for more than 1 second and the selected object will turn red (Figure 3.7 (b)). After two more seconds of consistent pointing, the object is confirmed to be chosen and the system reaches state 4 (3.7 (d)). Then the object can be virtually moved around. Finally, when the object reaches a certain area of the target location for more than 1 second, the system gets a command that it was successful and enters the final state 5 (Figure 3.7(e)).

3.9 Future Work

As shown by experiments and an HRI application, our framework is able to detect target objects in robotic applications in a reliable and effective way. In future work, we plan to improve the learning pipeline by generating template graphs automatically. Additionally, with the number of types of constraints increasing, how

to select a reasonable subset is another problem that deserves further investigation.

In the application scenario outlined we make a first step to teaching the robot via pointing to the target area. Using this approach of teaching the robot with bare hands, trajectory learning and adaptation [40] could become more friendly.

Chapter 4: Active View Point Control for Reliable Target Detection in Human-Robot Interaction

Object detection is a fundamental task for robots, and it becomes more challenging during the human-robot interaction when additional randomness from the user comes in. In this chapter, we put forward a practical viewpoint control system for object detection during the human-robot interaction. We not only consider the viewing condition constraints for vision algorithms but also incorporate the low-level robot kinematics to guarantee the reachability of the desired viewpoint. By selecting viewpoints fast using a linear time cost score function, our system can deliver smooth user interaction experience. Finally, we provide a learning from human demonstration method to obtain the score function weights that better serves task's preference.

4.1 Introduction

Human-robot interaction is a critical component involved in many robotics tasks such as robot assistance, visual learning, and command sending. A visual system that can robustly recognize the objects during the process will make the interaction more reliable and is vital to task's success. For example, to reliably

detect and recognize user's hand is essential to a gesture command system. Also if a user wants to pass an object to a robot, both the object and the hand need to be clearly seen to make sure the robot can catch it.

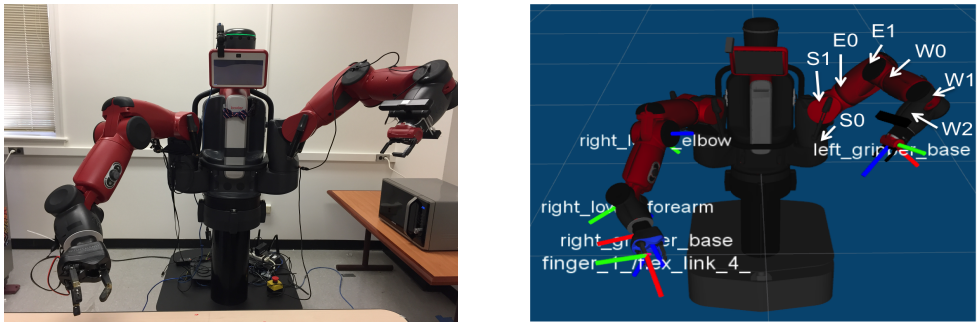
However, to reliably recognize the objects of interests is not a simple task. For example, an out-of-view problem may occur due to camera's limited view angle, human's movement and random positions. Also, the target object may get occluded during the process, especially the movement of the robot may introduce occlusions that the first viewpoint and scene settings do not have. Regarding the computer vision algorithms, a too far or twisted view may increase the sensing noise or violate the algorithms' preference, thence damage the algorithm's performance.

Many robotics vision methods [41, 42] have used active viewpoints control to handle the possible unsatisfactory view conditions in reliable object detection or recognition. But few of them consider the constraints from the low-level robot kinematics and the majority just pre-select a small amount of reachable end effector positions. Moreover, their viewpoint planning and execution process may take long, which would provide a poor user experience during the human-robot interaction process.

In this work, we propose a fast active viewpoint control strategy to solve the issues of limited view angle, occlusion, and view direction requirements. The general idea is to use a score function to select the view that fits the current object recognition task. The four scoring factors are the joint discontinuity of moving to a new viewpoint, target object's viewing centeredness, occlusion and viewing angle. To guarantee the view evaluation function have a fast response, we precompute and

store the important intermediate parameters such as extrinsic transformations and joint values from inverse kinematics.

Because different tasks may have their preference of view control strategy, and to collect training data with a human in the loop would be expensive, which increases the difficulties of obtaining appropriate parameters. We propose to learn system parameters from human's demonstration. The learning process is modeled as coactive learning. Assuming human's feedback viewpoint would better fit the task, our viewpoint module follows human's direction to update its weights. Experiments on our two human-robot interaction applications demonstrate the effectiveness.



(a) Robot system setting

(b) A visualization of robot joints and frames

Figure 4.1: Image of our robotic system settings and a visualization of joints and frames. A ReFlex hand is attached as right arm's end effector for grasping tasks.(a) We mount a depth camera onto the left arm that we can actively change the viewpoint. (b) S0-W2 denotes the seven joints of Baxter robot's left arm. In occlusion prediction, we mainly use the frames on the right arm to deduct viewing ray and body part overlap.

We summarize our contribution as follows:

1. We put forward an efficient and practical viewpoint planning procedure for object detection during the human-robot interaction process. Our method considers the low-level kinematics constraints for the feasibility and reachabil-

ity of the viewpoints besides the traditional vision constraints.

2. To make the selected view point better serve the task, we use coactive learning method to learn score function's weights, where training samples are expensive, and quantifying the view qualities is hard from the perspective of the whole task.

3. We implement two human-robot interaction applications and apply our viewpoint control module to them, demonstrating the usability of our proposed system.

4.2 Related Work

Target detection and recognition are fundamental tasks in computer vision. For the efficiency and accuracy, researchers have been working on different levels of the computer vision pipeline such as feature abstraction [10, 39, 43], context exploitation [17, 25] and multi-source fusion [27, 44].

Different with traditional computer vision, robotics vision has the potential to control the in-hand or on arm camera to a viewpoint actively that better serves the task [45, 46]. Atanasov [41] formulates the active object detection problem in a Bayesian framework and a non-myopically plan the viewpoint to minimize the sensor movement and error recognition probability. But it does not consider the kinematics constraints in moving between positions. [42] uses SIFT [43] matching and 3D shape alignment as detection techniques. The next viewpoint is chosen as a place maximizing the potential number of SIFT points detected, which is a

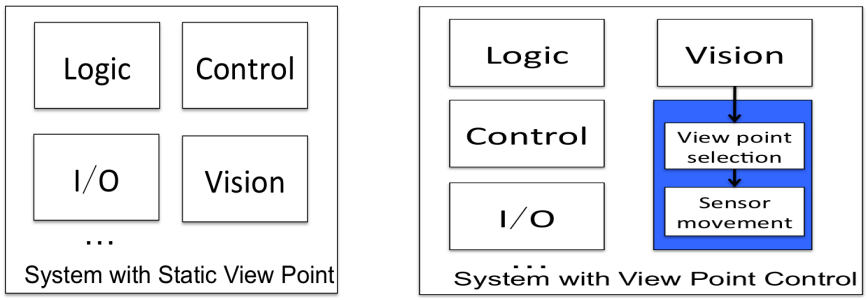
non-probabilistic approach. For tasks other than object detection, [47] considers a problem of surface reconstruction and chooses next best viewpoint by maximizing the information gain defined in terms of spatial resolution increment. With the allowance of changing scene settings, [48] uses a humanoid robot to grasp the object during the recognition process thereby avoiding the problems of occlusion and scene modeling. [49] surveyed approaches in next view planning.

However, few works have been done on active viewpoint control for the tasks of Human-Robot interaction [50], where human's random behavior would make the detection problem more challenging. [51] learns human user model with eight preset cameras. [52] develops collaborative Fetch-and-Deliver tasks via single static range sensor. But none of them deal with interaction with active cameras. In this work, we propose a fast active response viewpoint selection module for two human-robot interaction processes.

To learn viewpoint selection parameters under a rare training data situation, we present to use coactive learning method to learn from human's demonstration [53]. This learning model has been widely used in trajectory learning [40, 54], search recommendation [55], machine translation [56] etc. Our experiments demonstrate the effectiveness of coactive learning applied in viewpoint preference learning scenario.

4.3 System Overview

As illustrated in Figure 4.2, a robot system usually consists of multiple modules. Take grasping objects as an example, I/O module can receive commands from the user and vision is responsible for answering where the cup is and how it looks like. Then control module decides how to move the arm, and a logic module can organize high-level action execution order and prepare a failure plan.



(a) Static view system (b) Active view system

Figure 4.2: Illustration of static view system and active view system. The blue region in (b) is the viewpoint control module serving robot tasks.

Our work focuses on the controlling robot's visual condition actively for reliable target detection in the human-robot interaction situations, where a predefined set of camera positions would not work due to the random behavior of users.

In the following sections, we will first introduce our viewpoint control policy. Then show two applications of our viewpoint control module in passing an object to robot and gesture command sending tasks. Finally, we demonstrate how to learn the parameters in our viewpoint score function using coactive learning model.

4.4 View Point Control Module

We first detail on our active view point control pipeline.

As illustrated in Fig 4.2(b), vision algorithm's detection result will be fed to a view point selection module. This module will select a viewpoint based on current target's location and scene settings. If a new viewpoint is decided, the camera sensor will move to it.

When choosing the next view point to go, we use a score function to evaluate each viewpoint and mainly consider four types of constraints: joint discontinuity of moving to a new location, target's position in that viewpoint, occlusion conditions and viewing directions.

Formally speaking, we directly define viewpoints in the joint space with 7 degree of freedom as shown in Figure 4.1: $v \in \mathbb{V} \subset \mathbb{R}^7$. $\mathbb{V}$ is a predefined candidate set of viewpoint joint values. Using forward kinematics and coordinate transform, we can obtain the position of the camera $p \in \mathbb{R}^3$ and orientation $o \in \mathbb{R}^4$ (expressed using quaternion) under the robot base frame. $s(\cdot)$ is the score function of each viewpoint evaluation factor, ranging in $[0, 1]$. x is the current interaction scene setting, including the position of robot's arm and user's function part (e.g. hand). λ is a positive weight describing each components' contribution to the score function. Our next view point selection policy is to select a viewpoint with the minimum score:

$$\begin{aligned} v_{next} &\in \underset{v \in \mathbb{V}}{\operatorname{argmin}}\ score(v, x) \\ score(v, x) &\triangleq \lambda_{JD} s_{JD}(v, x) + \lambda_{CT} s_{CT}(v, x) + \lambda_{SO} s_{SO}(v, x) + \lambda_{VD} s_{VD}(v, x) \end{aligned} \tag{4.1}$$

Next, we are going to explain each evaluation factor.

4.4.1 Joint Discontinuity

To move the camera to the desired position in 3D space, we need to first convert the target effector's position to the robot joint space, then control each motor of the corresponding joint to the target joint value. Figure 4.1 (b) illustrates the seven joints belonging to the Baxter robot's left arm. Due to the constraints of joint values, not every point in the 3D space is reachable. Also, two adjacent points in 3D space may have a considerable distance joint space. So the Euclidean distance of end effector's between the source and destination is not a good approximation of the movement execution time.

Given $J_{curr} \in \mathbb{R}^7$ as the current joint values and $J_v \in \mathbb{R}^7$ as another view point's joint values, we define the joint discontinuity as:

$$s_{JD}(v, x) = 1 - \exp(-\frac{1}{C}(J_v - J_{curr})^T K_{JD}(J_v - J_{curr}))$$

where K_{JD} is a symmetric matrix defining joints distance. In the simple case, K can be an identity matrix and the score function $s_{JD}(\cdot)$ purely measures the Euclidean distance between two set of joint values. C is a positive constant. J_v is the same as v since we already represents view point in the joint space.

4.4.2 Centeredness of Target

Central of target constraint is to make sure the target object is in the view of the arm camera.

When checking the location of a target object in the new viewpoint, we assume the robot camera has moved to v, calculate the extrinsic transform from robot's base frame to camera's frame then project the result to 2D image using camera's intrinsic parameters, which is known before.

Given a point $p_b = (x_b, y_b, z_b)^T$ in the base frame and target view point v, we can transform p's coordinate to the camera's frame by:

$$p_c = R * p_b + T$$

where R and T is the rotation matrix and translation vector of the base frame from the camera frame.

Using pinhole camera model, we can get p_c's projection to 2D image coordinate (u, w) as:

$$\begin{pmatrix} u \\ w \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{x_c}{z_c} \\ \frac{y_c}{z_c} \\ 1 \end{pmatrix}$$

where f_x and f_y are the focal lengths in pixel units and $c = (c_u, c_w)$ is a principal point at the image center.

We then define the centeredness score as:

$$s_{CT}(v, x) = 1 - \exp(-\frac{1}{|\mathbb{P}|}\Sigma_{p\in\mathbb{P}}(p_{uw} - c)^T K_{CT}(p_{uw} - c))$$

where $p_{uw} = (p_u, p_w)$ is the image coordinate of point p after projecting to camera at view point v and $\mathbb{P}$ is the sampled set of points belonging to the target. K_{CT} is a distance's correlation matrix.

4.4.3 Occlusions

Even though there may be no occlusion when the target is first detected, but view point may get obstructed after the movement of the human or robot. Here we mainly consider the occlusion source from the robot because 1) we assume the user is cooperative and does not want to hide the functioning object to the camera in deliberate; 2) a comparatively free space for both human and robot interaction is assumed provided. Otherwise, human and robot's movement would be difficult.

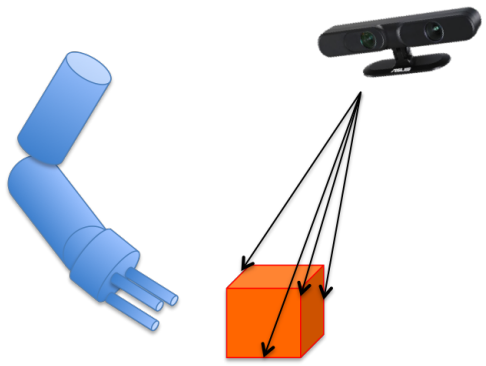


Figure 4.3: Illustration of occlusion detection. Robot parts are represented as cylinders and viewing rays from camera to the object are checked whether intersecting with robot parts.

In the occlusion detection, we represent robot parts (upper, lower arm, wrist, and fingers) as cylinders and check if any of them intersect with viewing segments starting from the camera ending at the object. The positions and orientations of the robot parts are from robot maintained joints' transform tree. A visualization of these transform links is shown in Figure 4.1(a) . We measure and store the size (height and radius) of each cylinder before.

Note that the occlusion from scene object can also be handled using similar ways if we know the scene object shape model and poses.

So the score function of occlusion for view v is defined as:

$$s_{SO}(v,x) = \frac{1}{|\mathbb{X}|}\Sigma_{x\in\mathbb{X}}(1 - \prod_{p\in\mathbb{P}}(1 - overlap(v,x,p)))$$

where $\mathbb{P}$ is the set of robot parts. Function $overlap(\cdot)$ will return 1 of the segment from viewpoint v to the sampled point x intersects with part p, which is represented as a cylinder in our case.

4.4.4 View Direction

View direction constraint is to make sure the target in a new viewpoint can be clearly seen. Thereby abstracting and analyzing the feature on that viewpoint can be more reliable.

For example, in Figure 4.4, suppose the triangle Δ in the air is the target object. Plane a and b are two view planes that are perpendicular to their viewing axis. After projection, two projected triangles Δ_a and Δ_b are of different size and

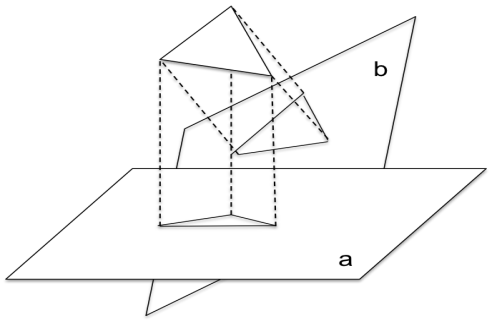


Figure 4.4: Illustration of a triangle projected to different planes.

we prefer the result on plane b since it is larger and could provide more features for the subsequent image processing.

For the convenience of implementation, we choose a triangle that lies inside the target object and projects it to each viewpoint v. The projecting process is the same as subsection 4.4.2. The view direction score is counted as 1 minus the ratio of projected triangle area and the largest possible projected triangle area.

$$s_{VD}(v, x) = 1 - \frac{area(\Delta_v)}{max\_area}$$

max_area is the area of the projected triangle whose view axis is perpendicular to the plane of 3D triangle and the distance between view point and triangle plane is the shortest (30cm in our implementation).

4.4.5 Precomputation

To reduce the online computation time for the viewpoint selection and guarantee an accurate control of the robot, we precompute a mapping that links the joint values of camera arm with effector's position and orientation.

The pipeline is shown in Figure 4.5.

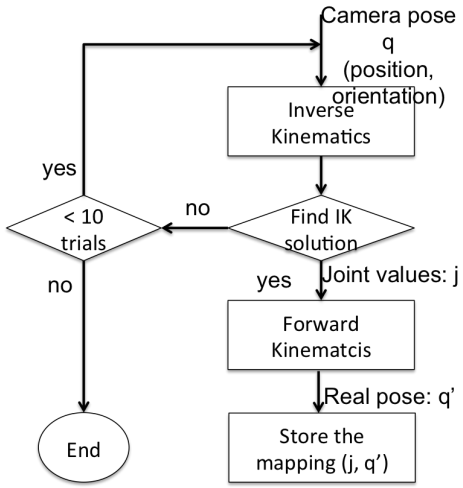


Figure 4.5: The float chart of precomputation pipeline.

For each target camera's pose q, that is composed of position and orientation, we first use inverse kinematics(IK) to get the camera arm's joint value j. Due to the randomness of inverse kinematics algorithm, a reachable end effector pose may not have a joint space solution. We try 10 times here before we drop off that camera's position.

However, inverse kinematics can only return a proximate result. Using q to analyze viewpoint property may bring incorrect result because the arm is not moving there. So we feed the joint values from IK to forward kinematics to get the camera's pose q'. And q' is the exact pose that camera will go. Finally, the mapping between q' and j is stored and used to represent a view point.

In the next sections, we are going to introduce two human-robot interaction applications that use the active vision module discussed above.

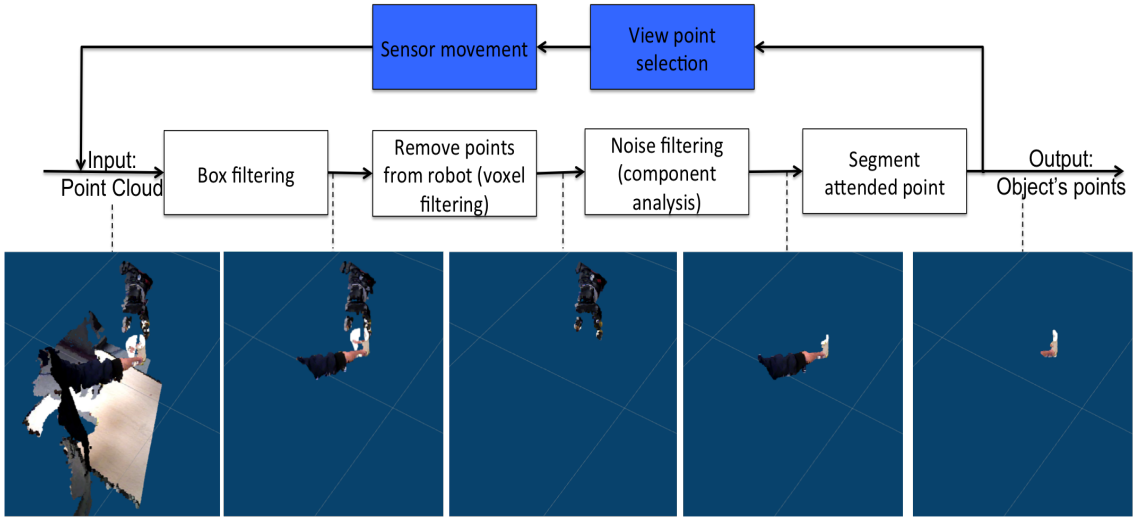


Figure 4.6: An illustration of the in-hand object detection pipeline for a user passing object to robot task. White boxes are the modules in detection algorithms. Blue boxes are the active viewpoint control modules we proposed. Point cloud processing results are exemplified at the lower row.

4.5 User Passes an Object to the Robot

The first application is for human giving an object to a robot. Because of the unpredicted position of user's hand and potential occlusion during the process of robot approaching, we need to control the viewpoints to make sure human's hand is always in view and can be detected.

Before we introduce the viewpoint control component, we first have an explanation of our visual detection system and robot grasping policy. Note that the vision algorithm will take input from a static camera (as shown in Figure 4.2(b)) before integrating our active viewpoint control component.

4.5.1 Detection of the Object in Hand

The in-hand object detection process is illustrated in Figure 4.6. We assume the user is cooperative and does not want to hide the object from the robot. In the running time, our detection algorithm is looking for the object that is above the table and closest to the robot.

Given a collected point cloud, box filtering module will first extract the points above the table because we assume the user will hold the object above the table and in the front of the robot.

Because the points belonging to both the robot and the object may appear in the table above area, we have to remove the robot owned points in the next step. The general procedure is to first obtain the pose of the selected joint frames in the right arm (as shown in Figure 4.1(a)) by looking up robot maintained transform system. Then find the points nearby these joints and extract all the points that in the same connected component as robot's points. However, due to the noise and viewing issue, there would be some points belonging to the robot but untracked. So we create a voxel grid of 8 cm^3 and filter our points lying in the same voxel with robot points.

Before locating the nearest point's connected component to the robot, we need to filter out the sensor noise, which appears randomly in the image but of small size. So we find all the connected components of points and treat the components of the size less than 500 as sensor noise and throw them.

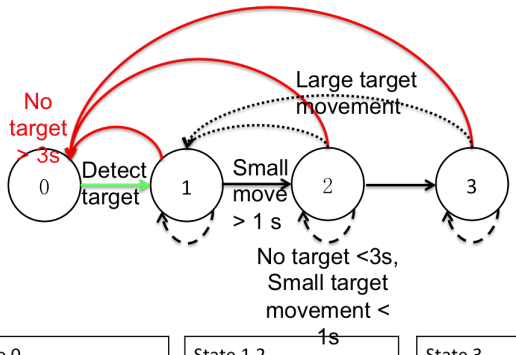
In the last step, we extract the point that is closest to the robot as our attention

point, and we believe this is a point of in-hand object or hand. Then segment a box area around the attention point and treat the largest connect component inside this box as points of the target.

4.5.2 Robot Action Policy

After getting the location of the target, the robot will try to approach and grasp the object.

However, due to the random and hesitant behavior of human being, the robot will interpret the user's intention via a state machine based on target's location history. Also depending on the human intention state, the robot will have four types of actions. Figure 4.7 shows the state machine of human purpose interpretation and corresponding robot actions.



State 0	
Cond	Action
All	reset

State 1,2	
Cond	Action
Large gap	Plan move
Small gap	IK move
Read	IK move

State 3	
Cond	Action
Large gap	Plan move
Small gap	IK move
Read	Grasp

Figure 4.7: The state machine of robot interpreting human's intention and robot's action policy under each state.

In general, human's intention state is categorized into four states, 0 denotes no human presence, 1 and 2 are the intermediate states where the user is interacting

but not determined. State 3 means hand's position now is stable and ready for grasping. Once a hand is detected, the state machine will advance to state 1. A steady holding of the object for more than 1 second will move human's state into the next one. Once there is a significant movement in the process, the state machine will fall back to state 1. An absence of human interaction for more than 3 seconds to make the machine reset to state 0. Other actions will keep the state unchanged.

Four types of reaction can be chosen on the robot's side based on human's intention state as shown in Figure 4.7.

1. Reset action means the robot retreat its grasping (right) arm and open its gripper.

2. When the target is far away from the right hand, the robot will first move to a position before the target. Because this is a movement of long distance, the trajectory will first be planned before execution.

3. When the grasping hand is not far away from the target, we will use set joints method to control the arm directly. The target joint values are computed from inverse kinematics.

4. Finally, when the robot decides to grasp., its grasping hand will go to the target location, then closes the fingers. In our implementation, we use a ReFlex hand and wrap grasp the target. There is one more active viewpoint control and vision process to check if the gripper misses the object before the robot closes its gripper. Because the process is similar to the whole passing process, we do not repeat them here.

4.5.3 View Point Control Component

Because user's hand may move out of the view point of the camera or get occluded, we apply our active viewpoint control module (blue boxes in Figure 4.6) here to actively change the view position of the camera to keep the target on track.

Figure 4.8: An illustration of the triangle used to determine the view direction of the passing object application.

The input to the view point control component is the setting of the interaction scene (x in equation 4.1) that provides necessary information for calculating view score function.

Target segmentation results shown in Figure 4.6 is given to the view point selection module for centeredness and occlusion score. Also occlusion score needs robot's joint information illustrated in Figure 4.1(b). For view direction evaluation, we select a triangle residing in the 3D bounding box of the target (see Figure 4.8) and calculate its projection w.r.t different camera views. Because joint's value have been precomputed and stored, $s_{JD}(\cdot)$ can be evaluated without any other input.

4.6 Human Sends Command via Pointing

In this section, we show another application in visual command sending, where the user stands in front of the robot and send commands to the robot by a sequence

of pointing actions. Due to the random initial positions of the hand and potential movements, the hand may go out of the view or the view direction become too narrow the detection algorithm to work.

4.6.1 Arm and Hand Detection

Arm and hand are detected via Luan's graph matching framework in [57]. The general idea is to represent the target as a template graph and, the detection procedure becomes mining a subgraph from the input scene graph that matches the template graph.

An arm and hand detection result is visualized in Figure 4.9 (a), (b).

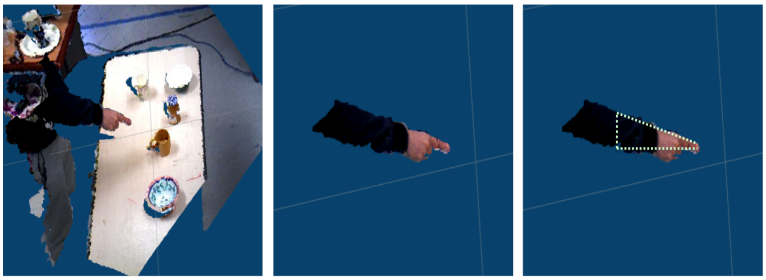


Figure 4.9: (a) (b) An illustration of an arm and pointing hand detection result. (a)Input point cloud. (b)Detected arm and hand. (c)The triangle selected to determine the view direction for the arm ad hand detection.

4.6.2 Visual Command Sending Example

A state machine is also used here to interpret the visual command sent by user's sequence of pointing actions. [57] provides an example of "heating object" command. Here we show another case of just selecting the target. This would be useful in solving the ambiguities in object references.

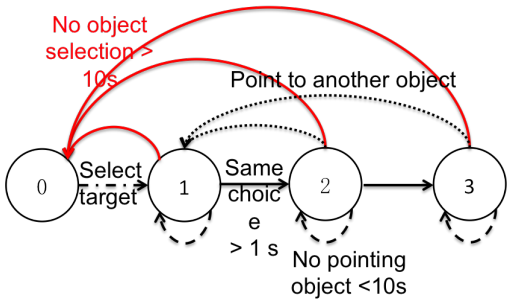


Figure 4.10: The state machine of selecting a target object.

In the beginning, the system is in state 0. Once we detect an object pointed by the user, the state goes to 1. If the user keeps pointing to the same object and holds for more than a second, the state machine advances to the next state. Once reaching state 3, the target object is confirmed chosen. If the user chooses another object while in state 1 and 2, the machine will go to state 1 and update the pointing object.If no pointing object detected for more than 10 seconds, the system falls back to the initial state 0.

4.6.3 View Point Control component

For the same reasons with passing object application, we apply active view control component here to keep user's hand in view and detection reliable.

Segmented user's hand and arm (Figure 4.10 (b)) are the input for calculating centeredness score and occlusion. The triangle for view direction checking is shown in Figure 4.10 (c), where the one point is the finger tip and the other two points are from the mid arm.

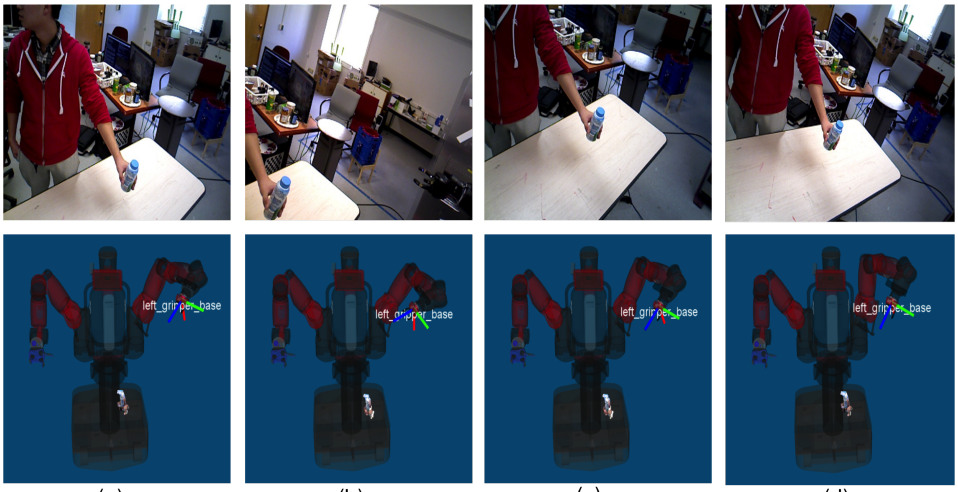


Figure 4.11: An example of human teaching viewpoint selection in object passing application. Top row: Viewing image from the camera; bottom row: corresponding robot pose. (a)The viewing image when a user hands a bottle to the robot; (b) The selected viewpoint by the initial set of score function parameters;(c) The view after human demonstrates a better view; (d) The newly selected viewpoint with learned parameters

4.7 Learning Score Function's Weights

Because the viewpoint quality is evaluated by a score function that considers viewing factors such as occlusion, view condition and movement time, the set of function coefficients can represent the viewing element preference by controlling the contribution ratio of each factor. How to find a good set of score function parameters becomes a practical problem influencing the success and the smoothness of the task execution. To manually try a set of score function weights is one way of dealing with robot task preferences. But it could require a large number of trials due to the blindness of tuning. In this section, we put forward a method that learns function weights from human expert's teaching.

The idea is to model human's teaching as a coactive learning process, where

there is a human user and a learning system both aiming to provide a good result [53]. Take our linear score function as an example, coactive learning model assumes an existence of optimal weights for the task that the active viewpoint module is serving. Here optimal weights means a set of function coefficients that best serves the robot task. Due to the difference between the unknown optimal coefficients and the real ones, the active view module may choose a viewpoint that does not fit the task. Then, a human user would provide a better viewpoint to the robot, and the viewpoint component will update its score function parameters based on human's teaching. To guarantee a better view point will be provided, here we assume the human teacher has a good knowledge of the task and can always guide the robot to a better viewpoint.

To reduce the cost of data collecting, we learn our score function weights from target's trajectory segments instead of the whole task procedure. For example, in the pointing command case, a new viewpoint needs to be determined when the user changes the mind and point to another place. We will pause the interaction process if the viewpoint from our active module is not satisfactory and demonstrate it a better viewpoint. Also for passing object application, we focus on the situation when hand moves and new viewpoint needs to be selected.

Coactive Learning needs the learning system and the user provides feedback in the same solution space. However, in our case, the human expert would teach a viewpoint in the continuous space while the viewpoint control module has discretized the viewpoints space for computation efficiency. So to make sure coactive learning framework work, we take two actions. First, we densely sample the joint space that

corresponds to the desired camera (end effector)'s pose area to reduce the effect of discretion. We will reject learning if the feedback's camera's pose is not among the designed camera's pose area to make sure there are enough viewpoint candidates around human's feedback.

Define the view score function parameter vector $\Lambda = (\lambda_{JD}, \lambda_{CT}, \lambda_{SO}, \lambda_{VD})^T$, score vectors $S(v, x) = (s_{JD}(v, x), s_{CT}(v, x), s_{SO}(v, x), s_{VD}(v, x))^T$. The final view score learning function is shown in Algorithm 5.

Algorithm 5: Learning view score function parameters

1: Initialize Λ_1,
2: **for** $i = 1$ to T **do**
3: Observe the scene setting x_t,
4: Choose view point $v_t \in \underset{v \in V}{\operatorname{argmax}}\ \Lambda_t^T S(v, x_t)$ and move to v_t,
5: Obtain expert's feedback $\hat{v}_t$
6: **if** feedback camera pose is in the valid pose area **then**
7: Update $\Lambda_{t+1} = \Lambda_t + S(v_t, x_t) - S(\hat{v}_t, x_t)$,
8: **end if**
9: **end for**

4.7.1 Learning Effect

To demonstrate learning result from human's feedback, we let user teach the active viewpoint component once and feed the user-guided camera pose to algorithm 5 9 times, which is equal with human teaches robot with the camera arm pose 9 times. Figure 4.12 illustrates the distance between the viewpoint the active viewpoint module chooses and user's feedback under different coefficients learning iterations.

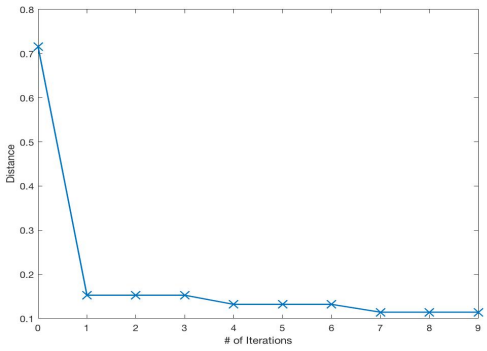


Figure 4.12: The distance between the selected viewpoints and human's demonstration under each learning iteration.

The distance between viewpoints is defined on the camera's pose:

$$dist(v_1, v_2) \triangleq ||p_{v_1} - p_{v_2}|| + (1 - \cos\theta_{v_1,v_2})$$

where the first term represents the Euclidean distance between the position of two viewpoints and second term measures the difference of two viewing angles.

As we can see from Figure 4.12, the distance to the human's feedback viewpoint is decreasing, which demonstrates the learning effect of Algorithm 5 after a repetitive teaching of the same view from a human expert.

4.7.2 An Example of View Point Learning

Here we use the task of the human passing object to a robot as an example to visually showing the effect of viewpoint preference learning from human's demonstration.

In Figure 4.11, top row is the image seen from the current viewpoint and bottom row is the corresponding robot joints visualization. In the top row, (a) is

the first camera image the robot when a user hands a bottle. Based on the initial set of score function coefficients, active view module chooses a viewpoint and moves to it, where (b) shows the image under this viewpoint and arm's position. (c) Realizing the current viewpoint control module is not giving enough preference to the target centeredness factor, a human expert moves the camera to a better viewpoint that serves the task. (d) After learning human's demonstration, the view point control module updates its weights and choose a new view point, which prefers centeredness factor more than the initial setting.

4.7.3 Results on Our Human Robot Interaction Applications

To demonstrate the result of parameter learning in the two applications we build in this work, we append a video for them.

In the passing object application, because a user sometimes changes her/his position and thereby may move out of the view of the camera. We prefer teaching the system for better keeping target in view.

For the case of sending a command via pointing, we assume the user would stand in a small area and move the arm to point to the intended object. So target object (hand in this case) is not easy to lose track and we prefer few camera movement in order to finish the command sending process quickly.

4.8 Conclusion and Future Work

In this work, we propose and implement a viewpoint control module for the human-robot interaction application. A linear time cost score function is employed to respond in time which is an essential requirement for the HRI visual modules. Additionally, we introduce coactive learning to help learn a good view selection strategy from human demonstration. It is necessary because training data is expensive when a human is in the loop, and the viewpoint's task level influence is hard to quantize while experts usually have a reasonable sense to provide a better result.

In future, we will apply this module to more human-robot interaction tasks in visual learning. Also, a motion prediction model can be introduced to handle a more dynamic human agent.

Chapter 5: Reliable Attribute-Based Object Recognition Using High Predictive Value Classifiers

5.1 Summary

We consider the problem of object recognition in 3D using an ensemble of attribute-based classifiers. We propose two new concepts to improve classification in practical situations, and show their implementation in an approach implemented for recognition from point-cloud data. First, the viewing conditions can have a strong influence on classification performance. We study the impact of the distance between the camera and the object and propose an approach to fuse multiple attribute classifiers, which incorporates distance into the decision making. Second, lack of representative training samples often makes it difficult to learn the optimal threshold value for best positive and negative detection rate. We address this issue, by setting in our attribute classifiers instead of just one threshold value, two threshold values to distinguish a positive, a negative and an uncertainty class, and we prove the theoretical correctness of this approach. Empirical studies demonstrate the effectiveness and feasibility of the proposed concepts.

5.2 Introduction

Reliable object recognition from 3D data is a fundamental task for active agents and a prerequisite for many cognitive robotic applications, such as assistive robotics or smart manufacturing. The viewing conditions, such as the distance of the sensor to the object, the illumination, and the viewing angle, have a strong influence on the accuracy of estimating simple as well as complex features, and thus on the accuracy of the classifiers. A common approach to tackle the problem of robust recognition is to employ attribute based classifiers, and combine the individual attribute estimates by fusing their information [34], [28], [58].

This work introduces two concepts to robustify the recognition by addressing common issues in the processing of 3D data, namely the problem of classifier dependence on viewing conditions, and the problem of insufficient training data.

We first study the influence of distance between the camera and the object on the performance of attribute classifiers. Unlike 2D image processing techniques, which usually scale the image to address the impact of distance, depth based object recognition procedures using input from 3D cameras tend to be affected by noise that depends on the distance, and this effect cannot easily be overcome [59].

We propose an approach that addresses effects of distance on object recognition. It considers the response of individual attribute classifiers' depending on distance, and incorporates it into the decision making. Though, the main factor studied here is distance, our mathematical approach is general, and can be applied to handle other factors affected by viewing conditions, such as lighting, viewing

angle, motion blur etc.

To implement the attribute classifiers, usually the standard threshold method is used to determine the boundary between positive and negative examples. Using this threshold the existence of binary attributes is determined, which in turn controls the overall attribute space. However, there may not be enough training samples to accurately represent the underlying distributions, which makes it more difficult to learn one good classification threshold that minimizes the number of incorrect predictions (or maximizes the number of correct predictions).

Here we present an alternative approach which applies two thresholds with one aiming for a positive predictive value (PPV), giving high precision for positive classes, and the other aiming for a negative predictive value (NPV), giving high precision for negative classes. Each classifier can then have three types of output: "positive" when above the high PPV threshold, "negative" when below the high NPV threshold and "uncertain" when falling into the interval between the two thresholds. Recognition decisions, when fusing the classifiers, are then made based on the positive and negative results. More observations thereby are needed for drawing a conclusion, but we consider this trade-off affordable, since we assume that our active agent can control the number of observations. Note that in sequential probability ratio test or similar works [60], two thresholds approach is also employed for a high confident result.

The underlying intuition here is that it should be easier to obtain the high PPV and NPV thresholds than the classical Bayes threshold (minimizing the classification error), when the number of training samples is too small to represent well

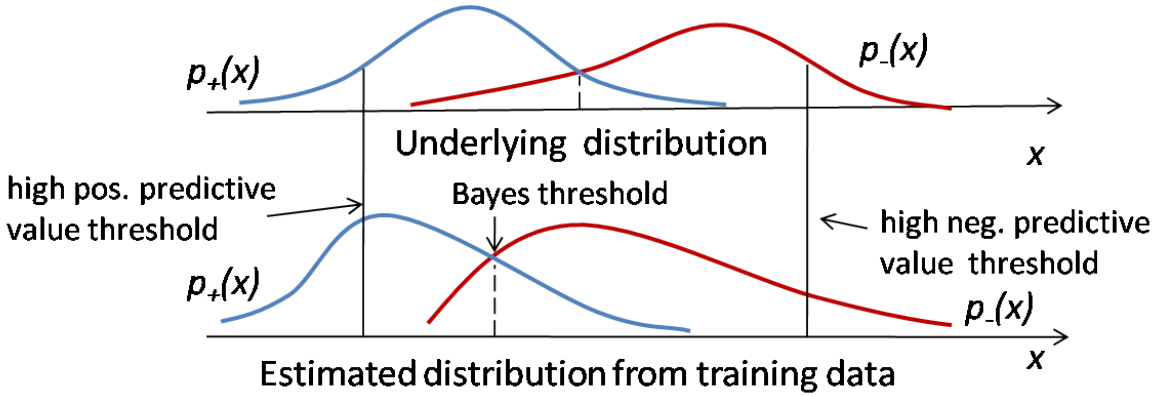


Figure 5.1: Illustration of common conditional probability density functions of the positive and negative class. Top: ground truth distribution of the two classes; bottom: a possible distribution represented by the training data. Blue line: positive class; red line: negative class. dashed line: (estimated) Bayes threshold; solid line: high PPV or NPV threshold.

the underlying distribution. Fig. 5.1 illustrates the intuition. The top figure shows the ground truth distributions (of the classification score) from the positive and negative class. The lower figure depicts the estimated distributions from training samples, which are biased due to an insufficient amount of data. Furthermore, as our experiment revealed, even the ground truth distribution could be dependent on viewing conditions, which makes it more challenging to learn a single optimal threshold. In such a case, the system may end up with an inaccurate Bayes threshold. However, it is still possible to select a high PPV (NPV) threshold by setting these thresholds (at a safe distance) away from the negative (positive) distribution. Also a certain detection rate can exist if there is enough non-overlapping area between the two distributions.

For each basic (attribute) classifier, we can also define a reliable working region indicating a fair separation of the distributions of positive and negative classes. Hence our approach can actively select "safe" samples and discard "unsafe" ones

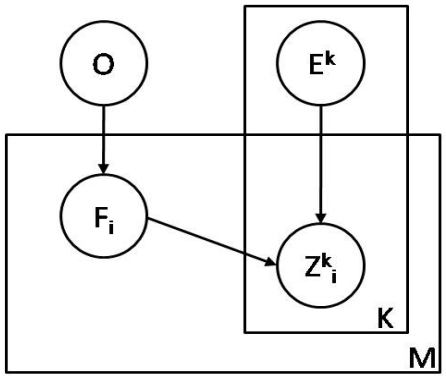


Figure 5.2: The relationship of Objects (O), attributes (F_i), environmental variables (E_k) and observations (Z_i^k) in our model.

in unreliable regions. We prove the asymptotic correctness of this approach in section 5.4.3.

Integrating both concepts, our complete approach to 3D object recognition works as follows: Offline we learn attribute classifiers, which are distance dependent. In practice, we discretize the space into n distance intervals, and for each interval we learn classifiers with two thresholds. Also, we decide for each attribute classifier a reliable range of distance intervals. During the online process our active system takes RGBD images as it moves around the space. For each input image, it first decides the distance interval in order to use the classifiers tuned to that interval. Classifier measurements from multiple images are then combined via maximum a posteriori probability (MAP) estimation.

Our work has three main contributions: 1) We put forward a practical framework for fusing component classifiers' results by taking into account the distance, to accomplish reliable object recognition. 2) We prove our fusion framework's asymptotic correctness under certain assumptions on the attribute classifier and sufficient

randomness of the input data. 3) The benefits of introducing simple attributes, which are more robust to viewing conditions, but less discriminative, are demonstrated in the experiment.

5.3 Related Work

Creating practical object recognition systems that can work reliably under different viewing conditions, including varying distance, viewing angle, illumination and occlusions, is still a challenging problem in Computer Vision. Current single source based recognition methods have robustness to some extent: features like SIFT [43] or the multifractal spectrum vector (MFS) [61] in practice are invariant to a certain degree to deformations of the scene and viewpoint changes; geometric-based matching algorithms like BOR3D [62] and LINEMOD [31] can recognize objects under large changes in illumination, where color based algorithms tend to fail. But in real complicated working environments, these systems have difficulties to achieve robust performance.

One way to deal with variations to viewing conditions is to incorporate different sources of information (or cues) into the recognition process. However, how to fuse the information from multiple sources, is still an open problem.

Early fusion methods have tried to build more descriptive features by combining features from sources like texture, color and depth before classification. For example, Asako et al. builds voxelized shape and color histogram descriptors [34] and classifies objects using SVM, while in [63] information from color, depth, SIFT

and shape distributions is described by histograms and recognized using K-Nearest Neighbors between scene and model features. Gould et al. [64] builds object-specific classifiers combining raw data, environment factor and abstracts features from different 2d and 3d sensors and classify the input using a trained logistic regression model.

Besides early fusion, late fusion also has gained much attention and achieves good results. Lutz at al. [58] proposes a probabilistic fusion approach to combine a 3D model matcher, color histograms and feature based detection algorithm MOPED [65], where a quality factor, representing each method's discriminative capability, is integrated in the final classification score. With classification score, meta information [66] can also be added to create a new feature and thereby be further classified.

Ziang et al. [28] blends classification scores from SIFT, shape, and color models with meta features providing information about each model's fitness from the input scene, which results in high precision and recall on the Challenge and Willow datasets. Considering influences due to viewing conditions, Ahmed [67] applies an AND/OR graph representation of different features and updates a Bayes conditional probability table based on measurements of the environment, such as intensity, distance and occlusions. However, these methods may suffer from inaccurate estimation of the conditional probabilities involved, because of insufficient training data.

In our work, we propose a framework for object recognition using multiple attribute classifiers, which considers both, effects due to viewing conditions and effects due to biased training data that systems face in practice. We implement

our approach for an active agent that takes advantage of multiple inputs at various distances.

5.4 Assumptions and Formulation

Before going into the details and introducing the notation, let us summarize this section. Section 5.4.1 defines the data fusion of the different classification results through MAP estimation. Section 5.4.2 proves that MAP estimation will classify correctly under certain requirements and assumptions. The requirements are restrictions on the values of the PPV and NPV. The assumptions are that our attribute classifiers perform correctly in the following sense: A ground truth positive value should be classified as positive or uncertain and a ground truth negative value should be classified as negative or uncertain. Finally section 5.4.3 proves asymptotic correctness of MAP estimation. The estimation will converge, even if the classifiers don't perform correctly, under stronger requirements on the values of the PPV and NPV.

Let the objects in the database be described by the set $\mathbb{O} = \{o_j\}$ $(j = 1, 2, ..., |\mathbb{O}|)$. Each object $o_j \in \mathbb{O}$ is represented by a attribute vector $F^j = [f_{1j}, f_{2j}, ..., f_{Mj}]^T$, where M is the number of attributes. For the i-th attribute F_i, there is a corresponding component classifier to identify it. Denote its observation as Z_i^k, where i is the index for the classifier and k is the observation number. Here we consider binary attributes $f_{ij} \in Range(F_i) = \{0, 1\}$, $\forall i \in \{1, 2, ..., M\}$, and there are three possible values for the observation : $Z_i^k = \{0, 1, u\}$ $k \in 1, 2, , , , K$, where u represents uncer-

tainty for the case that the classification score falls in the interval between the high PPV and NPV threshold.

The model also encodes effects due to viewing conditions (or environmental factors). In this work, we study the effect of distance, which has a significant impact on many depth-based recognition algorithms. Thus, E is the distance between the object and the camera. However, in future work, other environmental factors can also be encoded as additional components of the environment variable to make the framework more general. Fig. 5.2 illustrates the relationship between objects, attributes, environmental factors and observations in a graphical model.

In our notation $\mathbb{E}^K = \{E^1, E^2, ..., E^K\}$ represents the environmental variable at each observation, and $\mathbb{Z}_i^K = \{Z_i^1, Z_i^2, ..., Z_i^K\}$ is the set of observation results from i-th classifier. Here we assume that an observation of an attribute Z_i^k only depends on the ground truth attribute variable F_i and the environmental variable E^k. Because we assume that each object o_j can be represented by an M-dimension attribute vector F^j, we have

$$P(F|O = o_j) = \begin{cases} 1 & \text{if } F = F^j, \\ 0 & \text{o.w.} \end{cases} \tag{5.1}$$

5.4.1 Inference

With K observation results $\mathbb{Z}^K = \{\mathbb{Z}_1^K, ..., \mathbb{Z}_M^K\}$ and corresponding environmental conditions $\mathbb{E}^K$, we want to obtain the posterior probability of the target object being object $o_j \in \mathbb{O}$. i.e. $P(O = o_j|\mathbb{Z}^K, \mathbb{E}^K)$. Based on our graphical model

we have:

$$
\begin{aligned}
P(O = o_j|\mathbb{Z}^K, \mathbb{E}^K) &= \frac{P(O = o_j, \mathbb{Z}^K, \mathbb{E}^K)}{P(\mathbb{Z}^K, \mathbb{E}^K)} \\
&= \frac{P(O = o_j)P(\mathbb{Z}^K|F = F^j, \mathbb{E}^K)P(\mathbb{E}^K)}{P(\mathbb{Z}^K, \mathbb{E}^K)} \\
&= \frac{P(\mathbb{E}^K)P(O = o_j)}{P(\mathbb{Z}^K, \mathbb{E}^K)} \prod_{k=1}^{K}\prod_{i=1}^{M} P(Z_i^k|F_i = f_{ij}, E^k) \\
&= \lambda P(O = o_j) \prod_{k=1}^{K}\prod_{i=1}^{M} \frac{P(F_i = f_{ij}|Z_i^k, E^k)}{P(F_i = f_{ij})}
\end{aligned}
\tag{5.2}
$$

where $\lambda \triangleq \frac{P(\mathbb{E}^K)\prod_{k=1}^{K}\prod_{i=1}^{M} P(Z_i^k, E^k)}{P(\mathbb{Z}^K, \mathbb{E}^K)\prod_{k=1}^{K}\prod_{i=1}^{M} P(E^k)}$. Because

$$
\begin{aligned}
P(F_i = f_{ij}) &= \sum_{O} P(F_i = f_{ij}, O) = \sum_{t} P(O = o_t)P(F_i = f_{ij}|O = o_t) \\
&= \sum_{\{t|f_{it}=f_{ij}\}} P(O = o_t) \qquad \text{(From (5.1))}
\end{aligned}
\tag{5.3}
$$

Finally, we have

$$
P(O = o_j|\mathbb{Z}^K, \mathbb{E}^K) = \lambda P(O = o_j) \prod_{k=1}^{K}\prod_{i=1}^{M} \frac{P(F_i = f_{ij}|Z_i^k, E^k)}{\sum_{\{t|f_{it}=f_{ij}\}} P(O = o_t)} \tag{5.4}
$$

The recognition $\mathbb{A}$ then is derived using MAP estimation as:

$$
\mathbb{A} \triangleq \underset{o_j}{\text{argmax}}\ P(O = o_j|\mathbb{Z}^K, \mathbb{E}^K) \tag{5.5}
$$

In our framework, we use the high positive and negative predictive value observations ($Z = 0, 1$) to determine the posterior probability.

We also take into account the influence of environmental factors. That is,

only observations from a reliable working region are adopted in the probability calculation. When the environmental factor is distance, the reliable working region is defined as a range of depth values where our attribute classifier works reasonably well. We treat a range of distance values as a reliable working region for a classifier, if the detection rate in it is larger than a certain threshold, and the PPV meets the system requirement.

This requirement for the component classifiers is achievable if the positive conditional probability density function of the classification score has a non-overlapping area with the negative one. Then we can tune the classifier's PPV threshold towards the positive direction (towards left in Fig. 5.1) to achieve a high precision with a guarantee of minimum detection rate.

We will prove in the next section that our framework can yield a correct result asymptotically if the precisions are high enough in the recall lower bound existence working environment and input are sampled randomly.

Formally speaking, our $P(F_i = f_{ij}|Z_i^k, E^k)$ is defined as:

$$P(F_i = 1|Z_i^k, E^k) = \begin{cases} p_i^+ & \text{if } e_k \in \mathbb{R}_i \ \& \ z_i^k = 1, \\ 1 - p_i^- & \text{if } e_k \in \mathbb{R}_i \ \& \ z_i^k = 0, \\ \sum_{t|f_{it}=f_{ij}} P(O = o_t) & \text{o.w.} \end{cases} \tag{5.6}$$

where $\mathbb{R}_i$ is the set of environmental values for which the i-th classifier can achieve a PPV p_i^+ with a detection rate lower bound. As before, k denotes the k-th observation. If the above condition is not met, either the recognition is done in an

unreliable region or the answer is uncertain. Now equation (5.4) can be rewritten as:

$$P(O = o_j|\mathbb{Z}^K, \mathbb{E}^K) = \lambda P(O = o_j) \prod_{k=1}^{K} \prod_{i \in \mathbb{I}^k} \frac{P(F_i = f_{ij}|Z_i^k, E^k)}{\sum_{\{t|f_{it}=f_{ij}\}} P(O = o_t)} \tag{5.7}$$

where $\mathbb{I}^k = \mathbb{I}^{k+} \cup \mathbb{I}^{k-}$ is the index set of recognized attributes at the k-th observation with $\mathbb{I}^{k+} = \{i|e^k \in \mathbb{R}_i \ \& \ z_i^k = 1\}$ and $\mathbb{I}^{k-} = \{i|e^k \in \mathbb{R}_i \ \& \ z_i^k = 0\}$.

Intuitively, it means that we only use a component classifier's recognition result when 1) it works in its reliable range; 2) the result satisfies high PPV or NPV thresholds. In Section 5.4.2, we will introduce the predictive value requirements for the component classifiers.

5.4.2 System Requirement for the Predictive Value

Here we put forward a predictive value requirement for each component classifier to have correct MAP estimations assuming there do not exist false positive or false negative from observations.

To simplify our notations, we define the prior probability of object $\pi_j \triangleq P(O = o_j), j = (1, 2, ..., N_o)$ and the prior probability of attribute F_i being positive as $w_i \triangleq \sum_{\{t|f_{it}=1\}} \pi_t, (i = 1, 2, ..., M)$. For each attribute, the following ratios are calculated: $r_i^+ \triangleq \max(1, \frac{\max_{\{t|f_{it}=0\}} \pi_t}{\min_{\{t|f_{it}=1\}} \pi_t})$, $r_i^- \triangleq \max(1, \frac{\max_{\{t|f_{it}=1\}} \pi_t}{\min_{\{t|f_{it}=0\}} \pi_t})$. $\mathbb{I}_{F_j}^+$ and $\mathbb{I}_{F_j}^-$ are the index sets of positive and negative attributes in F^j, and the reliably recognized attributes' indexes at the k-th observation are denoted as $\mathbb{I} = \{\mathbb{I}^1, \mathbb{I}^2, ..., \mathbb{I}^K\}$ ($\mathbb{I}^k$ as defined in section 5.4.1). We next state the conditions for correct MAP estimation.

Theorem 5.4.1. *If the currently recognized attributes* $\bigcup_k \mathbb{I}^k$ *can uniquely identify object* o_j*, i.e.* $\bigcup_k \mathbb{I}^{k+} \subseteq \mathbb{I}_{F_j^+}$, $\bigcup_k \mathbb{I}^{k-} \subseteq \mathbb{I}_{F_j^-}$, $\forall t \neq j, \bigcup_k \mathbb{I}^{k+} \nsubseteq \mathbb{I}_{F_t^+}$ *or* $\bigcup_k \mathbb{I}^{k-} \nsubseteq \mathbb{I}_{F_t^-}$*, and if* $\forall i \in \{1, 2, ..., M\}$ *the classifiers' predictive values satisfy* $p_i^+ \geq \frac{r_i^+ w_i}{1+(r_i^+-1)w_i}$ *and* $p_i^- \geq \frac{r_i^-(1-w_i)}{w_i+r_i^-(1-w_i)}$*, then the MAP estimation result* $\mathbb{A} = \{o_j\}$.

This requirement means that if 1) the attributes can differentiate an object from others, and 2) the component classifiers' predictive values satisfy the requirement, then for the correct observation input, the system is guaranteed to have a correct recognition result.

Proof. Based on (5.7) and the definition above, the posterior probability of o_j is,

$$P(O = o_j|\mathbb{Z}^K, \mathbb{E}^K) = \lambda \pi_j \prod_{k=1}^{K} \left(\prod_{i \in \mathbb{I}^{k+}} \frac{p_i^+}{w_i} \prod_{i \in \mathbb{I}^{k-}} \frac{p_i^-}{1-w_i} \right) \tag{5.8}$$

Because the current observed attributes $\bigcup_k \mathbb{I}^k$ can uniquely identify o_j, we will have $\forall o_g \in \mathbb{O}/\{o_j\}$, $\exists \mathbb{I}_g \subseteq \bigcup_k \mathbb{I}^k$ and $\mathbb{I}_g \neq \emptyset$, s.t. $\forall i \in \mathbb{I}_g, f_{gi} = 0$ if $i \in \mathbb{I}^{k+}$ or $f_{gi} = 1$ if $i \in \mathbb{I}^{k-}$. Thus, $\forall o_g \in \mathbb{O}/\{o_j\}$,

$$\begin{aligned} P(O = o_g|\mathbb{Z}^K, \mathbb{E}^K) = \lambda \pi_g \prod_{k=1}^{K} \Bigg(& \prod_{i \in \mathbb{I}^{k+}/\mathbb{I}_g} \frac{p_i^+}{w_i} \prod_{i \in \mathbb{I}^{k+} \bigcap \mathbb{I}_g} \frac{1-p_i^+}{1-w_i} \\ & \prod_{i \in \mathbb{I}^{k-}/\mathbb{I}_g} \frac{p_i^-}{1-w_i} \prod_{i \in \mathbb{I}^{k-} \bigcap \mathbb{I}_g} \frac{1-p_i^-}{w_i} \Bigg) \end{aligned} \tag{5.9}$$

Since for each classifier, $p_i^+ \geq \frac{r_i^+ w_i}{1+(r_i^+-1)w_i}$ and $r_i^+ = \max(1, \frac{\max_{\{t|f_{it}=0\}} \pi_t}{\min_{\{t|f_{it}=1\}} \pi_t})$, we have $\pi_j \frac{p_i^+}{w_i} \geq \pi_g \frac{1-p_i^+}{1-w_j}$ and $\frac{p_i^+}{w_i} \geq 1 \geq \frac{1-p_i^+}{1-w_i}$. For similar reasons, we have $\pi_j \frac{p_i^-}{1-w_i} \geq \pi_g \frac{1-p_i^-}{w_j}$ and $\frac{p_i^-}{1-w_i} \geq 1 \geq \frac{1-p_i^-}{w_i}$. Also since $\mathbb{I}_g \neq \emptyset$, we can have (5.8) > (5.9), an thus

the conclusion is reached. □

From the proof, we can extend the result to a more general case: if the currently recognized attributes cannot uniquely determine an object, i.e. there exists a non-empty set $\mathbb{O}' = \{o_j | o_j \in \mathbb{O}, \mathbb{I}_{F_j^+} \supseteq \bigcup_k \mathbb{I}^{k+} \ \& \ \mathbb{I}_{F_j^-} \supseteq \bigcup_k \mathbb{I}^{k-}\}$, the final recognition result $\mathbb{A} = \underset{o_j \in \mathbb{O}'}{\operatorname{argmax}} \pi_j$. Furthermore, if an equal prior probability is assumed, then $\mathbb{A} = \mathbb{O}'$.

Theorem 5.4.1 proves the system's correctness under correct observations. For the general case, section 5.4.3 is going to prove that MAP estimation asymptotically converges to the actual result under certain assumptions.

5.4.3 Asymptotic Correctness of the MAP Estimation

Now we are going to prove that the MAP estimation will converge to the correct result when 1) the attribute classifiers' PPV and NPV are high enough in their reliable working region, where a lower bound of detection rate exists, and 2) the inputs are sampled randomly.

Denote d_i as the detection rate and q_i as the false-positive rate of i-th attribute classifier when applying the high PPV threshold in its reliable working region. Similarly, for the high NPV threshold, s_i denotes the true negative rate and v_i denotes the false negative rate.

Theorem 5.4.2. *We assume that the inputs are sampled sufficient randomly such that each attribute classifier gets the same chance to work in its reliable region where a lower bound exists for its detection rate,* $0 < A < d_i \leq 1$ *and all the objects have*

different positive attributes, i.e. $\forall i, j,\ i \neq j$ *s.t.* $\mathbb{I}_{F_i^+} \nsubseteq \mathbb{I}_{F_j^+}$. *If the component classifiers' predictive values* p_i^+ *and* p_i^- *are high enough, MAP estimation will converge to the correct result asymptotically with an increasing number of observations.*

Proof. Consider the worst case, where only two candidates $\mathbb{O} = \{o_1, o_2\}$ exist. Without loss of generality, assume o_1 has positive attributes $\mathbb{I}_{F_1^+} = \{1, 2, ..., M_1\}$ and o_2 has all the remaining positive attribute $\mathbb{I}_{F_2^+} = \{M_{1+1}, M_{1+2}, ..., M\}$, where $M_1 \geq 1$. Also assume o_1 is the ground truth object. In this case all the false-positive and false-negative recognition of attributes will drive the estimation result toward o_2.

Based on (5.7), the posterior probability distributions of o_1 and o_2 can be written as:

$$P(O = o_1|\mathbb{Z}^K, \mathbb{E}^K) = \lambda\pi_1 \prod_{i=1}^{M_1} (\frac{p_i^+}{w_i})^{n_i^+} (\frac{1-p_i^-}{w_i})^{n_i^-} \prod_{i=M_1+1}^{M} (\frac{1-p_i^+}{1-w_i})^{n_i^+} (\frac{p_i^-}{1-w_i})^{n_i^-} \tag{5.10}$$

$$P(O = o_2|\mathbb{Z}^K, \mathbb{E}^K) = \lambda\pi_2 \prod_{i=1}^{M_1} (\frac{1-p_i^+}{1-w_i})^{n_i^+} (\frac{p_i^-}{1-w_i})^{n_i^-} \prod_{i=M_1+1}^{M} (\frac{p_i^+}{w_i})^{n_i^+} (\frac{1-p_i^-}{w_i})^{n_i^-} \tag{5.11}$$

where n_i^+ and n_i^- are the number of positive and negative recognition results of the i-th attribute. Denote n as the number of times the i-th classifier works in its reliable region $\mathbb{E}^i$. Based on the centrum limit theorem, we have $P(n_i^+ > n\frac{d_i}{\alpha}) = 1$ and $P(n_i^- < n\alpha v_i) = 1$ for $i = 1, 2, ..., M_1$ when n goes to infinity and α can be any positive constant larger than 1.

For the same reason, we have $P(n_i^+ < n\alpha q_i) = 1$ for $i = M_1 + 1, ..., M$ when n goes to infinity. We use the same n here because of the assumption of the same likelihood of reliable working regions for each classifier. Actually it does not matter

if there is a constant positive factor on n, which means each classifier's reliably working chance is proportional.

Dividing (5.10) by (5.11), we obtain:

$$\begin{aligned}
\frac{P(O=o_1|\mathbb{Z}^K,\mathbb{E}^K)}{P(O=o_2|\mathbb{Z}^K,\mathbb{E}^K)} &= \frac{\pi_1}{\pi_2}\frac{\prod_{i=1}^{M_1}(\frac{p_i^+/w_i}{(1-p_i^+)/(1-w_i)})^{n_i^+}(\frac{(1-p_i^-)/(w_i)}{p_i^-/(1-w_i)})^{n_i^-}}{\prod_{i=M_1+1}^{M}(\frac{p_i^+/w_i}{(1-p^+i)/(1-w_i)})^{n_i^+}(\frac{(1-p_i^-)/w_i}{p_i^-/(1-w_i)})^{n_i^-}} \\
&\geq \frac{\pi_1}{\pi_2}\frac{\prod_{i=1}^{M_1}(\frac{p_i^+/w_i}{(1-p_i^+)/(1-w_i)})^{n\frac{d_i}{\alpha}}(\frac{(1-p_i^-)/(w_i)}{p_i^-/(1-w_i)})^{n\alpha v_i}}{\prod_{i=M_1+1}^{M}(\frac{p_i^+/w_i}{(1-p^+i)/(1-w_i)})^{n\alpha q_i}} \\
&\quad (p_i^+,\ p_i^-\text{ larger than the threshold in theorem 5.4.1}) \\
&= c_1\left(c_2\frac{\prod_{i=1}^{M_1}(\frac{p_i^+}{1-p_i^+})^{\frac{d_1}{\alpha}}(\frac{1-p_i^-}{p_i^-})^{\alpha v_i}}{\prod_{i=M_1+1}^{M}(\frac{p_i^+}{1-p_i^+})^{\alpha q_i}}\right)^n \geq c_1\left(c_2\frac{\prod_{i=1}^{M_1}(\frac{p_i^+}{1-p_i^+})^{\frac{A}{\alpha}}(\frac{1-p_i^-}{p_i^-})^{\alpha\frac{1-p_i^-}{w_i}}}{\prod_{i=M_1+1}^{M}(\frac{p_i^+}{1-p_i^+})^{\alpha\frac{(1-p_i^+)}{1-w_i}}}\right)^n
\end{aligned} \tag{5.12}$$

(for the upper bound of q_i and v_i see (5.13) (5.14))

Because $\lim_{p\to 1}\frac{p}{1-p}=\infty$ and $\lim_{p\to 1}(\frac{p}{1-p})^{1-p}=1$, the division will be larger than 1 when the predictive value of each classifier is high enough, which means the MAP will yield o_1 asymptotically.

The proof of upper bound of q_i and v_i:

$$q_i = P(Z_i=1|F_i=0) = \frac{P(Z_i=1)(1-p_i^+)}{1-w_i} \leq \frac{1-p_i^+}{1-w_i} \tag{5.13}$$

$$v_i = P(Z_i=0|F_i=1) = \frac{P(Z_i=0)(1-p_i^-)}{w_i} \leq \frac{1-p_i^-}{w_i} \tag{5.14}$$

□

Beyond providing theoretical background, in the next section we perform ex-

periments on a real object recognition task to first demonstrate the influence of the environment, and then to validate our framework's performance.

5.5 Experiments

In this section, we demonstrate our framework on the task of recognizing objects on a table top. We build a pipeline to collect our own data. The reason for collecting our own data is that currently available RGBD datasets [68], [69] focus on other aspect, usually pose or multiview recognition, and they do not provide a sufficient amount of samples under varying observation distance.

Three experiments are conducted to show 1) the necessity of incorporating environmental factors (the recognition distance in our case) for object recognition; 2) the performance of the high predictive value threshold classifier in comparison to the single threshold one; and 3) the benefits of incorporating less discriminative attributes for extending the working range of classifiers.

5.5.1 Experimental Settings

The preprocessing pipeline is illustrated in Fig. 5.3. After a point cloud is grabbed from a 3D camera such as Kinect or Xtion PRO LIVE, we first apply a passthrough filter to remove points that are too close or too far away from the camera. Then the table surface is located by matching the point could to a 3D plane model using random sample consensus (RANSAC), and only points above the table are kept. Finally, on the remaining points, Euclidean clustering is employed

Figure 5.3: Illustration of preprocessing pipeline. Left: input; Middle: point cloud after passthrough filter; Right: segmented candidate and removed table surface.

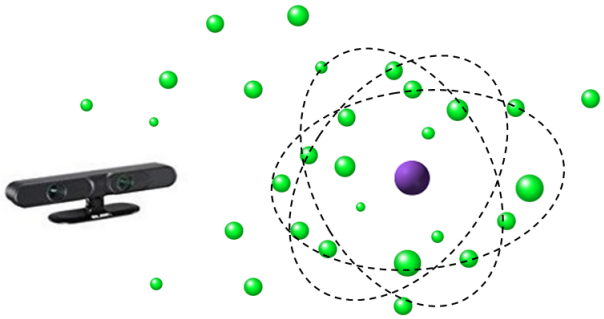


Figure 5.4: Illustration of our fine shape matching. Model point clouds (green balls) captured nearby input point cloud (purple) are retrieved first. Then we find the minimum matching distance of features between the scene and model.

to generate object candidates, among which segmented point clouds with smaller than 600 points are discarded.

For the segmented point clouds, three categories of classifiers are applied, which are tuned to attributes of fine shape, coarse shape and color.

Fine shape is recognized by the Viewpoint Feature Histogram (VFH) descriptor, which encodes a point cloud into a 308 dimensional vector. Radu [33] provides a pipeline of computing VFH features and retrieving the minimum feature distance matching between the scene object and the database model objects by fast approximate K-Nearest Neighbors, implemented in the Fast Library for Approximate Nearest Neighbors (FLANN) [70]. However, this approach tends to generate false

positives when matching different point clouds with very different distances to the camera. Considering this phenomenon, we adapt the original recognition pipeline to a two step matching. We first pick up model point clouds in our database which have similar distance to the given input point cloud to be compared. Among the nearby template point clouds, we use the minimum VFH feature matching distance as the classification score. Both steps use FLANN to accelerate neighbor retrieval while the former step employs the Euclidean distance and the latter one uses the Chi-Square distance.

We also use coarse shape as another type of attribute, which is less selective than the fine shape attribute. Our experiments later on demonstrate its advantage of having a larger working region, thence it can help to increase the system's recognition accuracy over a broader range of distance. Two coarse shapes, cylinders and planar surfaces, are recognized by fitting a cylindrical and a plane model, whose coefficients are estimated by RANSAC. The percentage of outlying points is counted as the classification score for the shape. Thus, a lower score indicates better coarse attribute fitting in our experiment.

The last type of attribute we implement in our system is color in order to augment the system's recognition capability. To control the influence of illumination, all samples are collected under one stable lighting condition. The color histogram is calculated on point clouds after Euclidean clustering, where few background or irrelevant pixels are involved. Hue and saturation channel of color are discretized into 30 bins (5×6), which works well for differentiating the major colors.

As shown in Fig. 5.5, there are 9 candidate objects in our dataset. To recognize

Figure 5.5: The objects we use in the task and their IDs

Object ID	plane surface	cylin-der	gable top carton shape	box shape	wide mouth bottle shape	cup shape	bottle shape	red color	blue color	yellow color
1	✓	-	✓	-	-	-	-	-	✓	-
2	✓	-	✓	-	-	-	-	✓	-	-
3	✓	-	✓	-	-	-	-	-	-	✓
4	✓	-	-	✓	-	-	-	✓	-	-
5	-	✓	-	-	✓	-	-	-	-	-
6	-	✓	-	-	-	✓	-	-	✓	-
7	-	✓	-	-	-	-	✓	-	-	✓
8	-	✓	-	-	-	-	✓	✓	-	-
9	-	✓	-	-	-	-	✓	-	✓	-

Table 5.1: Object IDs and their list of attributes

them, we use 5 fine shape attributes: shape of cup, bottle, gable top carton, wide mouse bottle and box; 2 coarse shape attributes: cylinder and plane surface; 3 major colors: red, blue and yellow. Each object's attribute is listed in Table 5.1. We set the recognition distance as the only changing factor in the following experiments and also fix the object's pose.

5.5.2 Experimental Results

EXPERIMENT ONE: The first experiment is designed to validate our claim that the classifiers' response score distribution are indeed distance variant. Therefore, it is necessary to integrate distance in a robust recognition system.

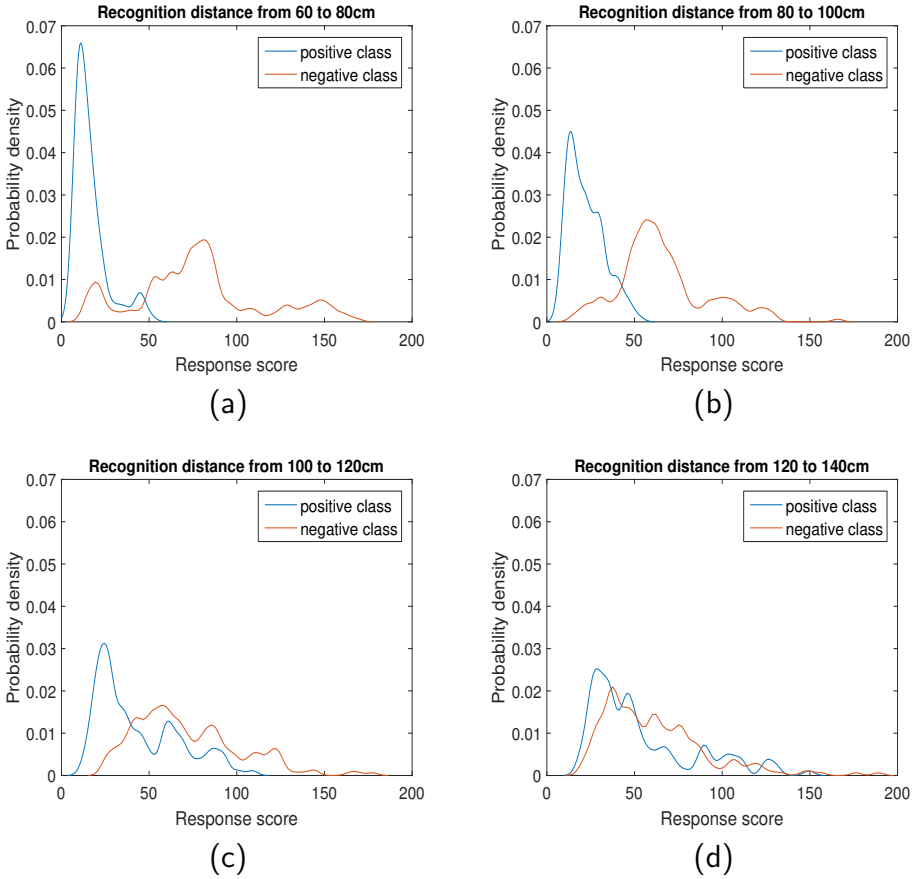


Figure 5.6: Estimated distribution of bottle shape classifier's response score under 4 recognition distance intervals.

Taking the fine shape classifier recognizing bottle shape as an example, we divide the recognition distance from 60 cm to 140 cm into 4 equally separated intervals and collect positive samples (object id $7, 8, 9$) and negative samples from the rest of the 9 objects in each distance interval. The number of positive samples in each interval is 120 with 40 objects from each positive instance while the number of negative samples is 210 with 35 from each instance. The distribution of the bottle classifier's response score is approximated by Gaussian kernel density estimation with a standard deviation of 3, and plotted in Fig. 5.6.

We observe that the output score distribution depends on the recognition

distance interval. Therefore, relying on one single classification threshold across all the distance intervals would introduce additional error. More importantly, we observe that with a larger distance, the overlapping area between the positive and negative distribution becomes wider, which makes classification more difficult.

EXPERIMENT TWO: Experiment one demonstrated the difficulty of learning a distance-variant ground truth distribution and corresponding classification thresholds. Therefore, we propose to use two high predicative value thresholds when multiple inputs are available. The second experiment is designed to validate this idea by comparing the classification accuracy of an estimator that 1) uses two high predicative value thresholds, to an estimator that uses 2) one optimal Bayes threshold, which minimizes the classification error on the training data.

To have a fair comparison, we set our task as recognizing 5 objects (id $1, 4, 5, 6, 9$) with 5 fine shape attributes such that each object contains one positive attribute that uniquely identifies it. Both training and testing point clouds are collected at a distance of 100 cm to 120 cm from the camera. To learn the classification threshold, we sample 26 point clouds for each object and uniformly select 20 from them for the training. The testing data for each object consists of 22 point clouds that we can randomly choose from to simulate the scenario of an active observer moving around to gather multiple perception inputs. Here we want to mention a special case. When our framework is uncertain based on the current input, it randomly select one of possible objects with equal probability. The classification accuracy between using a single threshold and using two high predicative value thresholds are shown in Fig. 5.7 respectively.

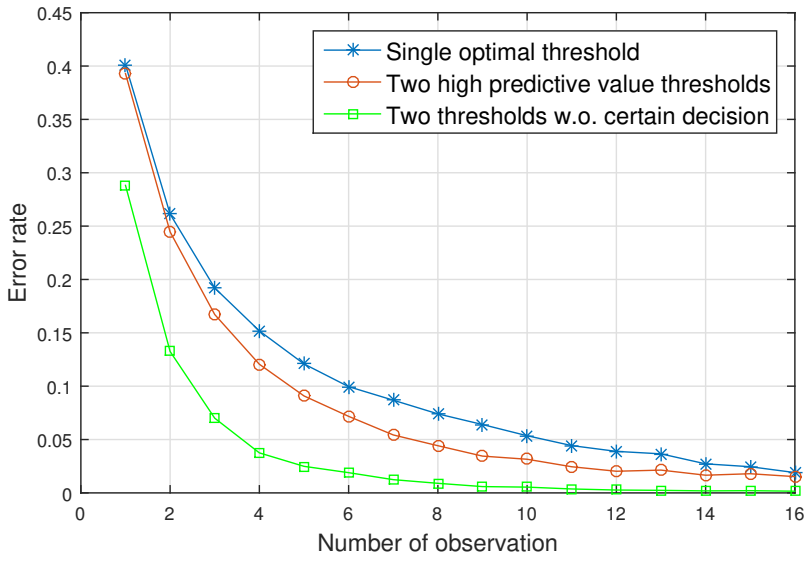


Figure 5.7: Error rate using single threshold (blue) and two high predicative value thresholds (red) classification. The green line depicts the error introduced when the two thresholds method has to randomly select for cases when more than one object is estimated as possible candidate.

We can see that both methods' error rates decrease when the number of observations increases. The approach using two thresholds has lower error rate than the one using a single threshold. The green line shows the error introduced by random selection, when our framework cannot make a sole decision. This error makes up the major part of the total framework error and it approaches zeros with the number of observations increasing. It is worth mentioning that under theoretical conditions, the classical Bayes single threshold should still be the best in minimizing the classification error. Our method provides an alternative for cases when the training data does not represent very well the underlying distribution in real world scenarios.

EXPERIMENT THREE: The third experiment demonstrates the benefits of using less discriminative attributes for extending the system's working range. To recognize the 9 objects

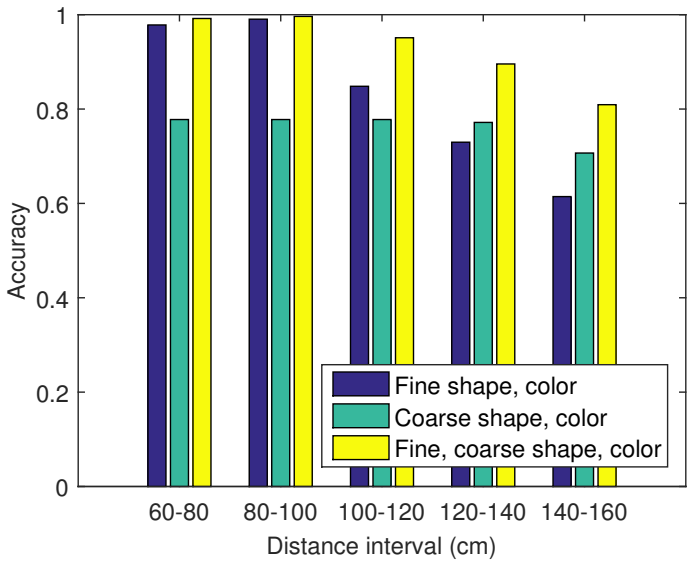


Figure 5.8: Three systems' recognition accuracy in different working distance interval.

in Fig. 5.5, we build three recognition systems utilizing attributes of fine shape with color, coarse shape with color, and all of the three attributes, respectively. Considering the influence of the recognition distance on the response score distribution, the complete distance from 60 cm to 160 cm is split into 5 equal intervals. We then learn the classification thresholds and predictive values accordingly. Both, the training and the testing data, consist of around 100 samples from each object across recognition distances from 60 cm to 160 cm. We learn the PPV and NPV by directly counting the training data w.r.t. the thresholds and select thresholds satisfying a predictive value larger than 0.96. The minimum detection rate for the reliable working distance interval is 0.3. This means if 1) an attribute classifier cannot find a threshold with PPV larger than 0.96, and 2) detection rate larger than 0.3 in a certain distance interval, the output of this attribute classifier in this interval will not be adopted for decision making. In the testing phase, we constrain the multiple

input point clouds collected from the same distance interval for a fair comparison of the system's performance in each working region. Around 120 point clouds for each object are collected to sample from. Similar as in the second experiment, random selection is applied when multiple objects are found as possible candidates.

Fig. 5.8 displays the systems' recognition accuracy after observing three times in each distance interval. As expected, the classification performance starts to decrease when working at a larger distance between the camera and the objects. In the distance region from 120 cm to 160 cm, the system using a fine shape attribute (blue) even performs worse than the system using the less selective coarse attributes (green), which validates that the coarse shape based classifier has a larger working region, though its simple working mechanism restricts its capability in differentiating compared to fine grain attribute based classifier. Finally, due to the complementary properties, the system accuracy (yellow) using all attributes achieves the best performance at each working region.

5.6 Summary

In this work we put forward a practical multiple attributes based object recognition framework incorporating recognition distance into the decision making. Considering the difficulties of finding a single best classification threshold and the availability of multiple inputs in testing time, we propose to learn a high PPV and a high NPV threshold and discard the uncertainties during decision making. The framework's correctness was proven and a fundamental experiment was conducted

to demonstrate our approach's feasibility and benefits. Additionally, the advantage of less selective attributes compared to the sophisticated ones are shown since their relatively simple mechanism could lead to high reliability when the system is working at larger distances.

For future work, we plan to experiment on a variety of environmental factors such as lighting condition, blur and occlusions. Additionally more attribute classifiers could also be incorporated to beef up the system's overall recognition capacity.

Chapter 6: Conclusions

In this dissertation, we propose methods on various levels of object detection pipeline to achieve efficient target detection and reliable object recognition for robot tasks.

For single input image and when only a binary target classifier is available, we propose a method to active sample to detect the targets efficiently. The method exploits the classifier's response score pattern to avoid an expensive, exhaustive searching for targets. A decay function is used to model the pattern of classifier response score in the positive classification region. We estimate the probability of an unobserved window containing targets by comparing the predicted response score and the observed one. Posterior sampling is applied to decide the next window to observe. Experimental results on human detection show that our approach can achieve higher detection rate than the MS-PW and sliding window method using the same total windows budget.

If an offline learning and interaction of target object are allowed before the detection, we describe the target with a set of template graphs over segmented surfaces and present the concept of early recognition, which combines the candidate proposal and classification process to achieve fast and reliable detection performance.

A greedy policy is also put forward to generate a sub-optimal target detection constraints checking order. We prove it has bounded time cost compared to the optimal checking sequence. Experiments on one rigid object and one non-rigid body part detection validate our pipeline. To show that our framework's application, we further present a human-robot interaction system based on our non-rigid body part detection.

When robot's camera arm can be controlled, we propose and implement a viewpoint control module for target detection in the human-robot interaction application. A linear time complexity score function is employed to respond in time which is an essential requirement for the human-robot interaction vision modules. Additionally, we introduce coactive learning to help learn a good view selection strategy from human demonstration. It is very useful because training data is expensive when a human is in the loop, and the viewpoint's task level influence is hard to quantize while experts usually have a reasonable sense to provide a better result.

Finally, we put forward a framework for attribute-based object recognition. It incorporates environment factor into the decision making. Due to the difficulties of finding a single best classification threshold and the availability of multiple inputs at the testing time, we propose to learn two thresholds for the two classes and discard the uncertain observations during decision making. The framework's correctness was proven, and an experiment was conducted to demonstrate our approach's feasibility and benefits. Moreover, we demonstrated the benefits of less selective attributes (compared to the sophisticated ones) because their simple mechanism can lead to high reliability when the system is working in different regions.

Bibliography

[1] Bo Peng, Lei Zhang, and David Zhang. A survey of graph theoretical approaches to image segmentation, 2012.

[2] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[3] G. Gualdi, A. Prati, and R. Cucchiara. Multistage particle windows for fast and accurate object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(8):1589–1604, Aug 2012.

[4] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

[5] Pablo Negri, Xavier Clady, Shehzad Muhammad Hanif, and Lionel Prevost. A cascade of boosted generative and discriminative classifiers for vehicle detection. *EURASIP J. Adv. Sig. Proc.*, 2008.

[6] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, Sept 2010.

[7] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[8] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, September 2004.

[9] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition*, 2014.

[10] M. Nishigaki, C. Fermuller, and D. DeMenthon. The image torque operator: A new tool for mid-level vision. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 502–509, June 2012.

[11] C.L. Teo, A. Myers, C. Fermuller, and Y. Aloimonos. Embedding high-level information into low level vision: Efficient object search in clutter. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 126–132, May 2013.

[12] Christof Koch and Shimon Ullman. Shifts in selective visual attention: Towards the underlying neural circuitry. In LuciaM. Vaina, editor, *Matters of Intelligence*, volume 188 of *Synthese Library*, pages 115–141. Springer Netherlands, 1987.

[13] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(11):1254–1259, Nov 1998.

[14] Abel Gonzalez-Garcia, Alexander Vezhnevets, and Vittorio Ferrari. An active search strategy for efficient object detection. *CoRR*, abs/1412.3709, 2014.

[15] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[16] Varun K. Nagaraja, Vlad I. Morariu, and Larry S. Davis. Searching for objects using structure in indoor scenes. In *British Machine Vision Conference (BMVC)*, 2015.

[17] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 2204–2212, Cambridge, MA, USA, 2014. MIT Press.

[18] Yanwei Pang, Jiale Cao, and Xuelong Li. Learning sampling functions for efficient object detection. *CoRR*, abs/1508.05581, 2015.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In Cordelia Schmid, Stefano Soatto, and Carlo Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[20] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2249–2257. Curran Associates, Inc., 2011.

[21] Daniel Russo and Benjamin Van Roy. Learning to optimize via information directed sampling. *CoRR*, abs/1403.5556, 2014.

[22] Dinesh Jayaraman and Kristen Grauman. Look-ahead before you leap: end-to-end active recognition by forecasting the effect of motion. *CoRR*, abs/1605.00164, 2016.

[23] A. Richtsfeld, T. Mrwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4791–4796, Oct 2012.

[24] Aleksandrs Ecins, Cornelia Fermüller, and Yiannis Aloimonos. Cluttered scene segmentation using the symmetry constraint. In *ICRA*, 2016.

[25] W. Luan, R. Mao, and J. S. Baras. Active sampling exploiting detector response pattern for efficient target detection. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1916–1922, July 2016.

[26] Xiaodong Yu, C. Fermller, Ching Lik Teo, Yezhou Yang, and Y. Aloimonos. Active scene recognition with vision and language. In *2011 International Conference on Computer Vision*, pages 810–817, Nov 2011.

[27] Wentao Luan, Yezhou Yang, Cornelia Fermüller, and John S. Baras. *Reliable Attribute-Based Object Recognition Using High Predictive Value Classifiers*, pages 801–815. Springer International Publishing, Cham, 2016.

[28] Ziang Xie, Arjun Singh, Justin Uang, Karthik S. Narayan, and Pieter Abbeel. Multimodal blending for high-accuracy instance recognition. In *Proceedings of the 26th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[29] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations online for robust visual tracking. *IEEE Transactions on Image Processing*, 25(4):1834–1848, April 2016.

[30] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. A global hypotheses verification method for 3d object recognition. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12, pages 511–524, Berlin, Heidelberg, 2012. Springer-Verlag.

[31] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary R. Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision - ACCV 2012 - 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I*, pages 548–562, 2012.

[32] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3212–3217, May 2009.

[33] Radu Bogdan Rusu, Gary Bradski, Romain Thibaux, and John Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, 10/2010 2010.

[34] Asako Kanezaki, Zoltan-Csaba Marton, Dejan Pangercic, Tatsuya Harada, Yasuo Kuniyoshi, and Michael Beetz. Voxelized Shape and Color Histograms for RGB-D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World*, San Francisco, CA, USA, September, 25–30 2011.

[35] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Mining and-or graphs for graph matching and object discovery. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[36] Kamesh Munagala, Utkarsh Srivastava, and Jennifer Widom. Optimization of continuous queries with shared expensive filters. In *Proceedings of the Twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '07, pages 215–224, New York, NY, USA, 2007. ACM.

[37] Haim Kaplan, Eyal Kushilevitz, and Yishay Mansour. Learning with attribute costs. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 356–365, New York, NY, USA, 2005. ACM.

[38] Daniel Golovin and Andreas Krause. Adaptive submodularity: A new approach to active learning and stochastic optimization. *CoRR*, abs/1003.3967, 2010.

[39] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[40] R. Mao, Y. Yang, C. Fermüller, Y. Aloimonos, and J. S. Baras. Learning hand movements from markerless demonstrations for humanoid tasks. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 938–943, Nov 2014.

[41] N. Atanasov, B. Sankaran, J. Le Ny, T. Koletschka, G. J. Pappas, and K. Daniilidis. Hypothesis testing framework for active object detection. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4216–4222, May 2013.

[42] K. Wu, R. Ranasinghe, and G. Dissanayake. Active recognition and pose estimation of household objects in clutter. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4230–4237, May 2015.

[43] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[44] X. Liu and J. S. Baras. Trust-aware crowdsourcing with domain knowledge. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2913–2918, Dec 2015.

[45] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International Journal of Computer Vision*, 1(4):333–356, 1988.

[46] Sven J Dickinson, Henrik I Christensen, John K Tsotsos, and Gran Olofsson. Active object recognition integrating attention and viewpoint control. *Computer Vision and Image Understanding*, 67(3):239 – 260, 1997.

[47] Richard Pito. A solution to the next best view problem for automated surface acquisition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(10):1016–1030, October 1999.

[48] B. Browatzki, V. Tikhanoff, G. Metta, H.H. Bulthoff, and C. Wallraven. Active object recognition on a humanoid robot. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 2021–2028, May 2012.

[49] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429 – 446, 2004.

[50] Michael A. Goodrich and Alan C. Schultz. Humanrobot interaction: A survey. *Foundations and Trends in HumanComputer Interaction*, 1(3):203–275, 2008.

[51] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie Shah. Efficient model learning from joint-action demonstrations for human-robot collaborative tasks. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, pages 189–196, New York, NY, USA, 2015. ACM.

[52] Vaibhav V. Unhelkar, Ho Chit Siu, and Julie A. Shah. Comparative performance of human and mobile robotic assistants in collaborative fetch-and-deliver tasks. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction*, HRI '14, pages 82–89, New York, NY, USA, 2014. ACM.

[53] Pannagadatta K. Shivaswamy and Thorsten Joachims. Online structured prediction via coactive learning. *CoRR*, abs/1205.4213, 2012.

[54] Ashesh Jain, Brian Wojcik, Thorsten Joachims, and Ashutosh Saxena. Learning trajectory preferences for manipulators via iterative improvement. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 575–583. Curran Associates, Inc., 2013.

[55] Karthik Raman, Pannaga Shivaswamy, and Thorsten Joachims. Online learning to diversify from implicit feedback. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 705–713, New York, NY, USA, 2012. ACM.

[56] Artem Sokolov, Stefan Riezler, and Shay B. Cohen. Coactive learning for interactive machine translation. In *MLIS@ICML*, 2015.

[57] Wentao Luan, Yezhou Yang, Cornelia Fermuller, and John S. Baras. Fast task-specific target detection via graph based constraints representation and checking. 2015.

[58] M. Lutz, D. Stampfer, and C. Schlegel. Probabilistic object recognition and pose estimation by fusing multiple algorithms. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4244–4249, May 2013.

[59] Yasir Salih, Aamir Saeed Malik, Nicolas Walter, Désiré Sidibé, Naufal Saad, and Fabrice Meriaudeau. Noise robustness analysis of point cloud descriptors. In *15th International Conference on Advanced Concepts for Intelligent Vision Systems - Volume 8192*, ACIVS 2013, pages 68–79, New York, NY, USA, 2013. Springer-Verlag New York, Inc.

[60] Tianfu Wu and Song-Chun Zhu. Learning near-optimal cost-sensitive decision policy for object detection. In *PAMI*, 2014 (to appear).

[61] Yong Xu, Hui Ji, and C. Fermuller. A projective invariant for textures. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1932–1939, 2006.

[62] M. Bertsche, T. Fromm, and W. Ertel. Bor3d: A use-case-oriented software framework for 3-d object recognition. In *Technologies for Practical Robot Applications (TePRA), 2012 IEEE International Conference on*, pages 67–72, April 2012.

[63] M. Attamimi, A. Mizutani, T. Nakamura, Takayuki Nagai, K. Funakoshi, and M. Nakano. Real-time 3d visual sensor for robust object recognition. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4560–4565, Oct 2010.

[64] Paul Baumstarck Stephen Gould and Morgan Quigley. Integrating visual and range data for robotic object detection, 2008.

[65] Alvaro Collet Romea, Manuel Martinez Torres, and Siddhartha Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *International Journal of Robotics Research*, 30(10):1284 – 1306, September 2011.

[66] T. Fromm, B. Staehle, and W. Ertel. Robust multi-algorithm object recognition using machine learning methods. In *Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on*, pages 490–497, Sept 2012.

[67] A.M. Naguib and Sukhan Lee. Adaptive bayesian recognition with multiple evidences. In *Multimedia Computing and Systems (ICMCS), 2014 International Conference on*, pages 337–344, April 2014.

[68] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824, May 2011.

[69] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 509–516, May 2014.

[70] Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.