LOW DIMENSIONAL VS POINT-WISE COMPLEX OBJECT DETECTION, REFINEMENT AND TRACKING

by

Mehmet Kemal Kocamaz

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer and Information Sciences

Spring 2014

LOW DIMENSIONAL VS POINT-WISE COMPLEX OBJECT DETECTION, REFINEMENT AND TRACKING

by

Mehmet Kemal Kocamaz

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____________________________________

Christopher E. Rasmussen, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____________________________________

Chandra Kambhamettu, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____________________________________

Jingyi Yu, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: ____________________________________

Herbert Tanner, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

I would like to express my immense gratitude to my advisor Professor Christopher Rasmussen for his invaluable guidance over the course of time I was in University of Delaware. The valuable discussions with him all through these years have brought the best in me, both academically and in developing my professional abilities. He did not only provide me with the freedom to find my own way in the research but also guides me to stay on track. He was always available to me and willing to spend enormous of time to mentor me. This dissertation would have remained a dream without his support. The motivation he provided, his enthusiasm for the research, and his respect to his students have made a deep impression on me.

I would like to thank Professor Chandra Kambhamettu, Professor Jingyi Yu, and Professor Herbert Tanner for serving on my advisory committee. They have given me invaluable comments and suggestions on both my study and my research, moral support, and provided wise career advisement.

I would like to thank MERL for providing a wonderful internship experience. It was great pleasure to work with Professor Fatih Porikli, Dr. Yuichi Taguchi, Dr. Srikumar Ramalingam, Dr. Tyler W. Garaas, Dr. Oncel Tuzel, and Dr. Jay Thornton. Valuable discussions with each of them enhanced my research ideas. Special thanks to Professor Fatih Porikli for his support, motivation, advices about my PhD study and career path.

I would also like to thank other members of UD Dynamic Vision lab (Yan Lu, Brad Fletcher, Qiaosong Wang, Conor Gilsenan, Kevin Graney) for sharing their thoughts, codes and providing such an enjoyable and supportive environment during my Ph.D. study. Thanks to Alph for coffee breaks during writing this thesis. Thanks to all of my friends in UD, I enjoyed the time that I spent with everyone of you.

I owe my deepest gratitude to my family. This thesis would have never been written without their support, love and patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this dissertation, I focus on algorithms to detect, refine, and track complex visual objects, as determined by their shape, appearance, or range of motion. Two different classes of representations, namely low-dimensional and point-wise (non-parametric), are explored, compared, and analyzed by combining multiple features from both imaging and depth sensors. Such perception modules may provide crucial decision-making and planning information for intelligent systems, and we show links to several robotic application areas.

On one side, a multimodal human detection algorithm which utilizes multiple sensor data sources is detailed. The human is represented in a low-dimensional space, and a ladar-camera architecture is constructed which with visual and geometric cues improves on the detection rate and speed of conventional human classifiers. Unlike existing approaches, the proposed human classifier does not make any restrictive assumptions on the range scan positions, and thus is applicable to a wide range of real-life detection tasks.

On another side, a modified graph cut-based method is proposed to refine complex objects–e.g. human hands, hiking trails for navigation, and household objects–from a rough estimate of the object pose given by a low-dimensional shape detector or tracker. The standard graph cut method is modified to incorporate color and shape distance terms, adaptively weighting them at run time to favor the most informative cue in different visual conditions. Also, this method is extended for point-wise tracking of the objects through iterative refinement without estimation of the object displacement over time.

Moreover, a novel algorithm which combines low- and high-level observations in a graph cut framework is developed for the purpose of refining low-dimensional representations of the human shape provided by some human detection or tracking methods. A multi-layer graph cut framework is introduced to combine low-level observations such as color and depth with high-level cues including the estimated ground plane and point-wise shape confidence scores given by a classifier.

Lastly, a point-wise tracking algorithm which fuses multiple cues obtained from depth and color cameras is developed. It employs a motion estimation technique in which displacements are calculated from the 3-D locations of image keypoints matched between frames. Color and point-wise shape cues are combined in the same feature vector to allow adaptive weighting in different parts of the scene. This tracker is generic in that it does not make any assumptions or restrictions about the shape of the object, and experiments demonstrate better dense point-wise tracking performance than comparable algorithms over large viewpoint changes and object deformations.

Chapter 1
INTRODUCTION

In this thesis, we focus on the perception algorithms to detect, refine, and track the complex objects using two different representations, namely low dimensional and point-wise (non-parametric), by combining multiple features from a single or multiple sensors. The differences between two groups of algorithms which represent the complex objects in low dimensional space and point-wise are explored, compared and analyzed. This chapter first makes an introduction to the complex object perception, and defines the object *Refinement* concept. Second section explains low dimensional and point-wise object representations, their advantages and disadvantages. Then, the importance of incorporating multiple features in the perception systems is outlined in the next section. Finally, it describes the thesis statement, contributions and the outline of the work.

1.1 Complex Object Perception

According to the dictionary definition, *complex* means something complicated, hard, and intricate in the structure [63]. In the computer perception field, the characteristics of a complex object can be listed as follows: It is difficult to parameterize and represent in a general form. It might be non-rigid, not in one particular shape, capable of being reshaped by external or internal forces and effects. It might be formed multiple non-similar parts. These parts might be articulated. Also, it might be rigid, but it is formed by many vertices and difficult to represent by several basic shapes. The shape, the degree of freedom, or the range of motion can determine the complexity of an object. There are many complex objects on the earth and easy to see them around us. First of all, we all the humans are complex. A cable, animals, a human hand, head, lungs, some biological cells, clothes, towel, trails, trees, Statue of Liberty and many other objects can be categorized as complex. Their complexities generate some challenges in the perception of these objects by the computers and robots. Figure 1.1 shows some examples of complex objects.

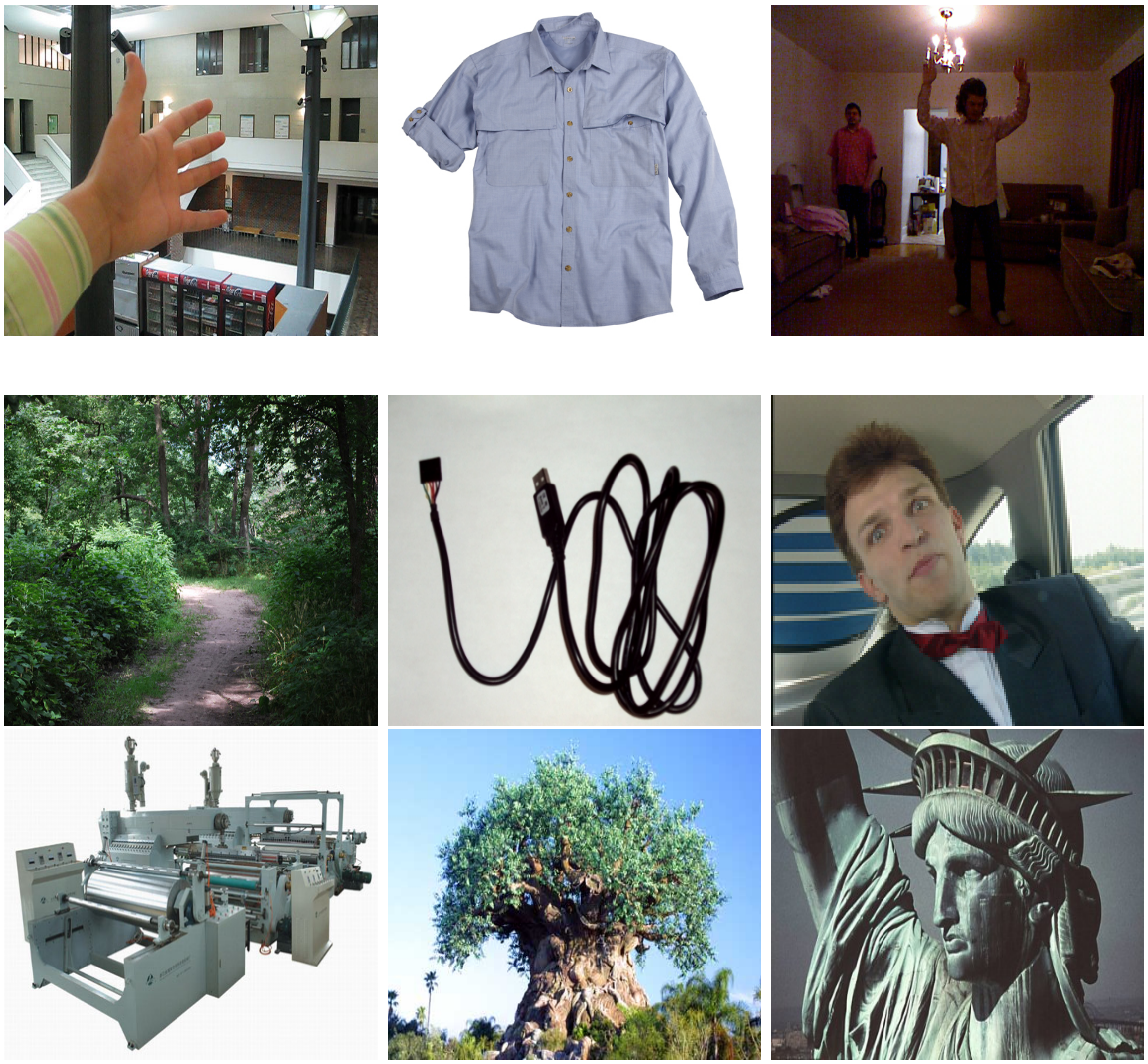


Figure 1.1: Complex objects examples. One complex object is illustrated in each image. A human hand, shirt, person, trail, cable, human face, paper coating machine, tree, and Statue of Liberty. Paper coating machine and Statue of Liberty are rigid objects, but their shape and structure are complicated.

Seeing, detecting, tracking and understanding the behaviors of the complex objects are so important tasks for the computers and the robots. These all goals construct the perception modules of an intelligent system. Perception modules are the heart of all intelligent systems. Their next acts, movements and the decisions are made based on the information provided by the perception modules. Since there might be many complex objects in the interest of the intelligent system, they need to be handled specially.

Complex object perception algorithms have a rich variety of usages, some of the applications are:

(1) Robot vision: The robots need to know its surrounding environment, so seeing the objects around itself is so important to accomplish its task[29] [23]. For example, an unmanned ground vehicle needs to track the road to drive autonomously [35]. It may need to detect a human to listen to his/her commands or to avoid from him/her not to hit [64] [134].

(2) Video compression and indexing: Object detecting and tracking is a part of some video compression algorithms and multimedia retrieving process from databases [131] [162].

(3) Automated surveillance: The perception system is used to monitor the object movements, people and events in the public or private places, such as in front of a building, a shopping mall, a road, car parks, etc. The system needs to distinguish the interest of the object, must detect and track it [75] [184] [74] [32].

(4) Medical data analysis: An organ, a tumor in the body of the human must be segmented and tracked in the medical images to be analyzed by a doctor [76] [5] [204].

(5) Interactive games and the computer aided programs: They need to detect and track a human body, or track a human hand to take the necessary inputs to the software system [165] [174].

In addition to well known and defined detecting, segmentation and tracking concepts in the object perception literature, *Refinement* is a complementary task in the perception modules. *Refinement* can be defined as the purpose of obtaining the exact border of an object from its given rough representation. For example, someone can provide a box which includes most part of the object and some background, then he can desire to label all object pixels in the image. The rough representation includes hints about the location and features belong to the object. In contrast to object segmentation, the *refinement* process might bound the region of interest. It captures its initial feature models from this region of interest.

1.2 Object Representation Approaches

Current object detection, segmentation and tracking approaches can be divided into two main groups, as illustrated in Figure 1.2. These two groups are defined and explained in the following sections.

1.2.1 Representation in Low Dimensional Space

In this category, no matter it is a detection, segmentation or a tracking method, the object is represented in low dimensional space. The object is considered as a rough shape. This approach does not represent the object in fully detail. The borders of the object cannot be stated in point-wise. For example, if a person in an image is tried to be detected, a bounding box, *B(w,h,x,y)*, represents the person. *B(w,h,x,y)* has 4 dimensions which are box height, box width, its top x, and top y locations in the image. The algorithm determines the person, it fits the bounding box around it and outputs the dimensions of this box. In the same way, the object state is the bounding box in a tracking procedure. This low dimensional state is estimated and updated during the tracking. Low dimensional representation helps to reduce the complexity of the tasks, but not enough some type of the jobs which require knowing object borders in point-wise. 1.3 shows the low dimensional representations of some objects in the image and 3d point cloud.

As illustrated in the top row of Figure 1.2, there are three paths followed by the perception algorithms to achieve the final low dimensional representation of an object.

First way is simply one frame object detection methods. The purpose is to find the object, if any exists, in a given scene and to determine its location. The scene might contain challenging background objects. Common approach is to train a classifier to distinguish and locate the object of interest from the background [176], [36], [143], [7]. A window is slided all over the image. Some features are obtained within this window to form a descriptor. This descriptor is asked to the classifier to decide whether there is a searched object inside the window or not.

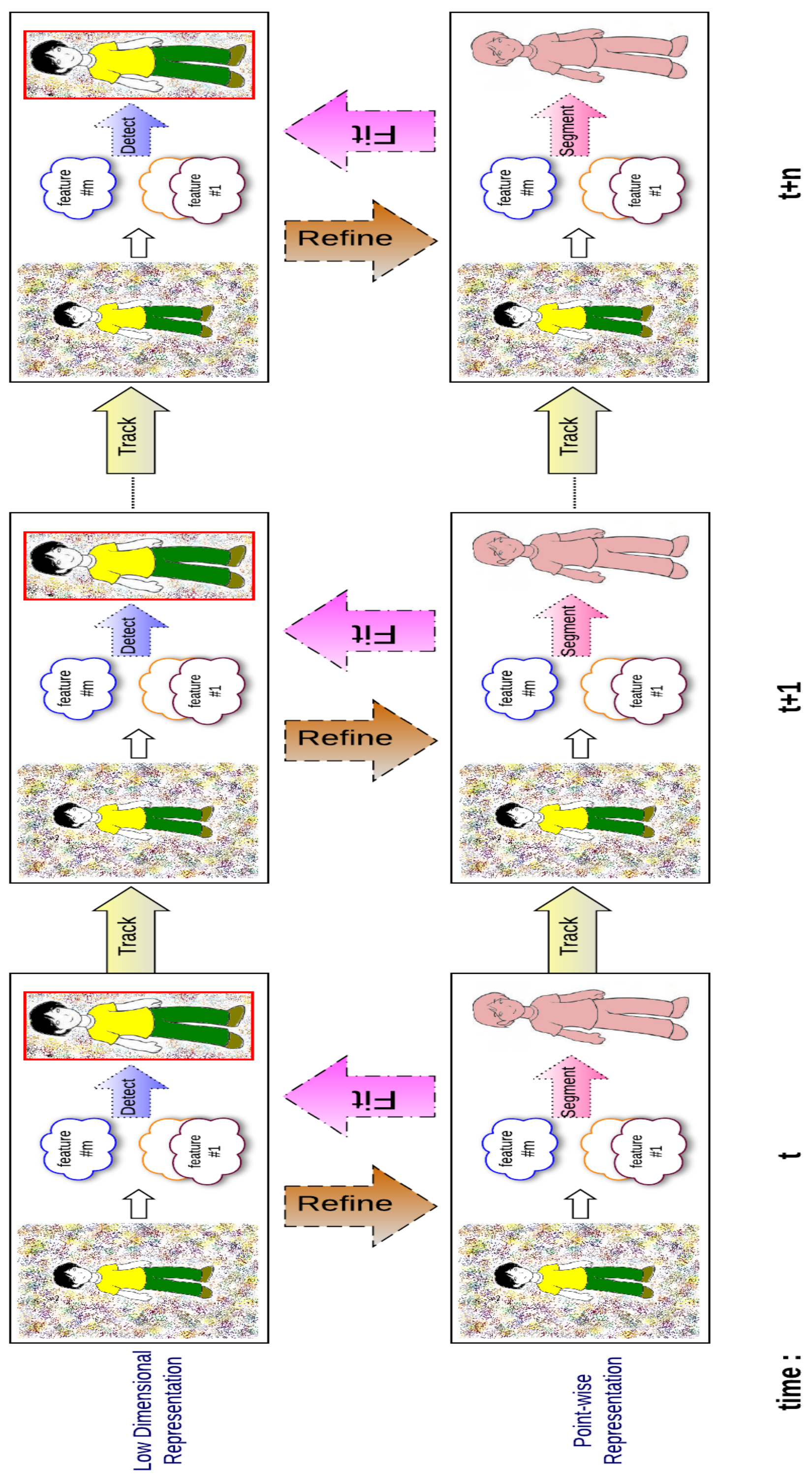


Figure 1.2: The diagram which shows different type of perception modules.

Second path which produces its results in low dimensional space is the tracking process, as can be seen in the first row of Figure 1.2. At each frame of the procedure, the next state of the object is estimated according to the features obtained from the object and the current state. The states are defined in the low dimensional space. Some methods can utilize from the multiple features. *Kalman Filter* [79], and its extended versions [78] [77] [181] became main this type of object tracking principle for years. *Kalman filter* is weak at the problems in which the posterior of the tracked object needs to be multimodal. *Particle Filter* is designed to handle this kind of problems. *Particle Filters* was born at early 1950s. Its first mathematical methods was described in [113]. It became one of the most popular object tracking methods in last two decades. *Particle Filter* has been highly started to applied in computer vision problems after it was showed in [9] [72] which can be used to track objects robustly in the image sequences.

Another perception task whose final result in the low dimensional space is called as *Fitting*. It is usually an intermediate step to pass from the point-wise to the low dimensional representation. It can be utilized to combine two types of methods in one mechanism. For example, if any algorithm outputs the object as a point-wise mask, an object model can be fit on this mask; hence a low dimensional representation is achieved. Then, this object representation can be fed to a *Particle Filter* type tracking algorithm.

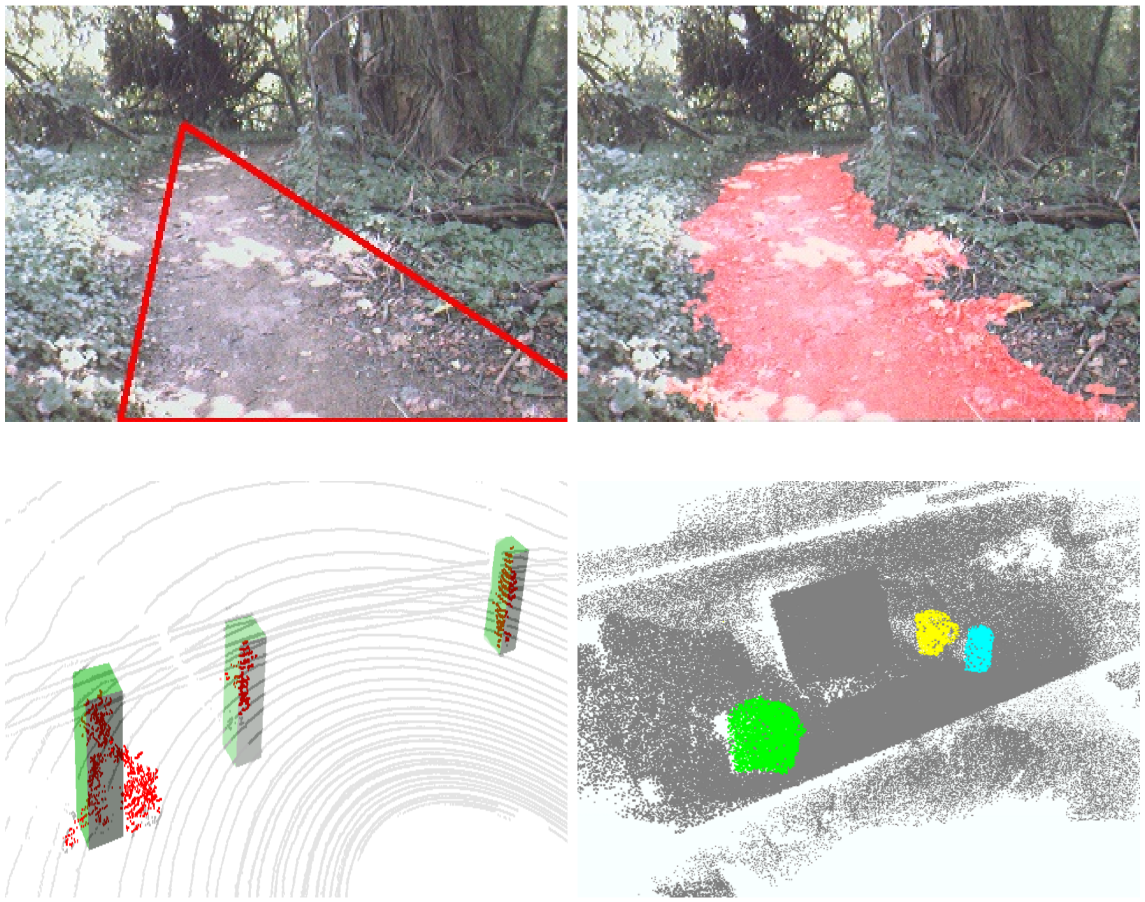


Figure 1.3: First row shows the low dimensional and point-wise representations of a trail in an image. Second row displays the low dimensional representations of the pedestrians and point-wise representations of several objects (A coffee mug, a soda can, and a hat) [97] [171].

1.2.2 Point-wise (Non-Parametric) Representation

This group of perception methods does not represent the object in parametric way. The object is defined as a mask consisting of the points. They output the border of the object in fully detail. Point-wise representation is more desirable for some applications, such as gesture recognition, human behavior analysis, robot motion planning, and etc. Knowing the every detail of the object improves the performance of these type of applications. 1.3 displays some examples of point-wise representation of the objects in the image and 3d point cloud.

There are three types of perception procedures whose output are point-wise representations of the objects as displayed in the bottom row of Figure 1.2. The first type is a *Refinement* process. In this process, rough shape of the object in the low dimensional space is provided [86], such as a bounding box around a detected human in the image. Then the *Refinement* process automatically segments and generate point-wise representation of this human as a mask. Second type of the processes can be considered as unconstrained segmentation methods. In contrast to *Refinement* procedure, they produce global results without restricting region of the interest. They usually employ prior knowledge or learned model of the object.

There are some object tracking approaches whose output are the silhouette or the mask of the object. This type of the methods can be considered as the approaches which represent the object in point-wise. For example, active contours model is a famous method which was first introduced in [81] and applied intensely in the tracking problems which requires exact representation of the object borders. This method, called as *Snakes*, attempts to adjust the initialized contours on the borders of the object in an iterative framework. An energy function is constructed and associated to the contour. Minimum value of the energy function represents the object borders. This energy function is minimized iteratively in the *Snakes* framework. The final position of the contour draws the object border.

Graph cut is really powerful technique to segment and track the objects by representing them point-wise [16]. Graph cut has three main advantages which make it unique and popular in image segmentation problems. First, it does not get stuck in local minima, so it produces globally optimal segmentations. This property of the graph cut is so important in a case that if the location of desired object in the image is not known and specified, but the algorithm is wanted to segment it. Even if the object is spread out two different unconnected regions in the image, graph cut is be able to segment and label them as the same object parts. One of the other unique point of graph cut is that its energy minimization framework is *discontinuity-preserving*. It means that the interactions between the graph nodes, simply considered as the pixels in the image, can vary over entire image. Since these interactions are different on the border of the object than in the rest of the image, unlike some other energy minimization systems, graph cut segmentation is expected to be more gentle and accurate on the border of the objects. This yields better segmentation results than the approaches which assumes that the image is smooth everywhere. Another advantage is that graph cut algorithm runs so fast. If the object resolution is not so high, nearly real time performance can be caught by today's powerful computers. Also, if the computational time is really a big concern, graph cut has its GPU version implementations, so that high resolution images can be processed. More background about graph cuts is explained in Chapter 3.

1.3 Complex Object Perception by Integrating Multiple Features

The perception algorithms use features to detect and distinguish the object from the background. Feature selection plays a key role in the accuracy and performance of the perception modules. The most distinctive feature specific to the object is chosen from the feature space and integrated into the process. In most cases, the object can be discriminated from its surrounding environment by its one of low level features, such as intensity, color, edges, texture, height, depth, contrast, etc. The computer vision algorithms usually uses the visual features (Intensity, color, texture, and edges) to represent the object. If good stereo retrieved information is available, also depth information can be feed to the process. Perception algorithms can be divided into two groups: The algorithms that uses single feature, and the approaches which integrate multiple features into its structure.

Single feature algorithms choose low level features and maintain it over the process. Their implementation and methods usually are simple. Color is the most used cue in most of the approaches that process camera images. Single feature based some of the deformable object tracking algorithms are presented in [34], [200], [148], [33], detection algorithms explained in [36], [186], [133] and the segmentation algorithms mentioned in [120], [67], [139]. In single feature approaches, the selected feature must be really discriminative for the robustness of the method. If the color wants to be used as incorporated feature to accomplish a task, the color of the object and the background in the image sequences should not be red. The object representation needs to be stable in that feature domain space over the sequences of the sensor information. Sudden changes in the object definition in the represented domain during the process breaks down the task inevitably.

Using single feature is not enough to construct a robust object perception algorithm in some cases. Many approaches fuse multiple features in their framework to reinforce the process. They start by deciding good features to be integrated into the system. Good features means the best object specific representation. This step involves careful observation of the object and its background. After selecting the features, the following question arises: How the chosen features are going to be combined? There are numerous of different approaches at this point. Weighting used features in the method is highly common approach. They can be weighted manually by a human at the initialization of the algorithm, or dynamically by the system. Another popular approach is to get help from machine learning algorithms. The feature data set is trained by a learning algorithm. The weights of the features are asked to the trained algorithms at the runtime of the system.

In most of the cases, high number of available good features brings better and robust perception results. Since a single sensor can provide limited number of features, using multiple different sensors assure more features which can be incorporated. For example, it is difficult to obtain the depth, height or planar information of an object from a single camera. Even stereo cameras do not always provide sustainable 3D features of an object over the image sequences. Therefore, usage of multiple different sensors is necessary for a powerful system. For example, some of the robots carry lidar, radar, sonar, IR cameras in addition to color cameras. This type of integrated system utilize from different information of its environment provided by its different sensors.

The advances in the technology has facilitated the availability of cheap sensors in the market for last ten years, such as IR cameras, GPS system, and IMUs, etc. Hence, the robots become not the only systems which include multiple sensors, but other devices start to include these sensors. For example, cell phones start to include GPS and IMUs. It is really important for the vision community that IR cameras start to show up in home devices, such as in Kinect game console by Microsoft. The sensors of Kinect has not been only used in the game consoles, but has taken the attention of the researchers. It became popular in vision and robotics community to solve field related problems by using Kinect obtained data.

I believe in that we are going to see more integrated devices which will include multiple different sensors as Microsoft Kinect in the future. The availability of multiple features obtained from these devices will provide opportunities to solve complex and challenging vision and robotics problems. Therefore, the methods to integrate multiple features in efficient way will play important role in the technology.

1.4 Thesis Statement and the Purpose

Integrating multiple features from one sensor, or if available from multiple sensors, provides more accurate and robust complex object perception algorithms. On one side, one of the objectives is to develop methods to refine and track complex objects in point-wise by incorporating multiple features from multiple sensors into the graph cut framework. These methods are expected to be robust to different illumination conditions, shape or pose changes and to work in cases in which the problem is not solvable by using only one cue, so they utilize from different cues adaptively and jointly.

Another purpose is to develop multimodal human detection algorithm which utilize from multiple sensor data. The human is represented in the low dimensional space in this method. Multiple sensor architecture is constructed to obtain multiple features from different sensors and to increase the detection rate of the human classifier. This architecture aims to supply complementary and redundant information to enhance the confidence level of the detections, so this approach increases the system reliability.

1.5 Thesis Contributions

This thesis makes the following contributions into the field:

Novel multimodal human classifier: An accurate and computationally very fast multi-modal human detector is proposed [88]. This 1D+2D detector fuses 1D range scan and 2D image information via an effective geometric descriptor and a silhouette based visual representation within a radial basis function kernel support vector machine learning framework. Unlike the existing approaches, the proposed 1D+2D detector does not make any restrictive assumptions on the range scan positions, thus it is applicable to a wide range of real-life detection tasks. To analyze the discriminative power of the geometric descriptor, a range scan only version, 1D+, is also evaluated. Extensive experiments demonstrate that the 1D+2D detector works robustly under challenging imaging conditions and achieves several orders of magnitude performance improvement while reducing the computational load drastically.

Trail tracking in low dimensional space: Several trail tracking algorithms for autonomous outdoor robot navigation are developed [149] [150] [151] [153] [154]. These algorithms maintain low dimensional representations of the trail over the image sequences. In one framework, the trail region is modeled as a triangle if the color camera is used. In other frameworks which utilize from omni-directional color camera data, a circular arc segment of constant width represents the trail. The visual cues, ladar and stereo camera derived structural information are combined in the frameworks. The low dimensional states of the trail are estimated via *Particle Filter*.

Refinement of low dimensional shapes of complex objects: This thesis highly examines *refinement bridge* between low dimensional and point-wise representations of complex objects in the literature. Large datasets of different kinds of the objects, such as trail, human hand, and household objects in front of different, complex background, under different illumination conditions, and poses are addressed to be refined. Different possible low dimensional representations are considered during the experiments. For example, the cases in which the representation covers some large amount of the background or it is highly includes object points are discussed and different methods are proposed. The problem of *object refinement* is handled by various ways, such as incorporating multiple cues from different sensors, with or without prior information, structure, shape and color related cues.

Refinement of complex objects with graph cut framework: The power of the graph cut framework was analyzed for *Refinement* of the objects in the images. The suitability of several previously published segmentation algorithms were compared in terms of both the agreement with ground truth and the speed on a range of trail images with diverse appearance characteristics. These algorithms include generic graph cut, a shape-based version of graph cut which employs a distance penalty, GrabCut, and an iterative superpixel grouping method. Some modifications are made in the algorithms to make them work better for the recorded data sets. It is concluded that graph cut is really a powerful tool which can serve as a refinement method if a rough estimation of the object is given by a low dimensional shape tracker. This work is published in [86].

Improvements on standard graph cut method: Standard graph cut method was modified to incorporate color and shape distance terms, also adaptively weight them at run time to favor the most informative cue in different visual conditions. Furthermore, single-frame refinement method is extended to serve as the basis of the tracker which works for a variety of object types with complex, deformable shapes in complex background. Some basic adaptive weighting method is investigated to incorporate the shape distance and the color features. This weighting method clusters the pixels in the image and measure the distance between object and background models by creating color histograms. It is published in [87].

Low Dimensional Human Shape Refinement: A novel algorithm which combines low and high level observations in the graph cut framework is developed for the purpose of refining low dimensional representation of the human shape. This representation is a generic shape which is provided by some other human detection or tracking methods. The proposed algorithm does not utilize from internal computations of those methods which provide the estimated shape. Therefore, it is generic and applicable for any kind of method which produce a similar human shape. For this purpose, a novel point-wise shape descriptor is developed. This powerful descriptor utilizes from the depth information of the scene by generating several shape related cues, such as relative geodesic distance, vectorial spatial distance of a point to the middle point of given estimated rough representation. Furthermore, it incorporates the normal of a point. The descriptors are trained by Random Forest algorithm which enable to favor most important cue in the descriptor. In order to combine multiple low and high level observed cues jointly, such as point-wise shape confidence scores, the color and estimated ground plane, multi-layer graph cut framework is introduced for this purpose. The proposed method outperformed existing suitable algorithms in the experiments.

Moreover, the same problem is tried to be solved by extending previously developed graph cut method which does not incorporate prior information, but only fuses generic cues obtained from the single test image.

Generic multimodal graph cut tracker: A graph cut based point-wise tracking algorithm which combines multiple cues obtained from depth and color cameras is developed. This tracker does not make any assumptions and restrictions about the shape of the object, so it is not a specific object tracker, but it is generic for all object types. It employs a location estimation technique in which SIFT keypoints are matched between the frames. In order to make it robust, it calculates the displacements of the points in 3D Euclidean space. This tracker generalizes the shape cues developed for the purpose of human refinement. It fuses the color and point-wise shape cues in the same feature vector to allow weighting the cues locally around the object. The proposed tracker is experimented with several datasets which includes complex and deformable objects. It performs better than compared tracking algorithms which are in the similar category allowing dense point-wise tracking of the objects.

1.6 Thesis Outline

The details of the proposed multimodal human detection algorithm is explained in Chapter 2. The existing methods which obtain cues from single sensor or fuse features extracted the data of multiple sensors are outlined in this chapter. The sensor setup, computation of the human descriptor, the details of *DontHitMe* dataset, and the experiments are described.

The basics of graph cut segmentation method is explained in Chapter 3. The background, different application fields of the graph cut, the improvements and its extensions are explored in this chapter. Furthermore, the priors, the cues and possible adjustment techniques of the parameters in graph cut formula are mentioned.

Chapter 4 makes an introduction to the refinement of complex objects such as, trail, a human hand and head by analyzing the performance of modified graph cut based algorithms for this purpose. It proposes a new improved graph cut based refinement algorithm for the complex objects by combining spatial distance penalty depends on the object shape, and adaptively changing between most beneficial color spaces. Moreover, the results of the proposed method is presented for the household objects.

Different approaches are proposed to refine given low dimensional human shape in Chapter 5. These approaches are categorized in two groups which use generic cues only available in the current processing image without fusing the prior information, and the other method which fuse prior shape information. A novel point-wise shape descriptor is introduced. Also, the details of constructing multi-layer graph to incorporate the low and high level observations, in this case it is an estimated ground plane, is explained.

In Chapter 6, possible approaches for point-wise object tracking is discussed. On one side, previously developed graph cut based refinement methods is extended to track the objects which are deformable and whose displacement between the frames is so small. This method does not estimate the location of the object. On another side, a multimodal point-wise tracker which utilizes from multiple sensor data is proposed. In contrast to previous tracker, it estimates the displacement of the object by computing the keypoint matches between the frames. Also, it learns the shape and the color of the object on the fly.

The possible extensions of the proposed methods in this thesis, and the future work in this field is discussed in Chapter 7.

Chapter 2
LOW DIMENSIONAL MULTIMODAL HUMAN DETECTION

Perception modules are the heart of the intelligent vehicles which provide the information about the surrounding environment of the robot. The next acts of the robots are determined based on the types, localizations, arrangements, poses of the objects around them. Humans are considered as special type of the objects, especially for the robots which interact with them, work in proximity to the humans, or try to avoid from hitting them. Humans are living creatures, so they more important than any other regular obstacles. Knowing the positions, motions, poses and presence of the humans enable the robots to better move and react to their actions.

According to National Highway Traffic Safety Administration reports [48], thousands of pedestrians die in the traffic accidents only in USA. At least same numbers of pedestrians get injured because of the car hits, some of them become handicapped as the results of these hits. Since human life is most important value on the earth, intelligent outdoor systems should have the ability to detect and localize the humans to reduce the accidents in which a human might be injured or died. Detecting the humans is not only important for outdoor robots, it is also crucial part of indoor robot systems which interact to the humans, work closely to them, and wait for their commands must be able to detect the humans.

Human detection is a complicated and difficult task for the intelligent systems. There are many various challenges which make this task hard. These challenges can be categorized in two main groups. First group is formed by the external, also can be called as environmental, factors. These factors are not object depended, but they are the challenges caused by the entities which surrounds a human. Illumination conditions, night/day times, insufficient lighting, shadows, headlight of the cars on the roads fits into this category and have absolute effects to the performance of the detection process. Even, weather conditions, such as snow and rain, might create different pitfalls by generating noise in the scene, so they must be handled accordingly. Partial occlusion and dynamic objects in the background are other types of external factors which might require special attention.

Second group of challenges are due to the human itself, thus may be called as the internal factors. A human consists of different body parts. Some body parts, such as a hand, foot, and whole body can deform, so they might appear in different shapes. For example, they can sit, walk, run, bend to take something on the floor, and make some gestures. Also, a human can have various poses from distinct view points. Their height and weights can differ from one to another. Their body colors and hair lengths change. Their different clothes augment their shape variations. As an example, someone can wear a long skirt and a coat which cover his/her legs and arms. These all factors complicate the human detection objective.

Human detection methods are divided into two main groups, namely single modal, and multimodal. Single modal human detection algorithms utilize from just one sensor data. Color camera is the most common device used in the intelligent systems for single modal detection algorithms. Huge amount of research has been conducted on single modal human detection in last decades. Even though some of these algorithms perform very well, they still suffer from the sensor specific limitations. For example, color camera based single modal detection methods fail or does not reach the expected accomplishment level under different visual conditions. Therefore, multiple sensor architecture is constructed to increase the detection rate of the classifier. The methods that process and integrate multiple sensor data to increase the system reliability and redundancy are called as multimodal human detection algorithms. In these approaches, complementary and redundant information are supplied from the multi sensor architecture to enhance the confidence level of the detections.

A multimodal human detection classifier which fuses LIDAR (Light Detection and Ranging) sensor and color camera data is developed. This algorithm integrates

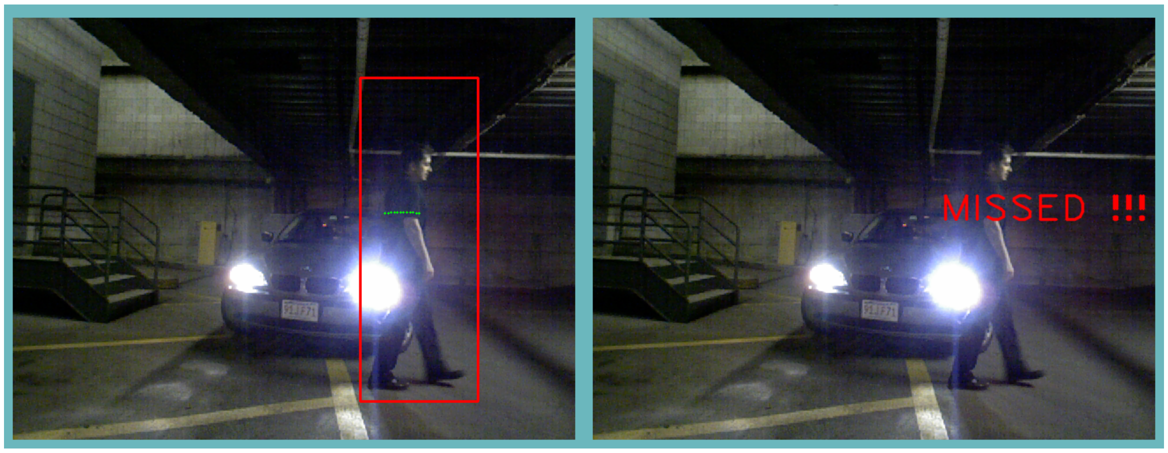


Figure 2.1: Left image shows the detected pedestrian by the proposed multi-modal human classifier under severe illumination conditions. Single-modal classifiers [36] (and conventional multi-modal approaches) are not be able to detect as can be seen in the right image.

the features obtained from both of the sensor data in a joint way. It is robust under different illumination conditions, can detect a human even if the LIDAR hits any part of the body as can be seen in Figure 2.1. It narrows the region of the interest on the image, so it runs fast. Fusing multiple sensor data not only improve the computational time, but also increase the detection accuracy.

This work makes some contributions to the human detection area in the following ways:

1: A highly accurate and computationally fast multi-modal human detector that fuses 1D range scans and 2D images is presented. This 1D+2D detector does not make any restrictive assumptions about the range scan positions.

2: A simple yet effective geometric descriptor is introduced for LIDAR data. A single-modal human detector, 1D+, using this descriptor is developed. This detector achieves higher accuracy than the state-of-the-art human classifiers based on 1D range scans.

3: It is shown that the multi-modal classifier can be trained with less precise range information, for instance using Kinect sensor depth data, to eliminate the need for expensive and cumbersome manual labeling.

A review of the related work and background about the human detection are summarized in the next section. Section 2 explains the human data set collection, ground truth generation and the registration method of 1D LIDAR data onto Kinect Depth and RGB images. LIDAR and image data feature extraction, fusion modules, the training algorithm and the classifiers are described in Section 3. The tests and the results of both single modal and multimodal classifiers are detailed and compared in Section 4. Finally, the future work and the direction of this work are drawn in the last section.

2.1 Related Work

Computer vision and robotics communities have been conducting excessive research on the human detection for years. In both of the fields, selected sensor types has a direct impact on the fundamentals of the developed method, so it is reasonable first to mention about the used sensors for this purpose. Monocular cameras are the most common sensors employed in the human detection systems. They are cheap and affordable. Stereo cameras are another type of the sensors chosen for the object detection. They can provide 3D structure of the human body as well as the scene. 1D and 2D LIDARs have the capability to provide structure information in outdoor and at dark, so they have been taken place in human detection systems. Infrared cameras, such as Kinect, have become so popular in the last years and their depth data has started to be used for developing human detection algorithms. The human detection algorithms can be categorized basically in two groups. They are called as single modal and multimodal detection methods. The details of these two groups are explained in the following sections.

2.1.1 Single Modal Human Detection

The methods which utilize from one sensor and obtain the features from its data are called as the single modal approaches. There are two essential sensor types used in this group of methods. First group includes the visual sensors, such as monocular cameras. The sensors which provide geometric cues, such as 1D and 2D LIDARs, Kinect type IR cameras form the second category.

2.1.1.1 Visual Single Model Human Detection

Visual human detectors take a camera input image and try to determine the existence and the locations of the humans in it. If there are no constraints and prior knowledge about the possible positions of the humans in the given image, a human is searched in each portion of the image. The search methods can be divided into two main groups. First group of approaches look for the whole human body in all possible sub-windows of the image. This search is applied by sliding a window on the image pixels. Different window sizes are tried in the process, because of the facts that a human might be close to the camera, so his/her body size on the image changes, and naturally a human size might vary from one to another. The features extracted from this sub-window are asked to the classifiers to decide whether there is a human inside of it or not. Haar wavelets are used to construct human descriptors in [136]. A linear support vector machine (SVM) is trained with these descriptors. This method was extended in [121]. It obtains multiple features for different human parts within the sliding window. Multiple classifiers are trained and their responses are aggregated to determine the existence of a human.

A novel gradient and edge based descriptor, called as histogram of oriented edges (HOG), is introduced in [36]. In this method, gradient directions are calculated for the each pixel in the image. The image is divided into cells and a histogram is hold for each cell. Gradient directions of each pixel in the cell are accumulated into the histogram bins. Accumulated value in that bin is simply gradient magnitude. For better invariance to illumination, shadowing; contrast-normalization process is performed in the block level. Each block consists of several cells. A feature descriptor is the vector of all components of the normalized cell responses from all the blocks in the detection window. HOG descriptors are trained by SVM. The performance of this approach is considerably good. Later, its cascaded model is trained in [212] and a high close to real time performance was achieved.

Covariance features are first introduced in [186] for human descriptors. A human is represented as several covariance matrices of features, such as intensity, high order derivates, etc. Covariance matrices do not lie on a vector space, so classical machine learning techniques are not sufficient. A classifier based on Riemannian Manifolds is developed to train covariance matrices in [185].

The second group of human detectors are based on identifying human body parts or common shapes [46] [71] [156] [116]. In these methods, local features of the body parts are obtained and combined to form the human models. Different part classifiers are trained using Adaboost in [119]. A part is represented by the co-occurrences of local orientation features. Joint likelihood of part occurrences is calculated to decide on the existence of a human. Human silhouette information is taken into account in [52] and [133]. In [133], a classifier is trained whose features are retrieved from the silhouette information. The method described in [52] tries to match extracted silhouette features from the test area to the real positive human silhouettes to determine the human locations.

2.1.1.2 Single Model Human Detection using 3D cues

This group of single modal human detectors consists of the methods which obtain their features from 3D or depth data. The algorithms in this category obtain their 3D data directly from the sensors, or they can pre-process the sensor data to generate 3D data. For example, depth information provided by a IR camera and stereo cameras is converted 3D point cloud. They do not utilize from the visual cameras. Some of these types of approaches are outlined in the following paragraphs.

3 types of body parts, head, hand, and foot, are classified by processing only depth images in [140]. In this work, 3D point cloud is generated and segmented to the meshes. K geodesic local interest points are found for each mesh. Then, a patch from each interest geodesic point is extracted. These local patches are rotated/normalized

with respect to their orientation direction. Orientation direction is calculated by the geodesic interest point detection algorithm. It relies on the assumption that geodesic distances on a surface mesh are largely invariant to the mesh deformations and rigid transformations. A dictionary of the local image structures is constructed by using the local patches and a boosted classifier is learned from the dictionary entries. This method claims that it outperforms the sliding window approaches. However, it does not mention about the similar segmented regions, such as a person standing and touching on a wall. These type of scene configurations might make the algorithm fail.

The depth images are converted to 3D point clouds in [70]. A detection window is slided in the point cloud. Each sliding window is reprojected to the depth image. The sliding window is divided into rectangles with different widths and heights. The depth values in the each rectangle are histogramed and their Bhattacharyya distances are is calculated for each possible pair of rectangles. These values form a huge feature vector. Occlusion in a sliding window is detected and those pixels are excluded by weighting in the boosting process. Adaboost is chosen as the learning method to train the feature vectors.

The method described in [84] processes the depth data without converting it to 3D point cloud. Depth images are transformed to spatial depth histograms in X-Z spatial axises. These histograms form method specific images. 4 different oriented, 0,45,90,135 degrees, elliptical filters are convolved on these images. Filter responses are threshold to obtain the back-front and left-right of the body. A bounding box in which a candidate human stands is achieved. In the second phase, human is verified by a face recognition or head-shoulder matching algorithm. This method does not include a training algorithm. It is purely based on the filter responses. As an improvement of this work, some training techniques can be tried after having the bounding boxes.

Some geometric features are obtained from a single layer Lidar in [4]. Firstly, the Lidar data is segmented into blobs. 14 geometric features are extracted for each segmented blob. These features build a descriptor for a leg of the human. Feature vectors are trained by Adaboost algorithm. A single leg model is learned in this way.

The Velocities of the matched parts between two frames are incorporated as a feature in some experiments. The Radius of the circle fitted to the segments is determined as the most informative feature. High accuracy is achieved in their tests. This algorithm is real-time.

Lidar data based features for the pedestrian detection in urban scenarios are exploited in [142]. Multi layer Lidar(Ibeo Alasca) is chosen as a sensor to retrieve 3D data. Each scan of Lidar data is segmented and 15 different geometric features are extracted for each segment. Naive Bayes, Gaussian mixtures models, Neural Networks and SVM are separately tested to classify a segment as a human leg. GMMs and NN perform better than the others. No correlation between multi layer data is considered due to synchronization issues among the layers.

A system to detect and track the humans using only Lidar data is presented in [129]. 2D virtual Lidar scans are obtained by taking a layer from 3D point cloud and projecting it to 2D plane. The virtual Lidar scan data is segmented, and then the bounding boxes are fitted to each segment. The bounding boxes are tracked and matched between the frames via Kalman filtering Each tracked segment is scored to detect the human. Human detection score of the segment is so simple only consists of 4 features. Some of the features are achieved from the velocity of the segments which means tracking and inter-frame process is necessary. The disadvantage of this method is that there are so many thresholds, and no learning method is involved.

A random finite set based method of detection and tracking of the pedestrians using 2D Lidar in dynamic environments is explained in [80]. 2D data of Velodyne Lidar sensor is used. Human detection method is so simple. Mean shift segmentation and blob extraction are performed on the data to find ROI. Simply, the pedestrians are assumed to be in ROI. Then the pedestrians are tracked by probability hypothesis density (PHD) filter. It is assumed that the objects move according to Gaussian dynamics.

2.1.2 MultiModal Human Detection

In this category, it is utilized from multiple sensors to detect the humans. The underlaying idea of using multiple sensors is that combining the advantages of multiple sensors and their data in the same system yields better results. Some of the existing techniques which construct multimodal human detectors are described in following paragraphs.

A fused Lidar and vision based pedestrian detection system is explained in [143]. This work proposes centralized and decentralized fusion methods of Lidar and visual features. Multi layer Lidar is used, but each layer is processed separately because of the problems of synchronization between the layers, pitch oscillations and road inclinations. HOG and COV features from color camera, 14 geometric features from Lidar data are obtained. Centralized approaches combine the Lidar and vision features in the same feature vector. FLDA and SVM perform best in the centralized classifier experiments. Decentralized approach combines a pair of classifiers, one from among Lidar based human classifiers, one from the visual classifiers. The pairs of the classifiers are selected by a minimal-redundancy-maximal-relevance method. The best pair consists of GMMC-Lidar and FLDA-Vision. Decentralized classifier test results are better than the centralized human classifier. Lidar based miss detections (false-negatives) occur in the cases where the pedestrians appear very close to each other or close to other objects, or if they are between other objects. Vision-based missing detections occur in low contrast images or if ROI has intense texture.

The method described in [55] integrates vision and range data for the object detection. 2D scan range data is interpolated to fill missing 3D points for each pixel in the corresponding image. A boosted visual only detector is constructed by sliding a window; the visual features are extracted inside this window. A separate boosted object detector which obtains the features from 3D data is built in the same way. Multimodal object detector is simple weighted combination of boosted 2D image and 3D classifiers. The computational time of this human detector is so high.

A human body part detection algorithm using 1D range data and images is explained in [213]. Human leg is detected processing Lidar data in this approach. 13 geometric features are extracted from Lidar data to train Adaboost for a single leg detector. The method assumes that a person has 2 legs, so it tries to detect both of the legs. The spatial positions of detected legs and their arrangements are modeled and learned from the data, called as part based leg model. The part based leg model is a probabilistic model of Gaussian distribution. The other parts of the human body are detected by obtaining visual features from the omni directional camera. Upper-body, lower-body, and full body in the color image are detected by a Haar-like-feature classifier. Part based model extended by adding 3 new parts detected in the image. The same method(Gaussian distribution) is used to model part based human. Now, a human consists of 5 parts, 2 detected from Lidar and 3 from image data. In this approach, both legs detector and other parts of the human detector must run separately. Features are not combined in a single Adaboost human model trainer.

A cascaded multimodal human detection method, which uses Lidar and color camera data, is described in [172]. In this classifier, two different classifiers are built. First classifier to detect human legs by training geometric features obtained from Lidar data. Second classifier is a visual person detector, off-shelf HOG classifier [36]. Human leg detector determines the region of interest and provides a score which states how likely there is a human in that region. Then, HOG classifier runs in that region of interest by trying different scales. Information achieved from these two classifiers are fused using a Bayesian model.

A recently proposed human classifier [7] focuses on reducing the computational time of the human detector in test time. This approach utilizes from two different data, color camera and 3D point cloud data retrieved from stereo images. There is no geometric features extraction or information fusion mechanism in this method. However, 3D information is exploited to reduce the search time in the images. In addition to this improvement, the time spending to scale the search window is moved to training phase of the classifier.

2.2 Sensor Setup

In supervised learning, the quality and quantity of training data are very critical for the final performance of the classifier. More training data prevents from overfitting, improves generality, and enables trained models to capture possible variations of target class samples. Since our purpose is to construct an inclusive and unconstrained classifier that performs accurately without making any assumption about the range scan position on the human body, a large number of training samples is required for training. However, it is cumbersome to collect such a large number of registered LIDAR and camera data where range scans hit humans on different parts of their bodies. To capture different pose, appearance variations and scan positions, the height and position of the LIDAR must be modified excessively. This is definitely a tedious and inefficient task with no guarantee of capturing sufficient amount and quality of data.

As an advantage, it is possible to generate a high number of diverse range scans for positive and negative samples by using a depth camera that provides the 3D structure of the scene. Any number of scans can be obtained from a depth image by converting the geometric information into LIDAR-like readings synthetically.

Towards this goal, a sensor setup composed of an Asus Xtion Pro Live IR and color camera, and a Hokuyo URG-04LX LIDAR can be constructed. In this way, three sensors, IR camera, color camera and LIDAR can be registered in the same coordinate system. The sensor setup can be seen in figure 2.2. Data registration steps are shown in Figure 2.2.

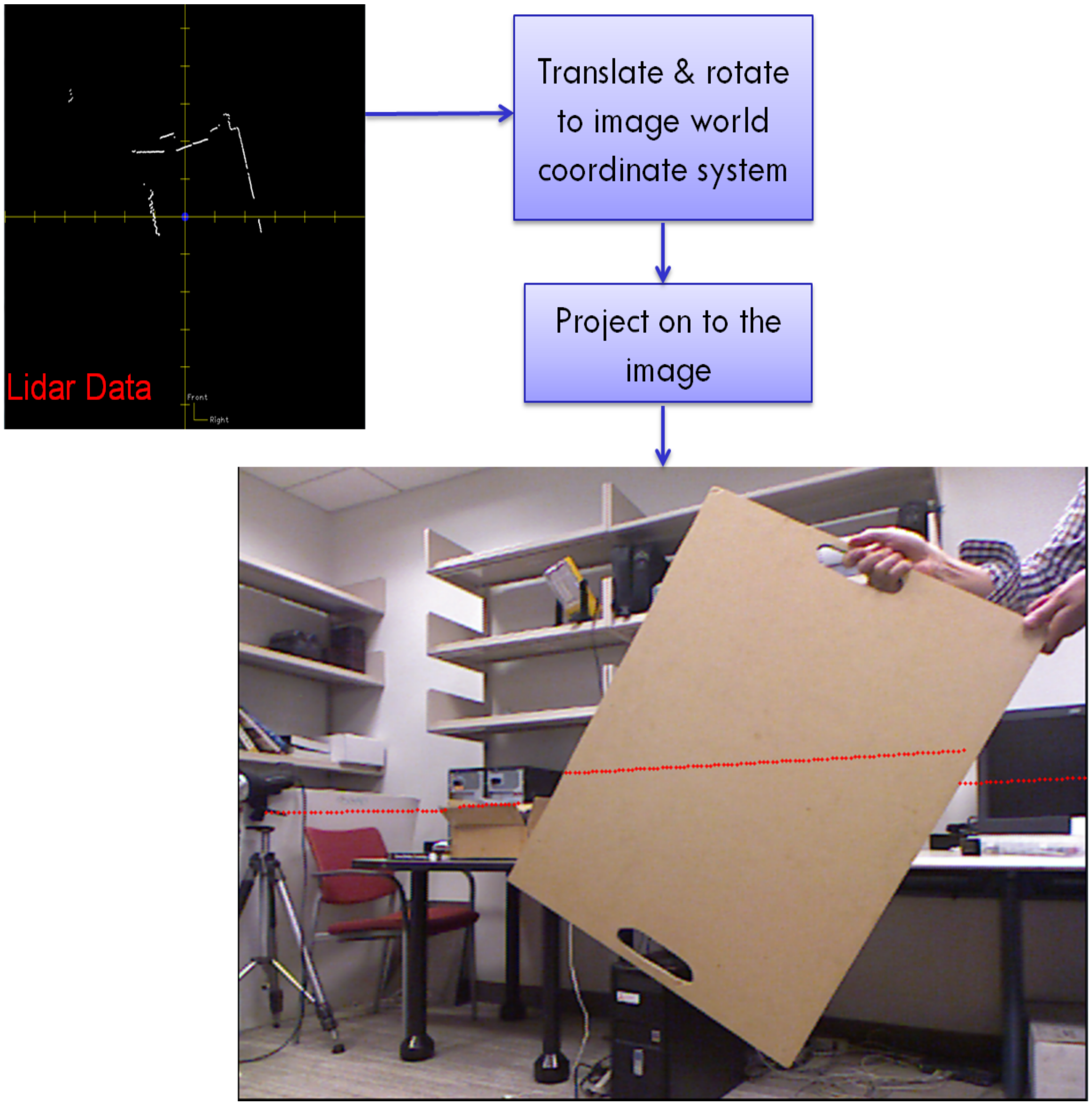


Figure 2.2: The registration of Lidar and color camera data.

A multi-modal human data set, called as *DontHitMe*, is used. It is collected in outdoors (parking lots, etc.) and indoor buildings. Since the IR camera is sensitive to the sunlight, outdoor data can be recorded when there was no direct sunlight in the scene. In addition to the color and depth images, this dataset also includes registered 1D LIDAR range scans. It contains 40,000 images of 450 different humans in different poses, appearance variations, lighting conditions, and shadow artifacts. It includes several human shapes that present a challenge to existing human classifiers, such as women in skirts. To capture the variance of the human poses, it is recorded sequentially at 8 fps. The location and height of the setup is changed during the process to take samples in different backgrounds. Modifying the height of the sensor setup diversifies the 1D range scans.

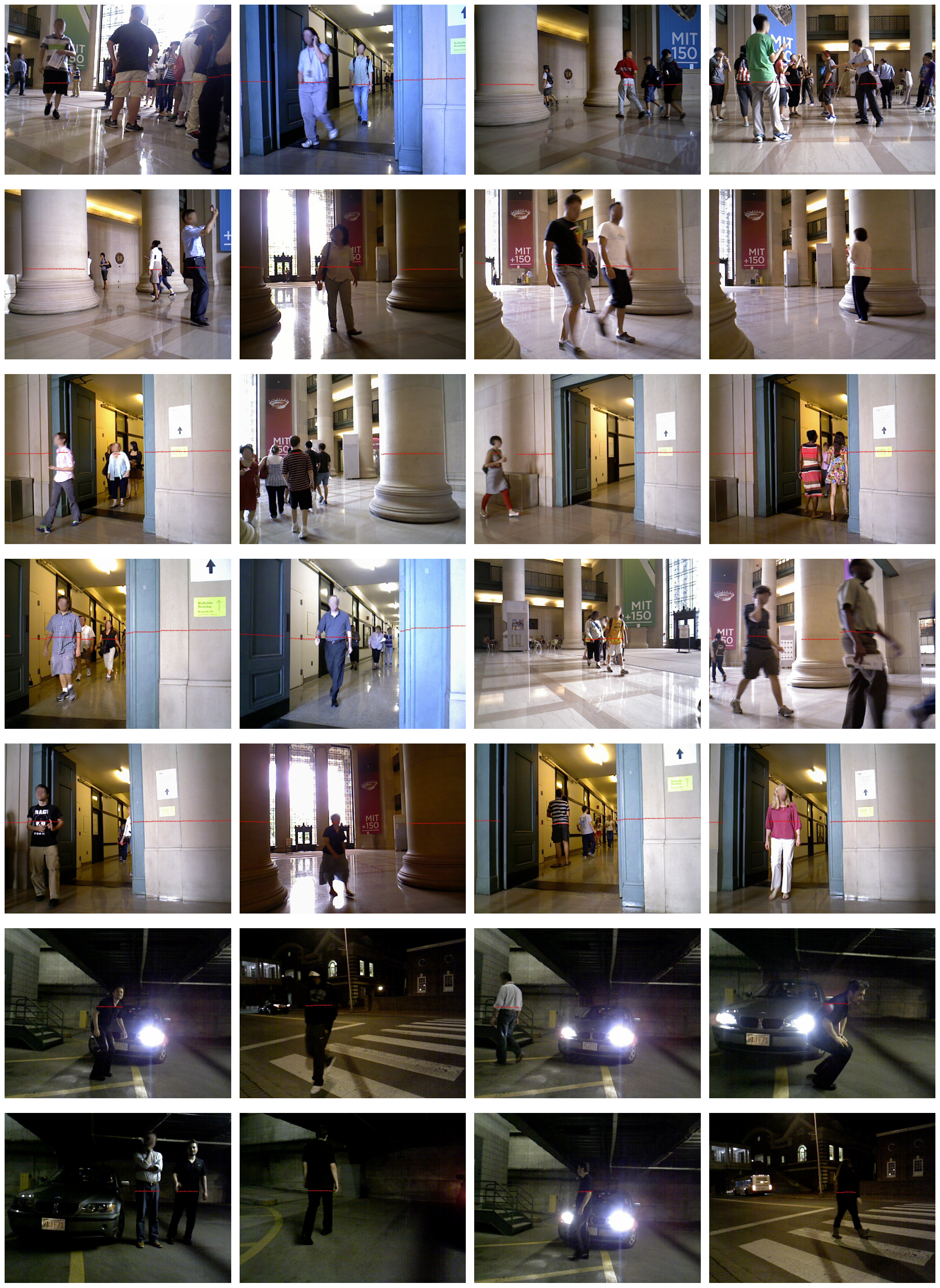


Figure 2.3: Example images from *DontHitMe* dataset. There are some image modifications in this figure. Please check the Appendix A for the details.

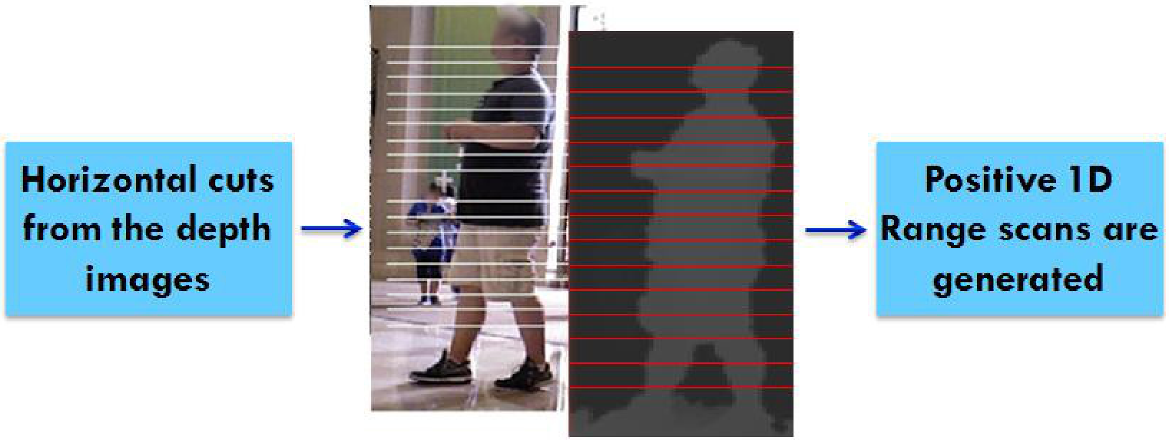

Horizontal cuts from the depth images
Positive 1D Range scans are generated

Figure 2.4: 1D range scans are generated from the depth camera data for each positive window.

The original LIDAR range scans hit human body on different parts from the legs to the head. A total of 3,600 manual ground truth positions in images, depth camera data, and range scans are annotated. Each human in the dataset is labeled with a bounding box, $W(x, y, \delta x, \delta y)$. *DontHitMe* dataset is divided into two different categories. The first dataset, called as *DontHitMe-Indoor*, includes 30K frames and 3,000 ground truths which are recorded indoor buildings. The second dataset is collected at night. It contains more challenging cases for human detectors, such as insufficient lighting and severe illumination changes because of car headlights. This dataset contains 10K frames and 600 ground truths, called as *DontHitMe-Night*. A gallery images of *DontHitMe* can be seen in Fig. 2.3.

To complement the original LIDAR data, the depth camera data in *DontHitMe-Indoor* are processed to obtain additional synthetic range scans as shown in Fig. 2.4. These horizontal scans are produced by uniformly sampling multiple positions vertically along the labeled human window W for the positive samples. In this way, multiple scans are generated from each part of human body, from the legs to the head. A depth scan $L_i = (d_1, ..., d_{m_i})_i$ is discarded if it contained points where the depth camera does not provide a valid distance.

2.3 Human Classifier

To take the advantages of the geometric and visual information, our 1D+2D multi-modal human detector combines the range scan and image descriptors into a single representation. It works in the joint higher-dimensional feature space. A diagram of the classifier is given in Fig. 2.5.

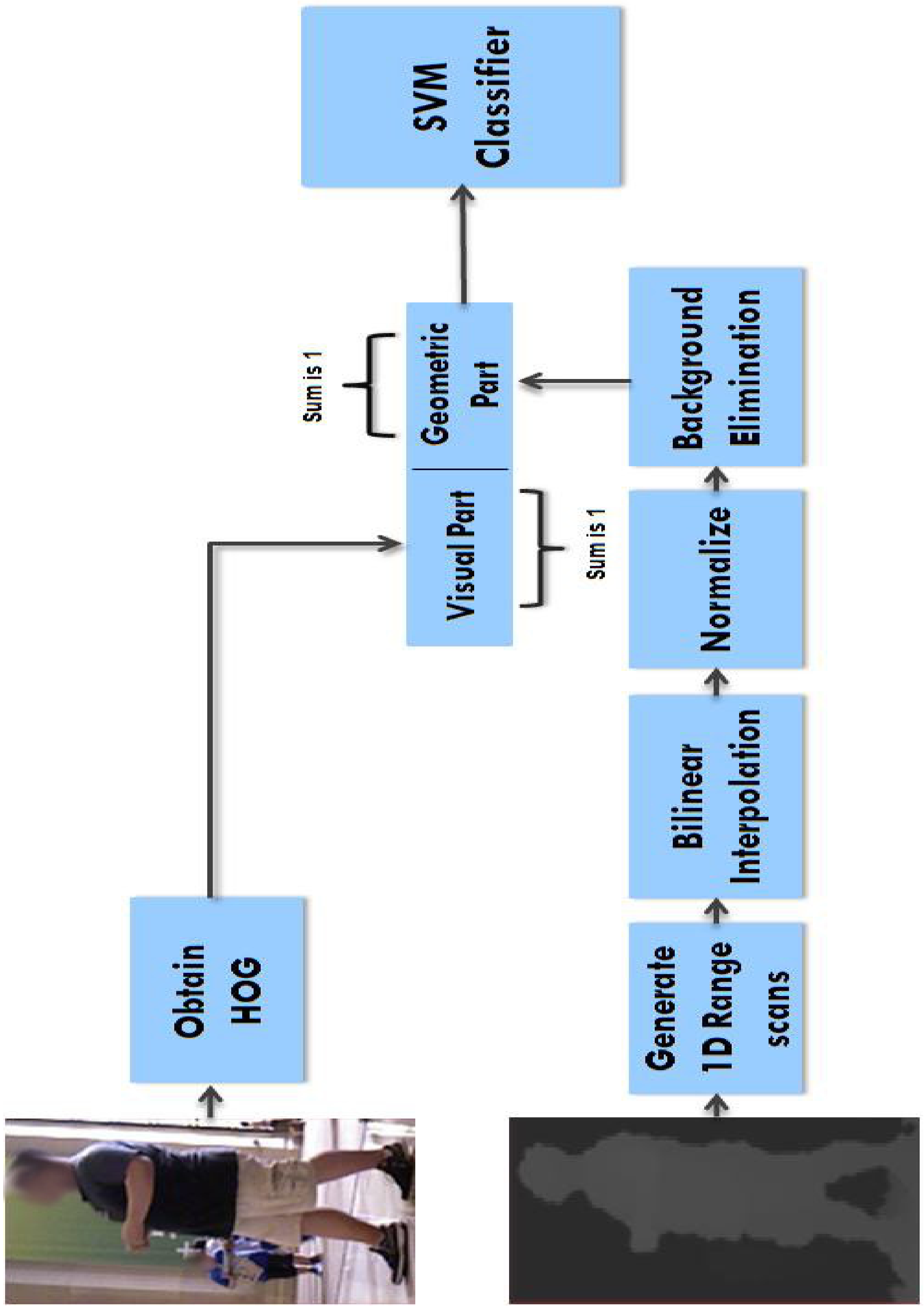


Figure 2.5: Training process of the 1D+2D detector.

For a training image window W_i, the corresponding 1D range scan segment $L_i = (d_1, ..., d_{m_i})$ within the window can be obtained either from the LIDAR or from the depth camera. In the case of the LIDAR sensor, there is a single, horizontal, synchronously acquired range scan segment within the window. On the other hand, the depth camera can provide multiple horizontal range scan segments, which are particularly valuable for training. Here, d is the depth, i.e. the distance of the sensor to a scene point in camera normal direction.

2.3.1 Geometric Descriptor

In contrast to [4] that assumes the geometric descriptor corresponds to leg region, our geometric descriptor f^{1D} applies to every part of the human body. It is obtained by the following procedure:

1) Depending on the size and depth of the human objects, range scans L_i for positive samples form arbitrary length vectors

$$f_i^{1D} = [d_1, ..., d_{m_i}]_i^T, \quad 1 \leq m_i \leq \max(\|w_W\|) \tag{2.1}$$

where $\|w_W\|$ is the width of the window. In order to map the arbitrary length feature vectors onto a uniform, fixed dimensional feature space R^m, an m-point bilinear interpolation, B_m, is performed on f_i^{1D}. After the interpolation, the dimension of f_i^{1D}, that is m_i, becomes m

$$f_i^{1D} = [d_1, ..., d_m]_i^T \leftarrow B_m(f_i^{1D}). \tag{2.2}$$

2) The distance between the sensor setup and a human differs significantly in the scene. To compensate for this distance, the closest point depth, d_C, in f_i^{1D} to the sensor setup is determined. Then, d_C is subtracted from f_i^{1D}.

$$d_C = min(d_1, ..., d_m), \quad d_C \neq 0 \tag{2.3}$$

$$f^{1D} \leftarrow f^{1D} - d_C = [d_1 - d_C, ..., d_m - d_C]^T. \tag{2.4}$$

3) Human can stand in front of all kinds of backgrounds. Background clutter, as well as other objects in the scene, may be positioned at different distances from the human objects. This causes considerable geometric feature variation around the silhouette of the human body. Capturing all this variation in the training data would be one approach. Yet, this requires a huge amount of training data, which would be impractical. Besides, it may cause the classifier to fail because of the weakened discriminative power of the descriptors.

Therefore, the depth values of the feature vector elements that are above a human shape threshold are upper bracketed. The threshold, d_H, is set to the maximum possible radius of a human. If a point in the feature vector f^{1D} has a depth value larger than the threshold, it is set to the maximum radius. As a result, the variation due to the other objects and background clutter are eliminated effectively:

$$d_k = \begin{cases} d_H & \text{if } \ d_k \geq d_H \\ d_k & \text{otherwise} \end{cases} . \tag{2.5}$$

2.3.2 Visual Descriptor

Due to its shape representation ability, computational simplicity, and robustness to illumination changes up to a certain degree, the HOG feature is used to form the visual part of our human descriptor, $f^{2D} = [v_1, ..., v_n]^T$ in the classifier. The HOG features can represent efficiently the local appearance by a distribution of the edge gradients in a cell within an image region. These cells, either overlapping or on a regular grid, are smaller components of an image window. A histogram is obtained within a cell. These local cell histograms are concatenated into a larger window descriptor. All cell histograms of the window descriptor are normalized using the accumulated energy within the window for additional illumination robustness.

2.3.3 Combined Descriptor & Classifier Training

The geometric f^{1D} and visual f^{2D} features are concatenated in the same feature vector to form the final multi-modal human descriptor, f.

The raw geometric and visual feature vectors have different dimensions, thus their individual contributions in the combined multi-modal descriptor are not balanced. To overcome this issue, individual vectors are normalized to unit norm:

$$f^{1D} \leftarrow \frac{f^{1D}}{\sum_{k=1}^{m} d_k} \tag{2.6}$$

and

$$f^{2D} \leftarrow \frac{f^{2D}}{\sum_{k=1}^{n} v_k}. \tag{2.7}$$

The combined descriptor in R^{m+n} is then $f = [f^{1D} f^{2D}]^T$.

In training, the negative samples are chosen from the windows where there are no human objects. Since the window size changes according to the depth value of the window center, the size variation of the negative samples comes naturally. Even though in practice only LIDAR sensor data is available with the image, our training process still benefits from the additional depth camera data.

We use Support Vector Machines (SVMs) as our base classifiers. SVMs fits a hyperplane between the positive and negative training samples in the feature space. The decision boundary is defined by a set of support vectors that separate the positive and negative samples in a maximum margin. The decision function of SVM is

$$h(f) = \sum_{i=1}^{m} \alpha_i [\phi(f).\phi(f_i^*)] \tag{2.8}$$

where α_i are the weight of the corresponding m support vectors f_i^* and ϕ is a mapping function to a space $\mathcal{H}$. The dot products in the decision function can be replaced by a kernel function:

$$k(f, f_i^*) = \phi(f).\phi(f_i^*) \tag{2.9}$$

By using a kernel function the classifier becomes a hyperplane in $\mathcal{H}$, yet it may be non-linear in the original input space. For given a set of labeled samples (x_i, y_i) where the labels $y_i = \{-1, 1\}$, the learning problem of SVM can be formulates as the minimization of

$$\underset{w,\varepsilon,b}{min} \; \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{m} \varepsilon_i \tag{2.10}$$

subject to

$$y_i(w.f_i - b) \geq 1 - \varepsilon_i, \quad \varepsilon_i \geq 0 \tag{2.11}$$

where ε_i a penalty for the misclassified samples. The above optimization tries to classify as many training sample as possible correctly. Also, the minimization of $\|w\|$ makes the margin as large as possible. C is a variable term to set the relative influence.

We use the Radial Basis Function (RBF) as the kernel function of SVM:

$$\phi(f).\phi(f_i^*) = exp(-\gamma\|f - f_i^*\|^2) \tag{2.12}$$

where γ is the width of Gaussian kernel width. By using RBF, it is always possible to find a decision function that perfectly represents a shape in a higher, possibly infinite, dimensional space. By incorporating RBF, SVM decision function takes the final form of

$$h(f) = \sum_{i=1}^{m} \alpha_i exp(-\gamma\|f - f_i^*\|^2) \tag{2.13}$$

the result of final classification is the sign of $h(f)$. This decision function depends on the distance between the support vectors and the data, thus normalizing the geometric f^{1D} and visual f^{2D} feature vectors to unit norm, as formulated in Eqns. 2.6 and 2.7, is necessary. Otherwise, higher dimensional features would be favored by the SVM decision function.

In addition to the above 1D+2D detector, a single-modal classifier, called as 1D+ detector, is also trained by SVM using only the 1D range scans to assess the discriminative power of the proposed geometric descriptor.

2.3.4 Fast Detection

Since the speed of the human detection is an important factor, the 1D+2D detector is employed in a joint fashion that takes advantage of the depth information to eliminate the unnecessary window evaluations.

To determine whether a test window depicts a human, the corresponding 1D and 2D features are computed on the registered data. The range scan line L is aligned with the 2D image I by a perspective transformation $L_I : T(L)$ to obtain a set of image pixel coordinates $L_I = (p_1, ..., p_n)$ in the image.

A search window $W(x, y, \delta x, \delta y)$ centered around p_k is slided on the coordinates of L_I. The size (width δx and height δy) of W is set according to the original depth value d_k of the point p_k such that for smaller depth values (objects closer to the sensor setup) the window size becomes larger. The window size is also proportional to the average human size at the corresponding original depth value.

There is no guarantee that the LIDAR beam always hits a specific level of the human body in a real application, thus the vertical position y of the image window W is not fixed. Instead, multiple windows at different vertical positions $y \pm \Delta y_j$ are tested for each p_k. Similar to the selection of the window size, the number of the vertical windows and their separation are determined by the original depth value of the center point. In this case, if d_k has a large value, a smaller vertical jumps Δy_j between multiple windows is desirable.

Within each window, the geometric descriptor f^{1D} and visual descriptors f^{2D} are computed, normalized, and concatenated into f. If the sign of the $h(f)$ in the SVM classifier is positive, a human is detected by the multi-modal classifier. Algorithm 1 outlines the testing procedure.

In contrast to the conventional visual-only human detectors that need to search entire image at different scales, our 1D+2D classifier reduces drastically the search space. It eliminates completely the image scaling step. Using L_I help to prune most of the image areas, which decreases the computational load greatly.

In practice, window evaluations can be ordered from nearest to far based on the LIDAR sensor depth values to determine the most critical object first.

2.4 Results

Several experiments were conducted to quantify the performance of the proposed multi-modal human classifier, 1D+2D, and its range scan only version, 1D+.

Algorithm 1 Detection Algorithm

```
    Inputs: $L = (d_1, ..., d_n)$ range scan points, $I$, $T$, $h$
 1: * Compute $L_I$, by $L_I : T(L)$
 2: for k=1, ... , n (all points in $L_I$)
 3: * Scale search window $W$ by $1/d_k$
 4: * Compute geometric descriptor $f^{1D} = [d_1, ..., d_m]^T$
      inside $W$ using Eqs. 1-5
 5: * Determine, $\Delta y_j$, vertical jump offsets from $d_k$
 6: for each $\Delta y_j$ for $W$
 7:   * Compute HOG $f^{2D} = [v_1, ..., v_n]^T$
 8:   * Normalize $f^{1D}$ and $f^{2D}$
 9:   * Concatenate $f^{1D}$ and $f^{2D}$ to $f = [f^{1D} f^{2D}]^T$
10:   * Compute $h(f) = \sum_{i=1}^{m} \alpha_i exp(-\gamma \|f - f_i^*\|^2)$
11:   * if $h(f) > 0$ detect human, remove underlying points
        from $L_I$
```

In the first experiment, the performance of 1D+ detector is analyzed. 46,000 positive and 376,000 negative samples from the LIDAR sensor scans and depth images of *DontHitMe-Indoor* dataset are obtained. A total of 43,000 positive samples are generated synthetically from the depth images by uniform sampling and additional 3,000 positive samples are obtained from the recorded 1D range scans.

In order to reduce the variability in the testing scores, multiple rounds of 10-fold cross-validation are performed. It is aimed to see the performance of the 1D+ detector at the different parts of the human body. Therefore, the positive samples in *DontHitMe-Indoor* dataset are divided into 3 categories, as upper body, torso and lower body.

The outcomes of the proposed and the existing state-of-the-art classifiers for the separate human body parts and for negative samples can be seen in Table 2.1. The results are compared to [4], which is a 1D range scan based human classifier. As visible, our 1D+ detector outperforms [4] at least by 20.2% for each part of the human body. The result of this experiment shows that assumptions on the visibility of the legs is not valid for real-life scenarios. The 1D+ is more robust and achieves remarkable accuracy at each level of the human body as can be seen in the detection performance curves of the classifiers in Fig. 2.6. As expected, the method explained in [4] shows its best

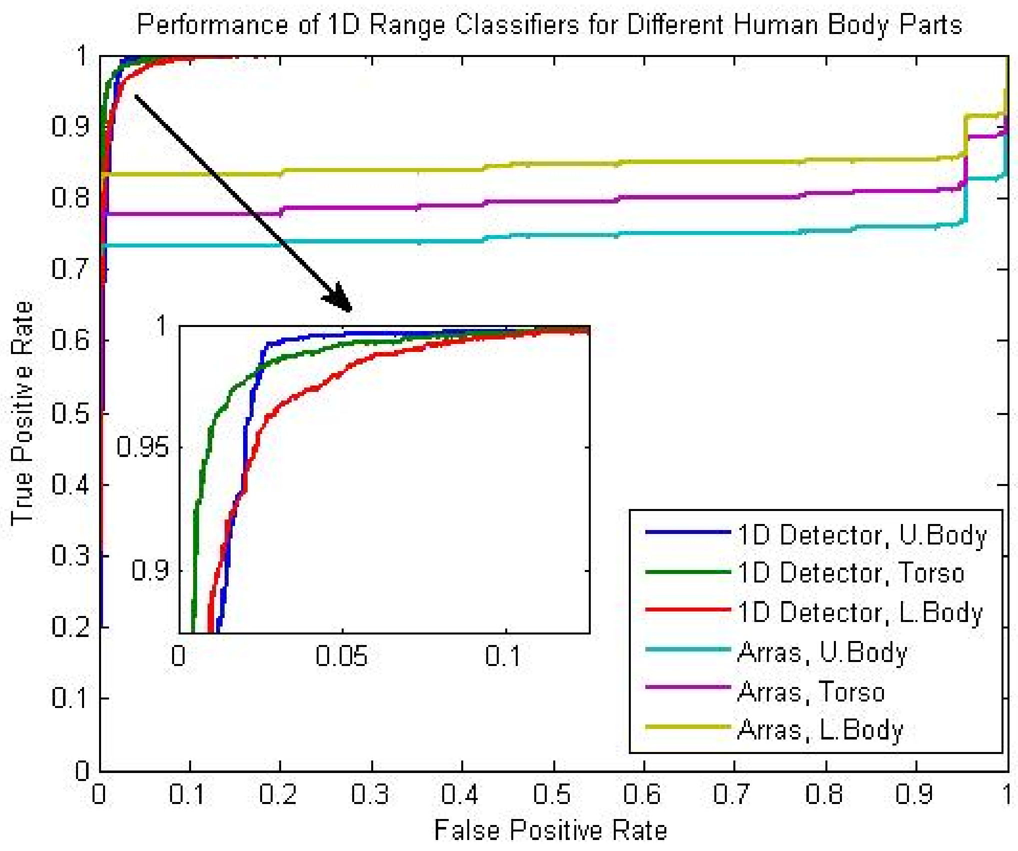


Figure 2.6: ROC curves of the 1D+ Detector and Arras's classifier [4] at different parts of the human body.

performance if the range scans hit the lower part of the human body. Whereas the performance of our detector is almost same at different parts of the body. Proposed 1D+ does not miss any human at 89% false detection level. One of the main reasons of the consistent performance of our classifier at each part is that the positive samples are provided to our detector uniformly from different body parts in the training phase. Also, it learns more diverse geometric cues from every different part of the body from head to the feet.

Another experiment is conducted to measure the performance of the proposed 1D+2D detector. A total of 1,000 positive and 10,000 negative visual descriptors are

Table 2.1: Comparisons of 1D range scan based human detectors for different human body parts

Test Set	1D+ Detector	Arras et al. [4]
Upper Body	97.5%	78.6%
Torso	97.9%	82.7%
Lower Body	96.8%	86.6%
Negative Samples	96.5%	5.6%

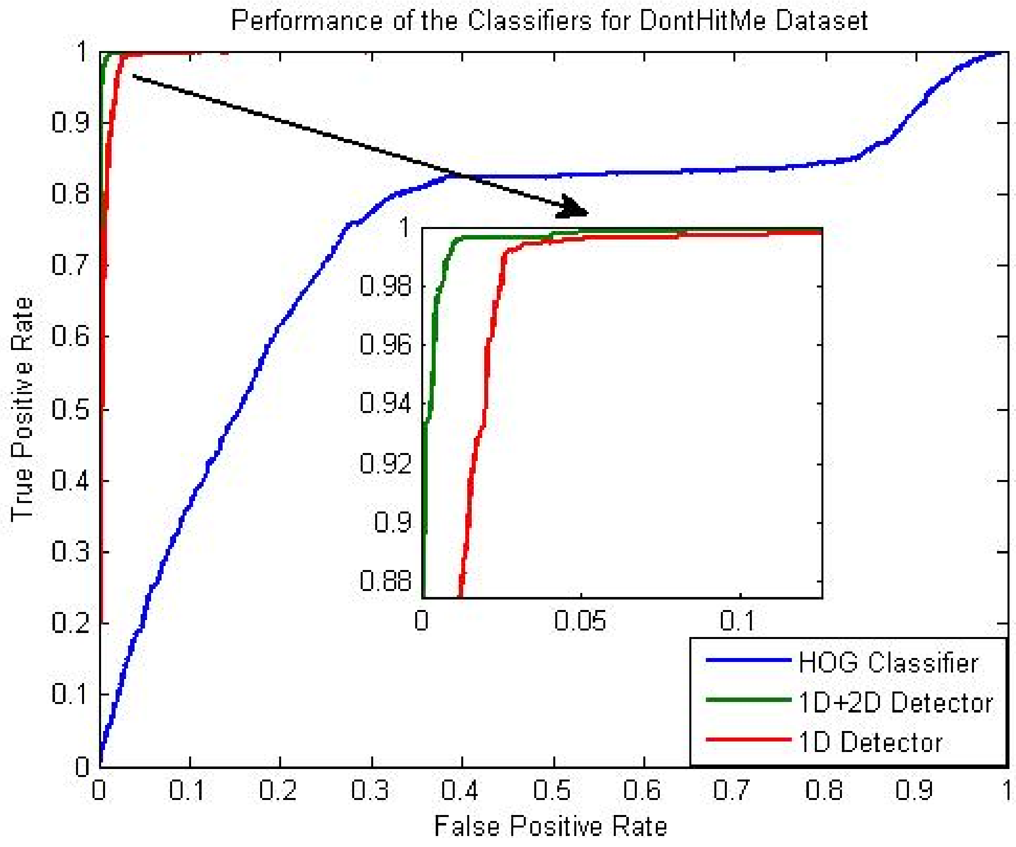


Figure 2.7: Performance of the benchmark HOG [36] and the proposed 1D+2D and 1D+ human classifiers tested on *DontHitMe* dataset.

obtained from *DontHitMe-Indoor* dataset. For each visual descriptor, 20 different geometric descriptors are generated synthetically from different parts of the body by uniformly sampling in their corresponding depth images. In this way, total of 20,000 positive and 200,000 negative multi-modal samples which merge visual and geometric descriptors are generated from *DontHitMe-Indoor* dataset to train the 1D+2D detector. Also, for comparison purposes, 1D+ detector is trained only with the geometric descriptors and the HOG human classifier [36] is trained with the visual descriptors of this set. The accuracy of the proposed 1D+2D detector and 1D+ detector are compared to the HOG human classifier. As in the previous test, multiple 10-fold cross-validations are performed. During this experiment, it is ensured that the test fold and training folds include the samples obtained from different humans. In this way, testing of the geometric and visual descriptors obtained from the same positive samples used in training are prevented. The ROC curves of this experiment can be seen in Fig. 2.7. The 1D+2D detector and 1D+ detector perform significantly better than the visual only detector. Sample detections of the 1D+2D multi-modal human detector from *DontHitMe-Indoor* dataset can be seen in Fig. 2.9.

The proposed classifiers are tested with 600 labeled ground truth images of *DontHitMe-Night* dataset to quantify the performance of the classifiers under severe illumination conditions in outdoor. In this experiment, the classifiers trained in the previous experiment are applied on the night dataset. No new classifier is trained by using *DontHitMe-Night* and no synthetic range scans are generated from the depth images of this dataset. The tested geometric human descriptors are obtained only from the recorded LIDAR scans. The ROC curves of the 1D+2D, 1D+, and [36] detectors are displayed in Fig. 2.8. It can be seen that the HOG descriptor is not enough to represent the human under insufficient lighting and at night times. Our single-modal human descriptor achieved better accuracy than the HOG descriptor. Fusing the visual and geometric cues in a joint feature vector helped to improve the performance; 1D+2D detector outperforms consistently the other alternatives. Sample results of the 1D+2D multi-modal human detector that are missed by the HOG based SVM-RBF [2] but accurately detected by the 1D+2D in *DontHitMe-Night dataset* can be seen in Fig. 2.10

Since the geometric descriptor is obtained from LIDAR scans, 1D+2D detector is more capable of handling image motion blur than the HOG classifier under low-light imaging conditions. Such motion blur examples (at the foot level of the pedestrians) can be seen in Fig. 2.9.

Note that, since it is accurate and computationally feasible at the same time, we compare against the HOG detector that uses SVM-RBF [36] for the most objective evaluations. There are other visual features that can generate higher detection results. Yet, such methods have prohibitively high computational loads for most practical applications.

2.4.1 Computational Load

A 64x128 detection window size was chosen for both the HOG and the proposed 1D+2D detector in the experiments. The dimension, m, of geometric feature f^{1D} is set to 40. The visual feature, f^{2D}, has the dimension of 3780. We used a machine which has 32GB RAM and Intel i7-2760QM quad processor to train and test the classifiers. The classifiers are implemented in native C++ language of Visual Studio 2010 Pro. The training phase of the 1D+2D detector consumed the largest memory among the classifiers in the second experiment since it requires 220,000 descriptors to fit into 29GB RAM, which took $\sim$5 hours.

The computational time and accuracies of the classifiers for *DontHitMe-Night* dataset experiment are compared as can be seen in Table 2.2. The average processing time of a 640$\times$480 scale-space image (10,000 detection windows) by the benchmark HOG classifier is about 0.6 second. At 95% true detection rate, false alarm rate of it is 86%, whereas the false alarm rate of the 1D+2D detector is 0 on the tested dataset. Since the search space of the 1D+2D detector is reduced efficiently by the factors explained above, its average processing time is just 0.05 second. The proposed geometric descriptor has much less dimensions in comparison to other descriptors and it is easy to compute. Thus, 1D+ detector was be able to run at 0.0002 second per scan in the same experiment.

Table 2.2: Average running time and False Alarm Rate (at 95% True Detection Rate) of different classifiers for *DontHitMe-Night* dataset.

Classifier	Time (in sec)	FAR at 0.95 TDR
HOG [36]	0.6	86%
1D+ Detector	0.0002	0.5%
1D+2D Detector	0.05	0%

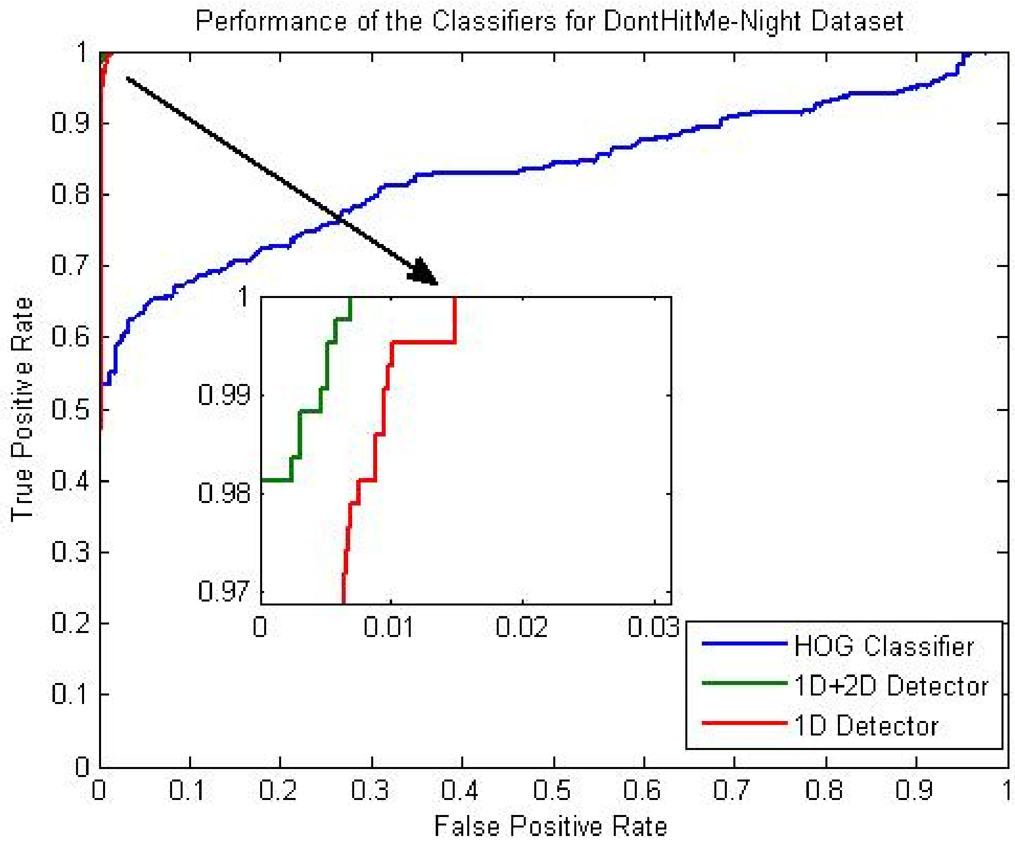


Figure 2.8: ROC curves of the classifiers for *DontHitMe-Night* dataset.

2.5 Conclusion

In this chapter, an accurate and computationally very fast multi-modal human detector is presented. This 1D+2D detector combines 1D range scan and 2D image information within a SVM-RBF framework. Unlike the existing approaches, the proposed 1D+2D detector does not make any restrictive assumptions on the range scan positions, thus this unconstrained detector is applicable to a wide range of real-life detection tasks. Also, a range scan only version, 1D+, of the detector is discussed.

Extensive experiments demonstrate that the 1D+2D detector works robustly under challenging imaging conditions and achieves several orders of magnitude performance improvement (99% true detection at 0.5% false alarm rate in comparison to 54% true detection at 0.5% same false alarm rate on the benchmark) while reducing the computational load drastically (from 0.6 sec to 0.05 sec).

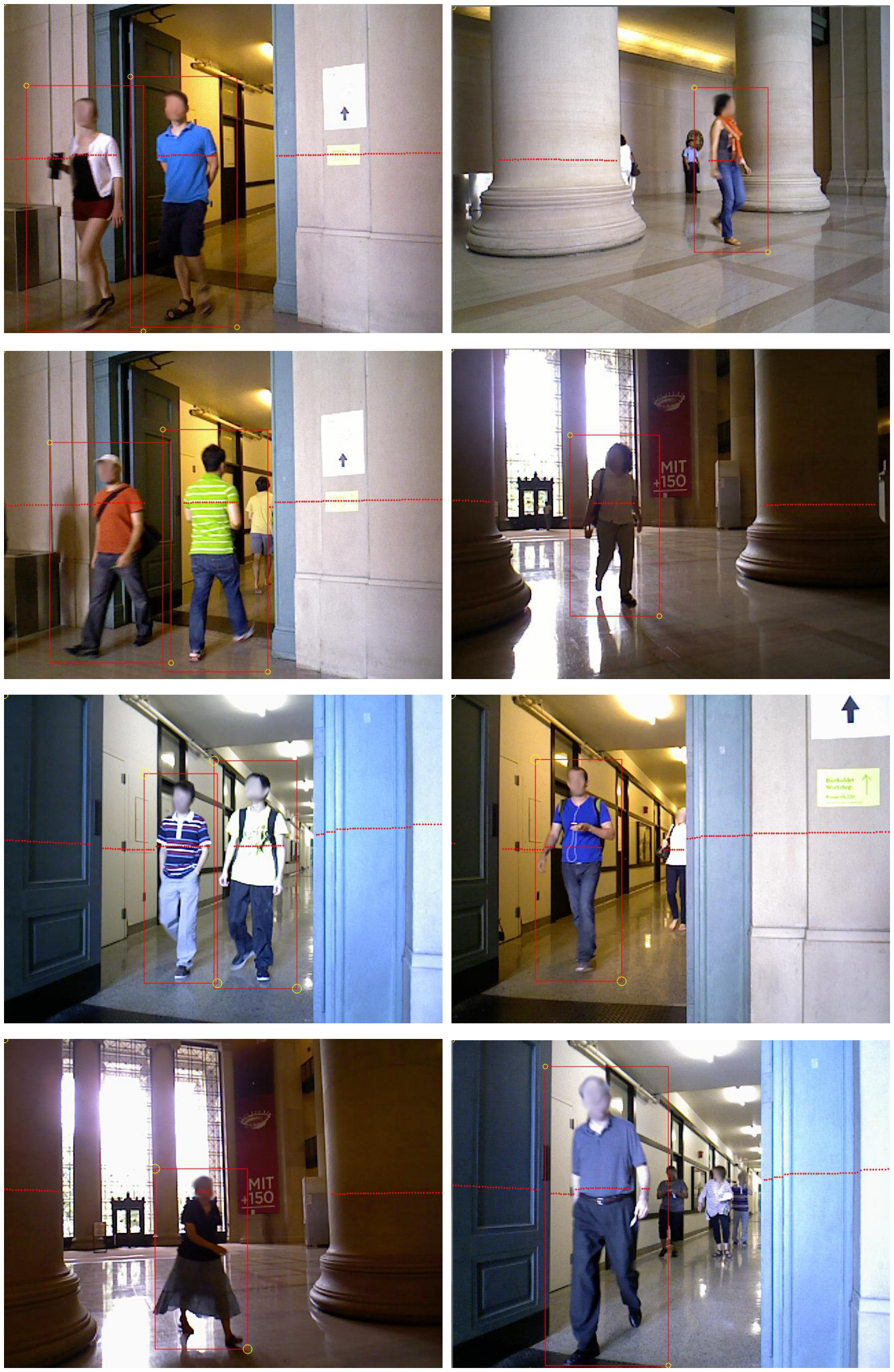


Figure 2.9: Sample detections of the 1D+2D multi-modal human detector from *DontHitMe-Indoor* dataset. Please check Appendix A for the details of the image modifications exist in this figure.

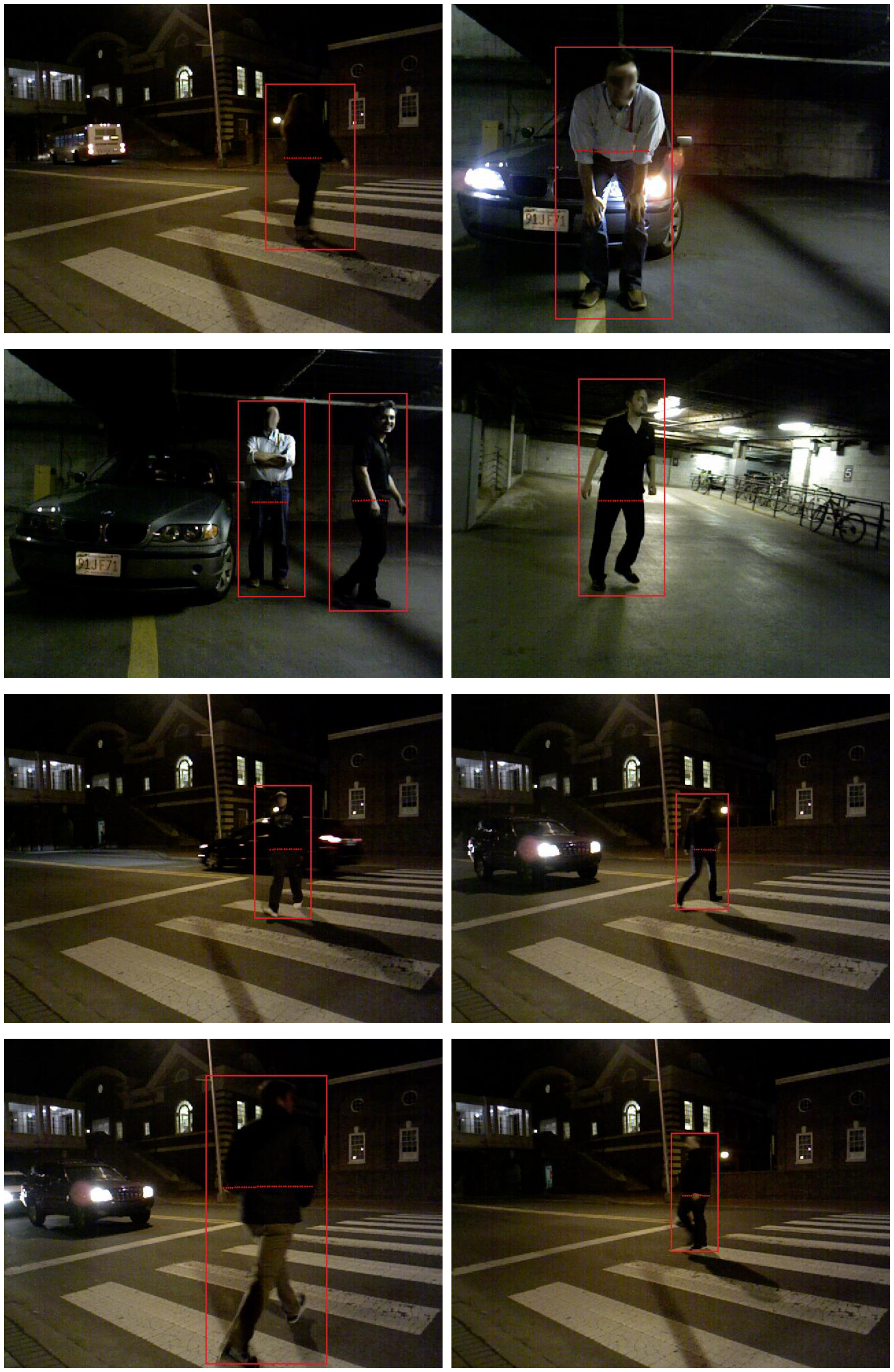


Figure 2.10: Sample results of the 1D+2D multi-modal human detector that were missed by the HOG based SVM-RBF [36] but accurately detected by the 1D+2D in *DontHitMe-Night* dataset.

Chapter 3
BACKGROUND ABOUT GRAPH CUTS

Graph cut techniques have taken considerably attention as an energy minimization method in the last decade, because of the fact that graph cuts produce locally optimal solutions for some low level vision problem, such as stereo, image restoration and segmentation. Its one of the first appearance in the computer vision field became in 1999 by Boykov and et al. described in [21]. Later on, it is extended an explained in more details by [16] and [17].

Graph cuts aims to convert low level computer vision problems, such as stereo [92] [83] [28] [196], image restoration [93] [95] and segmentation to a labeling problem or can be used to solve Markov Random Fields [90]. Every pixel $p \epsilon P$ in the image is assigned to a label l from a set of labels L. The meaning of a label set, L, varies from one problem to another. It might consist of possible disparities for the stereo, intensities for the image restoration, and the parts of different objects for the image segmentation problems. In these problems, the purpose is to find a labeling f which assigns a label $l \epsilon L$ to every pixel $p \epsilon P$ in the image.

A labeling problem can be defined in terms of an energy function. The label assignment of pixels f is obtained by minimizing the energy functions. The energy function that creates a labeling f usually includes two components. It is formulated as the following:

$$E(f) = E_{Data}(f) + \lambda E_{Smooth}(f) \tag{3.1}$$

$E_{Data}(f)$ is called as *Data* term. It describes the distance between the labeling f and the observed data. In other words, it measures the dissimilarity between the labeling f and the observed data. λ, called as regularization parameter, is the relative weight between the *Data* and *Smoothness* terms. *Data* term can be written as

$$E_{Data}(f) = \sum_{p \epsilon P} D_p(f_p), \tag{3.2}$$

where D_p is the distance of a given pixel p to the observed data.

Observed data is a common phenomenon used in many of the classification algorithms. One might think that if the observed data is not incorporated into the graph cut formulation in a special way, what makes the graph cuts different than any other simple classification method? The answer lies in the second term of $E(f)$ which is $E_{Smooth}(f)$. This term makes the graph cuts a unique framework. It formulates the disagreement between two pixels which are assigned to the same label.

Computer vision researchers need the methods which are more gentle and produce accurate results in the borders of the objects for image segmentation, stereo or restoration problems. Therefore, smoothness term is so important especially in the object boundaries. In some energy minimization approaches such as [60] and [177], the smoothness between the pixels are assumed fixed in the entire image, so they produce poor results in the border or edges. The energy functions that do not use fixed smoothness do not show this problem and called as *discontinuity preserving* functions. Some *discontinuity preserving* functions are described in [43], [100] and [180]. However their common problem is that they have many local minimas. Also, solving them are NP-hard problems, so they require very high computational time. For example, one way of solving this problem is by simulated annealing. However, it requires enormous time for a large set of labels. By moving just one pixel p from one label set to another, as in the simulated annealing, cannot be practical and fast enough.

[16] and [17] propose an algorithm which assigns a label to each pixel in the image where the final labeling is piecewise smooth and consistent with the observed data. Unlike the simulated annealing, this approach allows moving a group of pixels from one labeling to another, this move is called as a *large move*. At each iteration of the algorithm, a group of pixels are moved to a labeling until the energy functions is

no more minimized. In each iteration step, the energy is minimized by constructing a graph and applying a max flow min cut algorithm. Min cut value of the edges in the graph represents the formulated energy function of the problem. After a set of iterations, the energy is minimized locally. Since it is based on the max flow min cut algorithm, this algorithm is an approximation method. There is no guarantee that it produces the global optimal solution. However, the local optimal solution is enough for the low level vision problems most of the time. The energy is usually minimized by two-three iterations in the graph cut framework, so the algorithm runs in close to linear time.

Graph cut framework cannot minimize all kind of energy functions. There are some properties which must be conveyed by the functions to be minimized by graph cuts. These properties are defined and explained in detail by [91] and [94]. They deal with the functions which include pair-wise and triple-wise pixel interactions. Then, k-wise pixel interactions are examined by [49]. These publications help the vision researchers to understand what can be solved by the graph cut based algorithms.

Min-Cut/Max-Flow algorithm is the heart of the graph cuts algorithm described in [16] and [17]. Each iteration minimizes the energy by performing a cut in the graph. Therefore, the speed of a cut directly affects the computational performance of the graph cuts. [15] and [20] compare and analyze available min-cut/max-flow algorithms, such as based on "push-relabel" and "augmenting paths" methods. Also, a new min-cut/max-flow algorithm is proposed which performs better than other the other methods in their benchmark sets which consist of image restoration, stereo and segmentation problems. This new approach runs close to near real time.

Recently, a new approach based on graph cuts has been developed by Delong and etc [38] [40]. In addition to *Data* and *Smoothness* terms of graph cuts, it introduces a new term, called as *Label Cost*. It penalizes the solution based on the set of labels that appear in it. It can be considered as penalizing the number of labels in the final labeling of the pixels. Its one of the most useful application area in computer vision is multi-model fitting. In this direction, it is a solution like classical algorithms, K-Means and expectation maximization (EM).

The following sections discuss about the previous work of graph cut and its current state of art applications in computer vision.

3.1 Graph Cut Segmentation and Refinement

One of the main application areas of the graph cut in computer vision is to segment an object in a given image. The graph cut object segmentation algorithm described in [14] and [13] is an image segmentation version of the graph cut method explained in [17]. In this method, the energy function that is minimized by the cut consists of *data* and *smoothness* terms. *Smoothness* term sets the interaction energy between the pixels. *Data* term defines how a pixel fits to a given model. The relative influence between these two terms is set by a constant in the formulation. The energy function is minimized by the method in [17] to obtain the desired object.

Graph cut segmentation algorithm requires two observed data models, called as object and background models. Graph cut doesn't have an implicit mechanism to investigate the models by itself. These models need to be given to graph cut to initialize the method. There are several ways to obtain these models. They can be provided by a user with an interactive tool, or produced by other algorithms as in [86], [41] and [112]. [86] uses an algorithm that scores color contrast differences between the regions, [41] uses motion detection and [112] measures a superpixel based saliency to feed back/foreground models of the graph cut. The quality of the model descriptors directly affects the segmentation accuracy of the cut, so they have to be obtained carefully.

Some object segmentation or tracking techniques produce a rough object segmentation, or low dimensional representation of the object in the image. These representations are necessarily approximate, and thus can miss important border details. Retrieving exact borders of the objects might be necessary for some applications. In addition to object segmentation purpose of graph cut, it can also be used to refine object boundaries from a given rough estimation as described in [86] and [87].

A method which is more sensitive on the distinct border of the objects and incorporate geometric interactions on the borders is proposed in [39]. This method still cut the graph via an approximation solution, but it produces better results if the objects have thin layers. Mixture models do not carry the spatial distribution of the colors in their structure. However, this approach considers the spatial distribution of the colors within any object by adding extra terms into the graph cut equation.

An extension to the graph cut method [21] which employs truncated convex priors is explained in [188]. The improvement in this approach allows more accurate answers for the multi label energy minimizations. Standard graph cut method presents [21] restricted two category label movements. However, [21] provides a new range of moves which act on a large set of labels more than two.

3.1.1 Interactive Graph Cut

Obtaining the initial fore/background models from the users with the help of an interactive tool is very popular and first introduced by [18], [14], and [19]. Then, the following works [50] and [10] also choose to interact with the users before running the graph cut algorithm. In these methods, first the user draws some areas in the background and the object. The models are retrieved from the user drawn areas. Then the graph is constructed and cut. In this way, the user explicitly defines what the background and object will look like. After the object is segmented, the user can rectify the object boundaries with some additional interactions. [37] requires few user interactions and in addition it utilizes from the shape priors. Some other applications of graph cut method which require user interaction are explained in [144], [102], [125].

The graph cut algorithm described in [157], called as *GrabCut*, is another interactive graph cut technique. It extends the regular graph cut technique in several ways. First, the segmentation is performed on a given initial trimap T. The pixels are labeled as T_B for background pixels, T_F for foreground pixels and T_U for the pixels whose labels will be determined by*GrabCut*. Instead of one shot regular graph cut algorithm, iterative version of the graph cut optimization is combined with Gaussian

Mixture Models of background and foreground regions. After each iteration of GrabCut, a border matting algorithm is performed around the object boundary and the colors of foreground pixels.

Two phase interactive segmentation approach is explained for touching neuronal structures from electron micro graphs in [73]. In this method, the structures are first under-segmented. As the result of under-segmentation, the interested objects touch each other. The correction is made with the help of the user inputs.

The usage of discriminative Gaussian mixtures to boost the performance of the graph cut is analyzed in [198]. This method takes its initial seed models from the user interactively. Color distribution, texture and spatial information among the pixels are employed in the graph weights. This work claims that it produces better segmentation results for the texture-rich images.

Active contour models and graph cut minimization method are combined by [203]. This method allows user interaction to correct the first generated result. It can provide more global result by jumping over the local minima and smoothed contours can be obtained via the continuity preserving structure of the graph cut.

3.1.2 Iterative Graph Cut

Graph cut segmentation method requires to be given the object and background models explicitly. All the weights of the graph are set once and then a cut performed to obtain the objects. However, in some cases it is necessary to update the models to achieve converged final object borders. Iterative graph cut is one way in the literature to obtain more reliable and robust segmentation results. *GrabCut* [157] is one of the famous iterative graph cut techniques.

[126] describes an iterative graph cut image segmentation method. It tries to produce optimal solutions with a stepwise way by starting from a global segmentation and moving to the final local segmentation iteratively. Gaussian Mixture Models (*GMMs*) defines the models in their technique. At each iteration, multi scale smooth operation is applied and *GMMs* of the fore/background are updated at each iteration.

The iterative approach explained [138] differs than [126] in some ways. Firstly, [138] constructs the graph not in the pixel level but in the superpixel level. It means that each node in the graph represents a superpixel. Mean-shift algorithm is applied to cluster the pixels, and to obtain the superpixels. At each iteration of the graph cut, not all nodes are added to the graph. Instead, only the neighbor nodes to the object region are joined in the graph structure.

The algorithm in [126] is extended to segment spatio-temporal volumes in the videos [127]. The objects that move fast may be divided into different volumes, so it is not optimal to build a graph by only adding image volumes. Therefore, a new edge types are defined in addition to regular edges in standard graph. These new edges are formulated as new terms in the energy function. Then, the energy is minimized in the same way as explained citeNagahashi07.

Iterative graph cuts, in addition to object segmentation problem, are also developed for the stereo matching [28]. This method produces a dense disparity map for a given an estimated sparse disparity map. It first applies mean shift filtering in the color-disparity space. Then, a cut in constructed graph of the disparities are performed. Filtering and applying graph cut continue until it converges. This method is robust and reliable on discontinues boundaries.

3.1.3 Graph Cut with Priors

Standard Graph cut technique is capable of capturing areas similar to object of interest. As an addition to Graph cut method, [109][110] introduces the idea of using a distance penalty on pixels based upon the distance from the region of interest to bias the segmentation to remain in its region of interest area.

A shape prior which can be incorporated into the graph cut energy functions and for the convex objects is introduced in [189]. It is called as *Star Shape Prior* because of its suitability only for the convex shape objects. A user selected point or points inside of the object must be supplied to the method. The assumption of this approach is that any two points on a line which passes along the user selected point are highly possible to stay in the object. However, this assumption tries to be balanced with the *Smoothness* term of the graph cut algorithm.

Graph cut has some shortcomings to segment thin elongated objects due to shrinking bias. A connectivity prior is described to place into the graph cut technique is described in [193]. Dijkstra's famous shortest path algorithm is formulated as the connectivity constraint. The user inputs are considered as the seed points for new connectivity prior term in the graph cut energy. More accurate results are achieved in the segmentation of thin objects.

Learned shape models are also can be incorporated into the grap cut as the priors. [111] introduces a learned nonlinear shape model to segment the objects more accurate with the graph cut. The method starts with the training of the binary shapes of the object. Principal Component Analysis *PCA* computes the shape models. These shape priors are defined as a term in the energy function of the graph cut. Then, the energy function is minimized by graph cut iteratively. This method does not have any restriction on the shape types. A shape prior, based on the discrete version of the shape distance for the level sets framework, is defined and employed into the graph cut by [197]. In this work, object overlaps are handled by the multiphase graph cut. Multiple shape priors can be embedded into the graph framework.

3.1.4 Cues/Features in Graph Cut

Scale Invariant Feature Transform *SIFT* is a powerful feature to recognize objects in the given images. The method proposed in [175] combines *SIFT* features and graph cut framework to recognize and segment the objects at the same time. An object in the image first recognized by voting the *SIFT* points. Then the recognized *SIFT* locations are seeded to graph cut to segment the entire object. The system works fully automatic.

[173] explains a graph cut based method which combines the color information and 3D position features of the points in its energy function. In this way, multiple cues are incorporated into the algorithm to improve the segmentation accuracy. Unlike

in the image segmentation type of graph cut algorithms, the graph which will be cut is constructed in 3 dimensional space and weights are assigned according to the interactions between the nodes. Using all 3-D points in the graph is computationally expensive, so it firstly forms a surface mesh by applying a Union-Find algorithm to reduce the computation time in the cut process. After the mesh is constructed in 3-D, the normals of the patches are calculated and used in the weighting of the graph. Finally, for the given back/foreground models the graph is cut and the segmentation is produced.

A graph cut based for image and video texture synthesis algorithm is proposed in [96]. In this algorithm, a small patch of the texture which is desired to be augmented and enlarged is provided to the graph cut. A graph which will output syn-texture is constructed in a way that each node represents a pixel in the output texture. The interactions between the nodes are set according to the neighborhood of the input texture pixels. The constructed new graph is solved by the standard graph cut method. Another graph cut based approach which uses texture as a feature to segment the tree-like structures in the medical images is explained in [132].

Superpixels and supervoxels are commonly used in the object segmentation, tracking and recognition applications. An energy function which encourages producing regular superpixels is formed in [190]. This energy is minimized by graph cut framework. First advantage of this method is that accurate boundaries can be achieved. Also, it is computationally efficient and and applicable to produce 3D supervoxels.

3.1.5 Decreasing the Computational Time

Some applications may require close real time computational time. One of these kinds of applications can be object tracking methods. [89] describes a method to reduce the computational time of the graph cut segmentation between two consecutive frames. It keeps track of the pixels in the image if their location changes from one frame to another. The graph is constructed according to the location changes, and the weights of edges are updated only if their weights are changed in the next frame. In this way, the graph cut algorithm is not run for all nodes in the graph, but it is run only for the region of the interest.

Another way to decrease the computational time of the graph cut method is that to run it on a different hardware architecture. Graph cut algorithm is suitable to implement and run it on GPU. Some GPU versions of graph cut is available in the market [194]. Up to 10 times better speed might be achieved with these type of the implementations.

A novel heuristics method to solve the graph cut energy function is explained in [103]. This method is built on top of standard graph cut technique, but applies a heuristic to produce the result faster by using less memory. Its results are nearly same as standard graph cut algorithm.

3.2 Graph Cut Tracking

Graph cut has not taken so much attention in the object tracking field as in the segmentation area. This is mostly because of its nature which produces globally optimally solutions and intend to catch regions similar to the tracked object. Since some background region might show similar features of the object, no one wants those regions to be tracked by the algorithm. Even though its nature that produce globally optimal solutions is seen as an disadvantage of graph cut, I believe in that this feature of graph cut can be turned into an advantage to track objects in certain scenarios. For example, abrupt changes in the location of the objects over the tracking process can be recovered by the graph cut. Or it might be an alternative deformable object shape tracker as the preliminary results are showed in [87]. Or it might form an infrastructure for graph cut based human silhouette trackers. Since graph cut provides point-wise object tracking, the review of current graph cut based object trackers are left in Chapter 6.1. More details about graph cut tracking methods can be found in that Chapter.

3.3 Parameter Adjustment in Graph Cut Formulation

The energy functions that can be solved by graph cut include more than one term. For example, as described in equation 3.1, a typical graph cut energy function consists of two terms, *Data* and *Smoothness*. The weight between them is set by a regularization parameter, λ. In this simplest form of the graph cut energy function, a value to must be assigned to the parameter λ. The value of this parameter plays so important role in the quality of the result produced by the graph cut. If we consider it in the usage of the image segmentation problem, setting the value of λ bigger than it should be causes the graph cut to produce under-segmented results which means that it does not capture the details of the borders of the object and some parts of the object stay in the background region. If the λ is smaller than it should be, in that case graph cut causes over-segmentation and yields to capture the background parts of the image and mostly skips the details on the borders.

Each image conveys different properties, so there is no universal one acceptable value for this regularization parameter λ which can be used to produce excellent results in all graph cut energy formulation and segmentation problems. A value of λ that works very well for an image might not work for another image. Therefore, in most of the graph cut applications, λ is set manually by the user. The user follows a trial and test way to investigate which value of the parameter produces desired results. He/she sets a value and runs graph cut, then checks the results and decides whether the selected value good or bad. Even though the decided value produces the desired segmentation result for one image, the user might need to change it for another image. This way is impractical, also wastes the time of the user.

It would be great if graph cut itself would do parameter adjustment for each image automatically. Unfortunately, graph cut does not have such an internal mechanism to handle this problem. Hence, an auxiliary method to graph cut is necessary to solve this problem. It is open and unsolved problem in the vision community and no much research has been conducted on this problem.

One of the first attempt was by Peng and Veksler in [137] to adjust automatically the regularization parameter of graph cut for each different image. Their method is based on the Adaboost learning algorithm. A set of images are segmented by applying graph cut with different *lambda* parameters changing from 2 to 74 with step size of 8. A group of segmentation results are obtained for each image in the set. These segmentations are evaluated and labeled as "good or "bad" by humans. An segmentation result is associated with a feature vector. This feature vector contains intra and inter object/background properties. For example, intensity contrast inside and outside of the object near the borders are measured, difference in the textures of object and background is calculated, the corners in the boundaries are counted and added to the feature vector which describes the segmentation quality for that λ value of performed graph cut. Adaboost learning algorithm is trained with the feature vectors and segmentation quality labels retrieved from humans. This trained Adaboost is used to evaluate a segmentation result of a graph cut. To select the best regularization parameter for an image in the run time, graph cut is performed for different λ's for the desired image. Then each segmentation quality is asked to Adaboost classifier. The highest value given by Adaboost is considered as the best segmentation and associated λ is taken as the best possible regularization parameter for that image.

Besides the algorithm described in [137] proposes a method to solve graph cut parameter adjustment problem, there are some shortcomings of their method. First of all, running graph cut for each different λ and evaluate its segmentation, then selecting the best segmentation is computationally expensive. It is difficult to use this method in a real time application, because multiple graph cuts on each image takes so much time. It would be great if it tells us the best value of the parameter before running the graph cut multiple times. Also, determining step size of λ is another question that needs to be answered, because if it is kept as a big number, it may produce poor results. If it is kept as small, in this case, the computational time need increases.

The algorithm in [137] sets the regularization parameter of graph cut same for all the pixels in the image. The approach explained in [26] aims to adjust λ for each pixel separately in the image. Instead of selecting a best fixed parameter as in [137], [26] favors *Smoothness* term more on the borders in the image. Firstly, the method obtains the edge maps of the given image by applying Canny edge detector at different threshold parameter. These edge maps are used to calculate an edge probability map of the image. Each value in the edge probability map represents how likely it is a part of the object border. Then this map is incorporated to the graph cut to weight the regularization parameter of each pixel. In this way, graph cut acts more aggressive on the borders to separate the object from the background.

[26] is the first method in the literature which tries to adjust λ dynamically for each pixel in the image. However, it still needs another parameter to set the relative influence of *Data* term to *Smoothness* term. It is not capable of selecting different λ for different images, so it does not solve the problem of adjusting λ dynamically for different images. It just proposes a solution to increase the segmentation quality of the graph cut by favoring it more on the object borders. Also, it does not include a way to distinguish the interior object edges and the actual borders. There might be some specific cases which can affect the performance of graph cut for the objects which have so many interior connected edges.

The approaches in [137] and [26] are proposed only to adjust the regularization parameter of graph cut. In some extended work of graph cuts, such as [86] [87] [135] [110], there are more than two terms need to be combined in the energy equation that is minimized. This is a more challenging problem than just adjusting one parameter, λ, because each parameter in the equation need to be selected in a way that the system produces the best result. A mechanism that finds the best parameter combination, regardless the number of the parameters is a need. To best of my knowledge, currently there is no work that tries to solve it. A partial *unsupervised* method is proposed in [87] to adjust dynamically multiple parameters of graph cut energy function. However, their solution is specific to their problem domain and cannot be generalized for all kind of graph cut parameter adjustment needs. Hence, there is a still gap of solutions in this direction of graph cuts. A supervised or unsupervised method which is incorporated into graph cut would increase the quality of the results produced by it.

Chapter 4

POINT-WISE GRAPH CUT REFINEMENT AND SEGMENTATION

This chapter compares and analyzes some modified and newly developed graph cut based techniques for the purpose of the complex object refinement and segmentation. As it was introduced in the first Chapter, the *Refinement* process can be considered as a transition way in order to obtain the point-wise mask of an object from its given low dimensional representation. It is an important task if one application needs to know detailed border of an object in the availability of its rough representation. First section focuses on modified graph cut methods to refine the low dimensional representation of the trails for an unmanned ground vehicle. Second section introduces a novel graph cut based method to segment and refine the complex objects under different illumination conditions.

4.1 Modified Graph Cut for the Trail Refinement

A trail is a path or road used for hiking, walking or biking. Trails are navigationally useful regions for both unmanned aerial and ground vehicles in outdoor environment. Above-ground pipelines, rivers, canals and hiking tracks show the way to unmanned vehicles. Structurally finding and tracking a trail in the outdoor environment help an autonomous vehicle to drive or fly safely. Even though trail finding and tracking is an easy task for a human, it is a challenging problem for the robots.

Vision-based trail finding and tracking can be considered as a form of road following [141, 179, 170, 65, 187]. However, several factors make the computer vision task particularly hard, including indistinct borders, abrupt elevation changes, dead-ends and forks, sharply varying illumination conditions due to shadows, a wide range of trail materials and hence colors and textures, and the possibility of in-trail objects such as rocks, stumps, or grass.

In [149, 150], a trail tracking algorithm was presented which takes into account primarily the color contrast difference between the trail and neighboring image regions. While the method works well in many instances, one shortcoming is that it maintains a low-dimensional representation of the trail shape. In [149] the trail was represented strictly in the image domain as a triangle (to account for perspective), and in [150] the trail was represented as a circular arc in vehicle coordinates, projected to an omni-directional image. These representations are necessarily approximate, and thus can miss important border details and possible in-trail obstacles (see Fig. 4.1 for a sample of the difference between a coarse and detailed trail segmentation).

In this section, a second stage of shape estimation is described in which the initial, rough shape is *refined* automatically, without user interaction [86]. Several basic segmentation algorithms for this purpose, specifically graph cut [13], graph cut with distance maps [109], GrabCut [157], and a grouping method based on superpixel over-segmentation [45] are compared and analyzed. Although the focus in this section is on trail image data most relevant to mobile robot applications, the problem of automatically refining segmentations is a general problem. The refinement of other set of the objects will be discussed in the next section.

The changes made on the basic graph cut algorithms to use them as automatic foreground extraction methods are described in the following sections.

4.1.1 Obtaining Background and Foreground Models for the Graph Cut

Standard graph cut, graph cut with distance penalty and GrabCut algorithms require color information of background and foreground regions in the image. The weights of the edges in the segmentation graph G of these algorithms are set by provided background and foreground models, respectively M_B and M_F.

These two models may include hard constraints in addition to color information, such as some particular pixels in the image can be set as true background or object

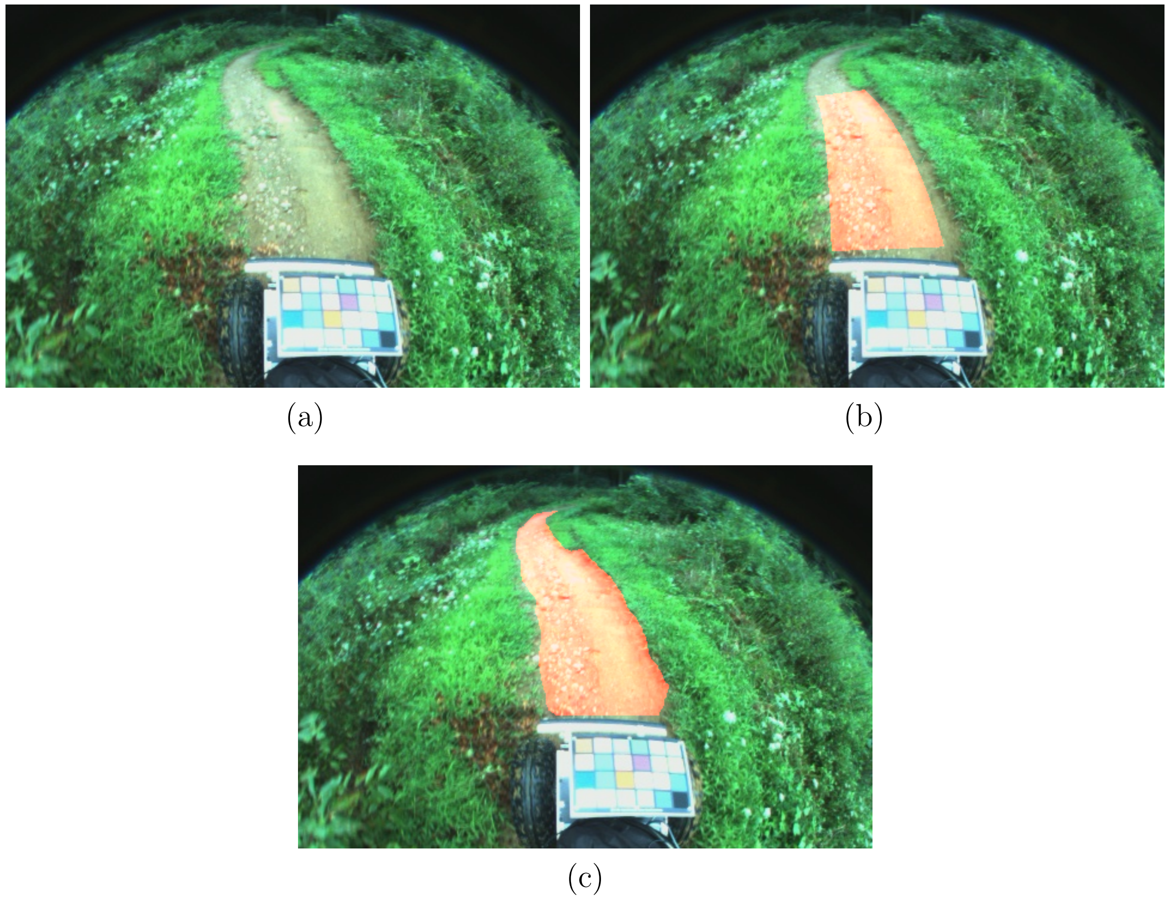


Figure 4.1: (a) Sample trail image; (b) Coarse ground truth overlaid; (c) Detailed ground truth

regions as in [14]. M_{back} and M_{obj} may be obtained from the users in interactive graph cut segmentation methods, or feeded to the algorithm as pre-prepared models by the programmers. The models which contain the real true properties of the objects provide better segmentation results.

In this work, M_B and M_F are obtained from the method explained in [149]. Briefly, it fits an estimated shape S (A *triangle*) to the trail in the image. S may cover some part of the background or not include all the trail region. S is scaled downed and S^- is formed. S^- provides color information to construct M_F. Some background pixels in S may generally be placed near the border of S. Scaling down process eliminates those pixels and yields better true trail pixels to construct M_F. Simply, all pixels in S^- is used to form M_F. To retrieve M_B, first S is scaled up and obtained S^+. All color information of the pixels outside of S^+ are employed in M_B. Scaling up and down factors are 60% and 40%, respectively. In Figure 4.4(b), the pixels colored as green form M_B, colored as red form M_F, the purple triangle corresponds to the initial estimated trail S.

The pixels in S^- are set as hard constraints in the segmentation graphs of the methods. No background pixels are provided as hard constraints to the algorithms. In the trimap of GrabCut method, unknown label is assigned to all pixels outside of S^-, and inside of S^- is considered as foreground.

4.1.2 Distance Map Construction

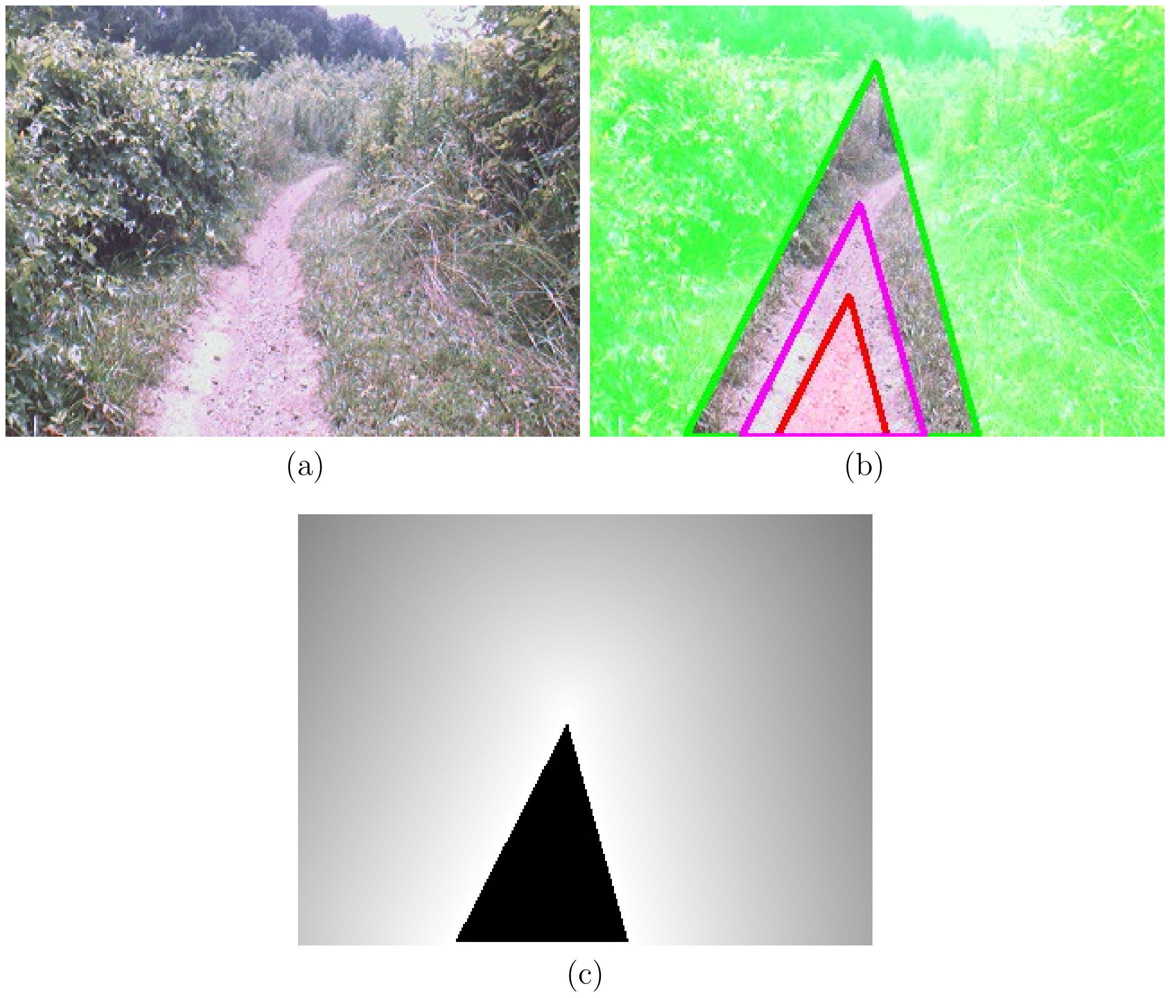
[109] uses a distance penalty function in its *regional term*. To set the penalty function of this method, a distance map, Map_{dist} is constructed, and provided to the algorithm. Initial estimated priori S of foreground includes three important cues about the object: Color information, estimated shape of the object, and spatial information of it in the image. To obtain color models M_F and M_B, color information provided by S is used. Other two cues coming with S help to construct Map_{dist}. Figure 4.4(c) shows the distance map of the given image. Instead of using object center to calculate the distances as in [109], the closest pixel to S is used. The distance penalty map is defined as in the following equation:

$$Map_{dist}(i) = \|i - closestTo_i\| \tag{4.1}$$

where $closestTo_i \in S$ is closest pixel of S to pixel i in the image space. $\|i - closestTo_i\|$ is the distance between $closestTo_i$ and pixel i.

4.1.3 Removing Weak Components

The raw foreground mask generated by graph cut techniques typically contains some noisy, small and weakly-connected foreground regions, because the images can


(a) (b)

(c)

Figure 4.2: The image, its background and object models, and its distance map.

contain a non-homogeneous color distribution inside the foreground regions and [14] uses the color histogram to assign weights to terminal links. In order to clean up those regions, first do morphological opening and closing are performed and found the connected components. The largest region is taken as the final refined foreground region.

4.1.4 Results

The algorithm in [45] is an efficient method to over-segment an image into self-similar regions. It defines a predicate for measuring the evidence for a boundary between two regions using a graph-based representation of the image. Their segmentation algorithm is constructed based on this predicate. Although this algorithm makes greedy decisions, it produces segmentations that satisfy global properties. [152] describes an iterative method for grouping superpixels to maximize both shape and appearance contrast criteria which starts from initial triangular model $\hat{T}$. Superpixels outside this initial region may be added and superpixels inside may be removed in a non-parametric process that allows a wide effective range of shape deformations, including the introduction of holes or outlier subregions within the trail which may be obstacles. The refinement results of this superpixel based algorithm is also compared to graph cut based algorithms.

The accuracy and efficiency of the algorithms are measured on a diverse set of trail images. The experiments are run on two set of images. Set-1 consists of the images collected from several trail image sequences taken by the robot platform as it was manually driven and Set-2 includes 30 images from the hiking trail, river and canyon sequences taken from the web. Images of Set-2 are available through the "Data/trail30" link at `http://nameless.cis.udel.edu`. The data sets are scaled to 320 by 240 as necessary. At regularly spaced intervals along trail sequences, ground-truth segmentations are manually generated . The results are compared to the ground-truth segmentations to quantify the accuracy of the refinement algorithms. The following polygon area overlap formula is used to measure the overlap between the ground-truth segmentation and the result of the refinement methods suggested by [159] :

$$Overlap(\mathcal{R}_1, \mathcal{R}_2) = A(\mathcal{R}_1 \cap \mathcal{R}_2)^2/(A(\mathcal{R}_1)A(\mathcal{R}_2)) \tag{4.2}$$

where $\mathcal{R}_1$ and $\mathcal{R}_2$ are given two regions to calculate the overlap between them.

Algorithm	Score of Set-1	Score of Set-2
Initial Models [149]	0.663	0.732
Superpixel Grouping [45]	0.740	0.713
GrabCut [157]	0.631	0.738
Graph Cut [13]	0.730	0.737
Graph Cut(With Distance Map) [109]	0.783	0.760

Table 4.1: Median overlap scores of the methods for Set-1 and Set-2. Initial models are the low dimensional representations of the trail regions which are the triangles by [149].

Two set of images by using each described method in the previous sections are refined. Then, the median overlapping scores and average segmentation time of refinement methods are calculated. Initial models are obtained from [149] for two set of images. The overlap scores can be seen in Table 4.1. Average segmentation time of GrabCut, superpixel, graph cut, graph cut with distance penalty methods are 2.85, 4.95, 0.17 and 0.19 seconds, respectively. Since the initial models may contain non-trail regions or not cover entire trail region, the median overlapping score of given initial models is not good in the first image set. Using standard graph cut method for refinement process improves the initial segmentation quality. However, incorporating distance map approach to graph cut performs better than standard graph cut in both two image sets. The results of the refinement methods are shown in Figure 4.3.

4.2 Novel Graph Cut based Segmentation Method

In this section, a novel graph cut-based algorithm is explained to refine automatically, without user interaction an object's estimated borders if an initial coarse estimate is given. The standard graph cut method [13] uses only intensity information in its formulation. Intensity alone is often not enough to segment or track objects with diverse appearance and shape characteristics in a large range of images. Therefore, several changes in the standard graph cut method are made to increase the accuracy of the refinement task. First, an adaptive color space selection mechanism is employed

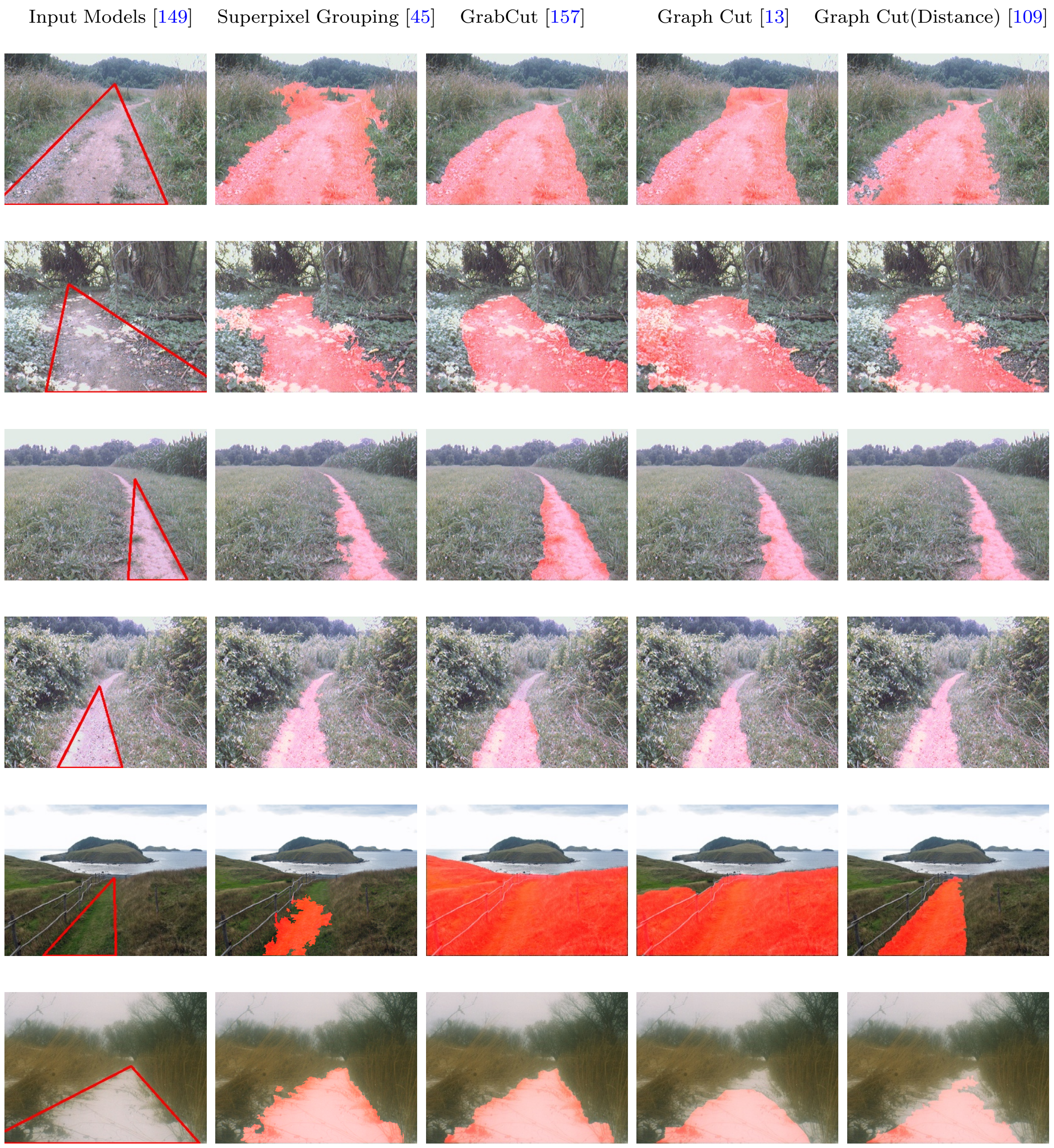


Figure 4.3: Trail refinement results of the methods. Column headings show the corresponding method names. Top 4 rows demonstrate images from Set-1 and, the last two rows contain images from Set-2.

in the algorithm. Second, color information in the image is clustered by k-means. The knowledge gathered from the clusters are included in the formulation of the region and boundary terms of the graph cut. Third, the spatial distance information is incorporated into the algorithm to constrain the final segmentation to stay around the region of interest. The influence of the distance penalty information is adaptively set by the algorithm.

For comparison, support vector machines (SVMs) [25] is applied to the same problem to learn a model of the object using both appearance and spatial features, and then to classify an image into object and non-object pixels based on the learned model as described in [87].

Both techniques require to take a background and foreground region information in the image, M_B and M_F, respectively. M_F is obtained by scaling down initial estimated shape of the object, M_t. M_B is obtained by scaling up M_t and taking all the pixels outside of that region. M_t can come from manually segmented ground truths of the object, previous frames in the tracking procedures, or can be given by any other algorithm. Figure 4.4(b) shows M_B and M_F .

4.2.1 Automatic Color Space Selection

Standard graph cut algorithm requires some feature information from the background and foreground regions in the image. Choosing these models as much as informative and distinctive helps graph cut to produce better segmentation results. Test image sets contain high illumination changes. Working and sticking to only one color space such as RGB does not always produce good results. In some cases it is investigated that using CIE-LAB color space gives better results. Therefore, a mechanism to switch among the color spaces during the run time of the algorithm to improve the accuracy is developed. Four possible color feature spaces are considered for this purpose: RGB, LAB which uses three channels of CIE-LAB color space, AB which uses only the chromaticity information, and L which uses only brightness of CIE-LAB.

To achieve self-adaptiveness among the feature spaces, the method collects some feature statistics from the inside and outside of the object regions in run time. To do

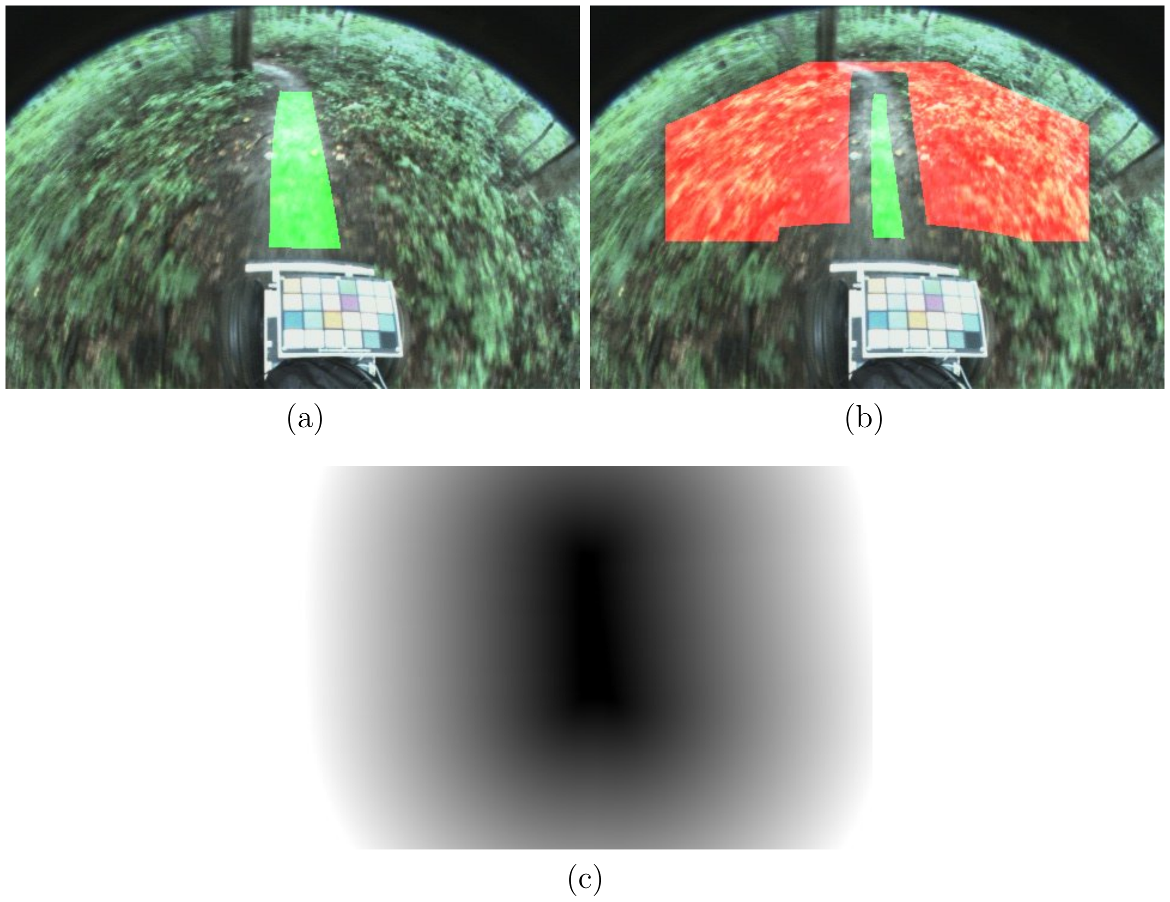


Figure 4.4: (a) shows the image overlaid with its ground-truth polygon; (b) its obtained background model, red colored pixels, and foreground model, green colored pixels, for the graph cut and SVM algorithms; (c) and its distance penalty map for the graph cut algorithm.

that, all the pixels in the image are clustered into k different labels by applying k-means algorithm. The number of the clusters is chosen as 12, $k = 12$, for k-means. K-means is separately performed four times in RGB, LAB, AB and L feature spaces and four different cluster labels are obtained from the image. An object region color distribution, M_F, is modeled by a histogram $h = (f_1, \ldots, f_k)$ of the label frequencies inside it. The background region model, M_B, is formed in the same way. This allows to capture multi-model color distribution from the inside and outside of the object. The appearance dissimilarity, d, between M_B and M_F is measured by histogram distance function which is chi-squared metric $\chi^2(h_i, h_j)$. This measurement is done for each feature space and separate dissimilarity values are retrieved, d_{RGB} is for RGB, d_{LAB} is for LAB, d_{AB} is for AB and d_L is for L feature spaces. The feature space which provides the highest dissimilarity taken as the working color space of the graph cut. In this way, more informative and distinctive foreground and background models are provided to the graph cut. *Regional term* and *Boundary term* of the graph cut are changed as follows:

$$R_i(''Object'') = -ln(\frac{Pr(l_i|M_F)}{Pr(l_i|M_F) + Pr(l_i|M_B)}) \tag{4.3}$$

$$R_i(''Background'') = -ln(\frac{Pr(l_i|M_B)}{Pr(l_i|M_F) + Pr(l_i|M_B)}) \tag{4.4}$$

$$B_{(i,j)} = |R_i(''Object'') - R_j(''Object'')| \tag{4.5}$$

where i and j are two neighbor pixels, l_i is k-means label of pixel i, M_F and M_B are the histogram label frequencies of the object and the background regions in the image, respectively.

4.2.2 Distance Penalty Map Construction and Weighting

Graph-cut method produces global segmentation and tends to catch some unintended regions which are similar to the desired object. However, in the tracking procedure of the objects, the pixels labeled as the object in the current frame are most likely will be labeled as the object again in the next frame. To incorporate this biasing information, the pixels which are far away from the object region in the last frame need to be penalized. This penalty information is added to the standard graph-cut by constructing a distance penalty map as in [109]. However, the distance penalty is weighted in a different way. Instead of guessing the location of the object in next frame as described in [109], the dissimilarity, d, between the object and background regions is chosen as a criteria to weight the distance penalty information adaptively. In this weighting technique, if the dissimilarity, d, is high between the regions, graph-cut is forced more to stay around the object location which is segmented in the last frame. Specifically, if M_F and M_B are similar to each other, graph cut does not go to go far away from the last position of the object and look for more pixels to add to the object region, The distance map of the image is constructed as: $Map_{dist}(i) = \|i - closestTo_i\|$, where $closestTo_i$ is the closest pixel of the object to pixel i in the image space. $\|i - closestTo_i\|$ is the distance between $closestTo_i$ and pixel i. Figure 4.4(c) shows the distance penalty map of a given image. The distance penalty function is added to *regional term* of the standard graph cut algorithm in the following way:

$$R_i("Object") = Pr(P_i|O) + \beta(\alpha(d_{cs})Map_{dist}(i)) \tag{4.6}$$

where $Pr(P_i|O)$ is the penalty of adding pixel i to the object region. β is a constant term to set the relative influence. $\alpha(.)$ is the negative log-likelihood function to weight the distance penalty adaptively. d_{cs} is the smallest dissimilarity value returned by chi-squared metric among all feature color spaces considered in the algorithm.

4.2.3 Results

The proposed algorithm was tested with 3 different data-sets. The *Trail* dataset from [150] consists of 17,358 frames of video taken along a hiking/mountain biking trail in a mixture of field and forested terrain. The *Head* dataset is a short 383-frame clip from a standard video compression benchmark in which a person's head bobs around in the back of a car in [62]. Finally, *Hand* is a 5-minute (5,616-frame) video recorded outside our lab of a hand waving and gesturing in front of a complex background. Ground-truth object segmentations are manually generated for about $5 - 10\%$ of each datasets at regular intervals. The same area overlap formula is used suggested by [159] to measure the accuracies between the ground-truth segmentations and the results.

The experiment is performed for the images which have ground-truths. In order to see the performance of adding distance penalty to the graph cut, the distance

Method Name	*Trail*	*Hand*	*Head*
Standard GC [13]	0.55	0.79	0.52
Color GC	0.69	0.95	0.84
ColorD GC	0.79	0.97	0.89
SVMs	0.76	0.95	0.86

Table 4.2: Median overlap scores of the single frame image segmentation results for the data sets.

information is removed from the graph cut method, called *Color GC*, and applied on the data sets. All experimented methods are initialized with the ground-truth segmentations, so M_t is set to be the ground-truth. Table 4.2 summarizes the median overlap scores between the methods and the ground-truth segmentations for each data sets. Some results of this experiment are shown in Figure 4.5. As expected, adding the color information to the standard graph cut helps to increase the accuracy of the segmentation. Also, incorporating the distance penalty and weighting it according to the dissimilarity between M_F and M_B improves the performance. The last row of Table 4.2 shows the results of SVM based method, described in [87].

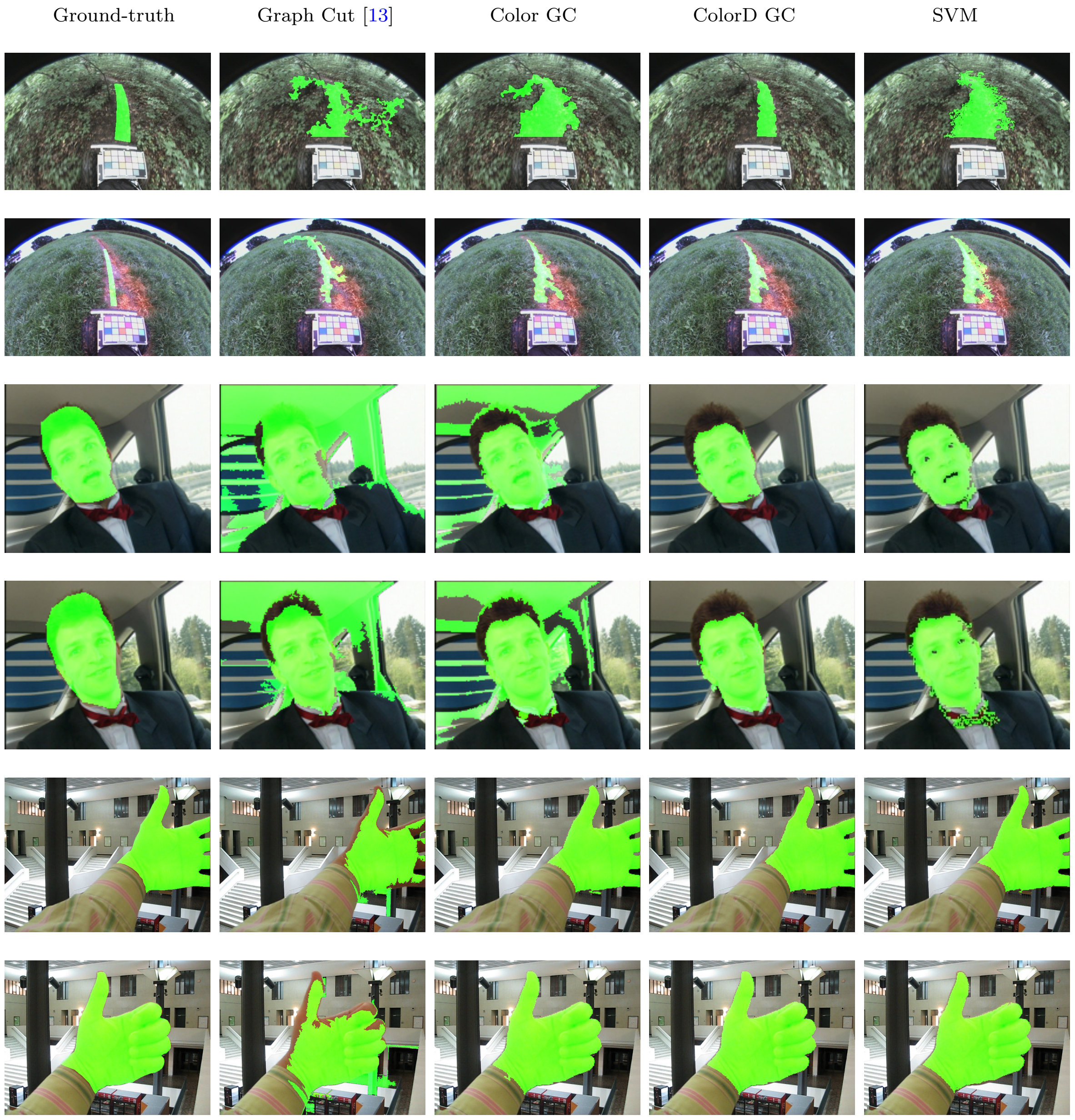


Figure 4.5: Sample results of different refinement methods for the images whose ground-truths are shown in left-most column. Column headings show the method names. First two rows display the results from *Trail*, the next two rows from *Head*, and the last two rows from *Hand* datasets.

The Methods	*Median Overlap Score*
Only RGB	0.673
Only Depth	0.62
RGB + Depth	0.671
GrabCut [157]	0.663

Table 4.3: The overlapping scores of *GC-Refine* and GrabCut [157] for *NYU-KinectDataset-Refine*. First three rows denote the score of the test in which the corresponding cues are incorporated in *GC-Refine*.

4.2.4 Results of the Household Objects

In the availability of the depth data, *ColorD GC* is extendable to fuse some other other generic cues, such as the depth and normal of a point, into the refinement process. The details of this extension is explained in Chapter 5 and this refinement method is called as *GC-Refine*. Simply, k-means clustering is performed by putting all cues to the same feature vector. *GC-Refine* is also experimented with a subset of *NYU-KinectDataset* [128] [168]. This dataset has of 2347 unique frames including 64 different indoor environments, such as residential apartments, living rooms, bedrooms, bathrooms and kitchens. The objects in this dataset were manually point-wise annotated. It was recorded by a Kinect setup system, so the registered color and depth images of the frames are available. The problem of refining other kinds of objects as in *NYU-KinectDataset*, if their low dimensional representation is also possible. In this case, it is assumed that their low dimensional representation is a bounding box around the objects.

A new subset of this dataset, called as *NYU-KinectDataset-Refine*, is formed by selecting 30 images which include different household objects, such as microwave, air conditioner, flower vase, pillow, kettle and etc. The bounding boxes around the objects are generated by leaving a certain of margin between the sides of the bounding box and the point-wise ground truth of the objects. The objects in this dataset were refined by *GC-Refine* which is explained in Chapter 5.2.1.1 by providing generated bounding boxes and incorporating different combinations of the rgb color and depth of the pixels. The median overlapping scores were computed for each of the tests. These scores can be seen in Table 4.3. The best performance achieved using only RGB color information. Combining the depth information with the color did not help to improve performance. This is mainly because of not employing a pre-processing steps to obtain more representative object models for *GC-Refine*. Since the points just around the object have the similar depth values, *GC-Refine* with the depth information was unable to outperform the case which only utilizes from the rgb color information. In addition, the results of *GC-Refine* are compared to *GrabCut* [157]. It was observed that *GrabCut* performs slightly worse than *GC-Refine*. Some sample results of this experiment with different combinations of the cues are shown in Figure 4.6.

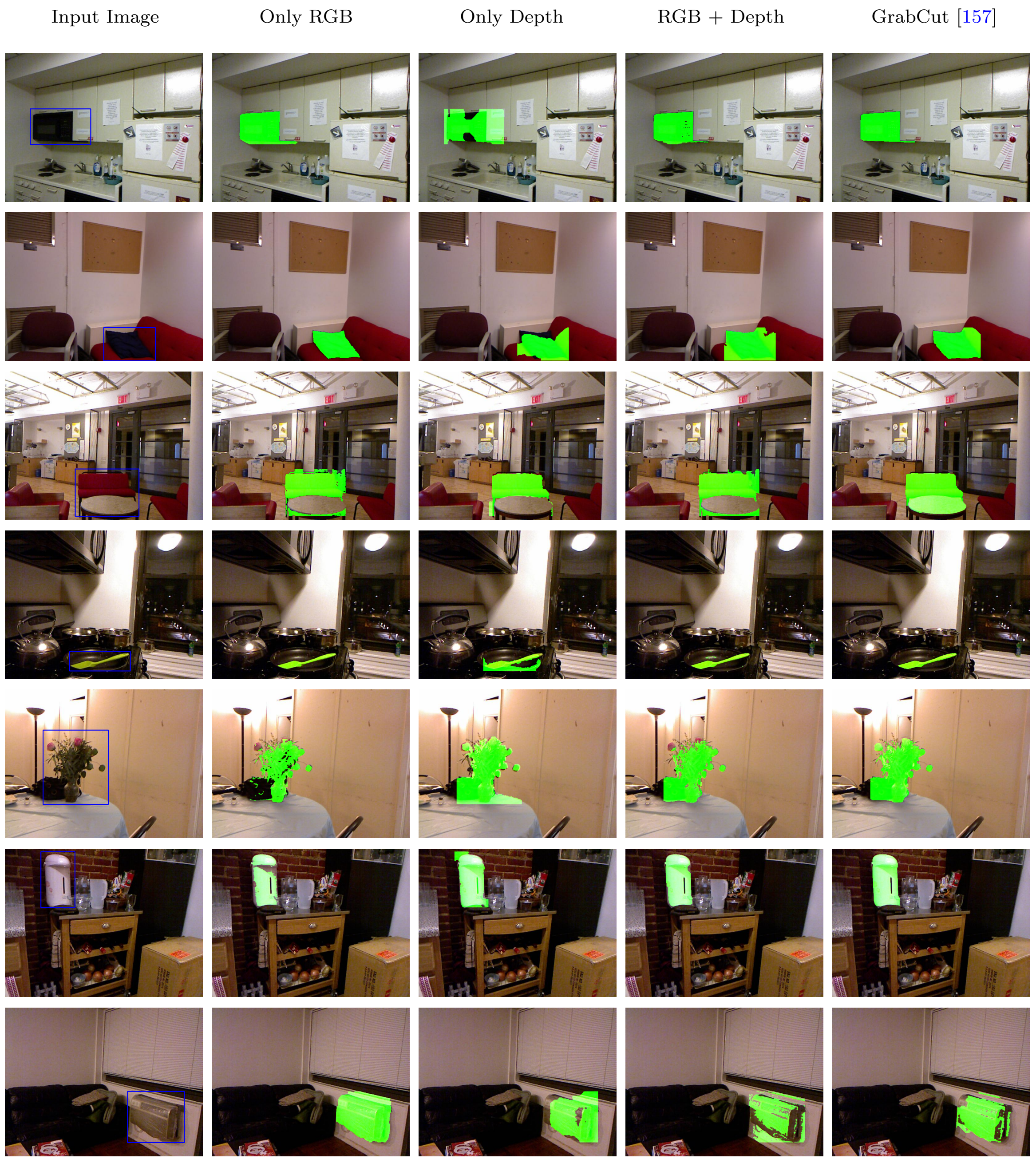


Figure 4.6: Sample results of *GC-Refine* and GrabCut [157] for *NYU-KinectDataset-Refine*. Column headings show the used combinations of the cues for *GC-Refine*.

4.3 Conclusion

Refining the rough estimated shapes of an object provided by some other algorithms is useful for the applications which require the detailed border, in other words, non-parametric point-wise representation of an object. For example, an unmanned ground vehicle can produce more accurate motion trajectories, if it knows exact the detailed borders of the trail in which it drives. Therefore, on one side of this chapter, the performance of several transformed methods which can refine automatically foreground regions for the purpose of robot trail following are compared and analyzed.

Trail refinement methods are developed by making some changes on graph cut, graph cut with distance penalty, GrabCut and super pixel segmentation algorithms. The refinement methods require to take initial information about the foreground region. These information contain cues about the color, shape and spatial position of the region in a given image. Initial color and shape models of the foreground and background regions by using these priori information are constructed. The performance of the algorithms are analyzed on several long sequences with diverse appearance and structural characteristics, and in addition to this set, with a set of trail images collected from the web. Ground-truth segmentations are used to quantify performance where available. Median overlap scores between the ground truths and the results of the methods are computed for both of the test sets. The graph cut method which penalizes distant points to the initial estimated trail region outperformed the other methods in the experiments.

In order to build a refinement method which is more robust to different illumination conditions and to handle other kind of complex objects, a novel graph cut based algorithm is developed. This method has the ability of switching between different color spaces dynamically. It computes the most distinctive color space between the background and foreground models of graph cut by a pre-processing step. The working color space of graph cut is set to most distinctive color space adaptively. The best color space is chosen among RGB, LAB, AB, and L. The pixels are clustered by *k-means* algorithm before the histograms of fore/background models are formed. The frequencies of *k-means* labels are accumulated into the fore/background histograms.

In contrast to a previous graph cut method which utilizes from shape distance penalties of the points in the image space by weighting them constantly, the proposed method employs a dynamic weighting mechanism. According to the distances between the models of graph cut which is computed by the color space selection step, the distance penalty map is adaptively weighted in the run time. This feature of the algorithm provides it to be able to capture distant object points if there is high color similarity around the object region, or it stays close to the object region if the dissimilarity is high.

This novel method is experimented with three different datasets. One dataset consists of 17,358 trail images recorded by an omni-directional camera in a mixture of field and forested terrain. Another one includes 5,616 frames of a video in which a hand is gesturing in front of a complex background. Last dataset includes 383 frames of a human head moving in a car. The ground truths are generated for 5-10% of three datasets to quantify of the performance the proposed method. The results are compared to standard graph cut technique and a SVMs based refinement method. The proposed method outperformed the compared algorithms for the results of three datasets. Moreover, the result of the proposed method is experimented with the images of household objects, such as air conditioner, flower case, pillow, kettle and etc. In this case, it is assumed that a bounding box around the objects are given as the initial rough shape estimations of the objects. The proposed method outperformed GrabCut which is known as the best suitable method for this purpose.

Chapter 5

REFINEMENT OF LOW DIMENSIONAL HUMAN SHAPE REPRESENTATION

There are many benefits of detecting and tracking the humans around an intelligent system. For example, an autonomous car driving on the streets requires to locate the pedestrians not to hit them. Knowing the exact position of the humans around itself provide useful information to the motion planner of the car. Since a human is a special type of obstacle, the motion planner might put some constraints to give more importance of avoiding from the humans while calculating the trajectories. A given rough shape estimation of the human in the low dimensional space, such as a bounding box, to the motion planner is enough in many conditions to produce human hit-free motion plans.

However, an autonomous car might need to know more than just a rough shape estimation of a human in some cases. For example, if there is a police officer at the intersection of a road, his/her gestures mean important traffic commands for the vehicle. In order to understand these commands, instead of a bounding box representation of the police officer, a point-wise representation which provides exact detailed borders of him/her is more desirable. This type of representation excludes the points which are the non-region of interests in the scene, consequently it provides a better human representation which is more beneficial for the method of understanding the commands of the police officer.

Recognition of the human gesture is not only necessary for the autonomous cars, but there are a wide range applications which might need to understand the gesture of the people. One of the them can be the applications which convert the sign language to the written sentences for helping the handicapped people. In fact, any gesture recognition application opens a gate for the human to communicate to the machines. If a computer can understand the gesture, input devices, such as a mouse, keyboard, and touch screen will not be required. Also, remote control of the devices will be possible by interpreting the gestures. It can generate the home environments in which someone can open the curtain of the room by clapping his/her hands.

Computer surveillance systems are commonly used in the airports, shopping centers, buildings, and the roads for the security purposes. Detecting and tracking the people is some functions of these system. More importantly, their one of the other functions is understanding the behaviors of the people which do some suspicious activities, such as detecting of an unattended bag in which there might be a bomb. Another purpose of the surveillance systems can be the understanding of the movements of the old people living alone in their home. In the case of an urgent situation of their health, the hospitals or their relatives can be informed. The applications which detect automatically these type of the activities help the officers who monitor the scene and save their time to do this cumbersome work. These applications inevitably need the point-wise mask of the people in the environment.

Kinect has became so popular in last years for in-home game players. The availability of the interaction between the player and the game allows the game developers to produce more realistic and enjoyable games. The depth camera employed in Kinect system give the opportunity of obtaining 3D shape features of the environment. This helps to build more accurate object perception algorithms. Better object perception algorithms yield better understanding the commands and gestures of the human to play the game. For example, an interactive soccer game need to know where the foot of the human in front of the camera. Also, it should be able to track the foot to detect the event of kicking the virtual ball. In addition to a soccer game, some other interactive games, such as a baseball, tennis, and bowling need to detect and track different parts of the human body. The detailed border of the body parts are more desirable for these interactive games. This can be achieved by the point-wise representation of the human body.

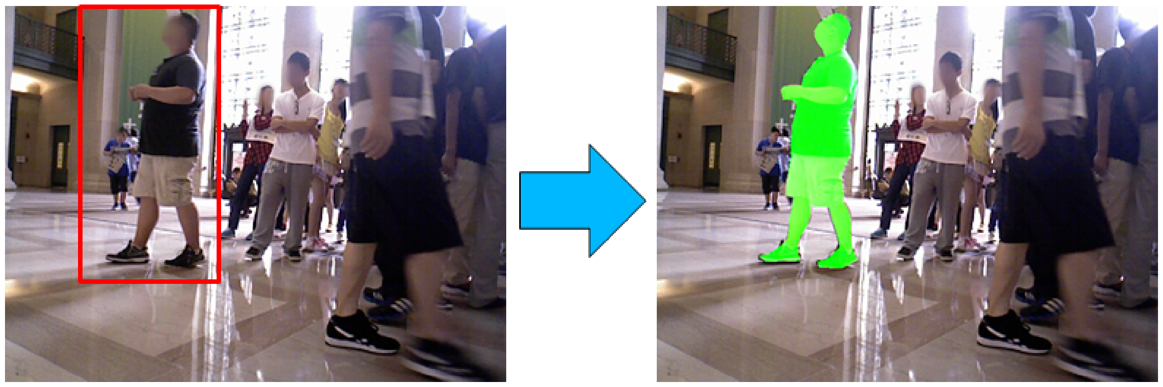


Figure 5.1: A given low dimensional representation of the human which is a bounding box in this case is refined to obtain more detailed point-wise representation.

A rough low dimensional representation of the human, such as a bounding box, is output by some human detection and tracking algorithms, (cite these algorithms). Even though some of these algorithms are so powerful and robust, low dimensional representation is not enough for the applications described above. Point-wise representation of the human is necessary for these applications. The detection of a human in point-wise is difficult problem in different and cluttered environments. Designing a point-wise human descriptor which can represent any point on a human is nearly impossible because of the variance of the background. However, it is possible to utilize from the low dimensional representation of the object to achieve its point-wise representation.

A bounding box representation of the human carry some useful information about the human. First of all, this representation guarantees that the human is inside of this box. It is possible to start from this initial rough estimation to obtain the exact detailed human borders. This process can be called as the refinement of the low dimensional representation of the human. This refinement process is a transition way of going from low dimensional to point-wise representation of the object. Its output is a mask which includes all the points belong to the human. This mask can be given to the applications described above as the initial input. An illustration of this process can be seen in Figure 5.1.

The refinement of low dimensional human representation is a challenging problem. Several reasons make refinement process difficult. Representation of the human, in this case a bounding box, not only contains the human points but it also includes some background points. The background in the bounding box might have similar colors, texture or 3D geometric features. It is difficult to learn a common human color or texture model. The background points need to be eliminated from the human color model. Also, the human might stand in any position in the bounding box. He/she might sit, stand, walk, open his/her arms, or bend. The camera might take the pictures from any position, so it might cause the pose variance of the human and cause the occlusion of the human body parts. These all factors make the refinement process complicated.

Some factors make the refinement of the low dimensional human shape different than the refinement of some complex objects, such as trail, human head and hand, explained in Chapter 4. The first main difference can be seen in the representations of the shapes of the objects. The provided shapes of the objects to the refinement process in Chapter 4 have tighter bounding polygons. They usually have a few background points, but almost everything inside of it is actual true points of the object. If the given polygons are scaled down by a small factor, it is possible to obtain representative models of the objects. Whereas the bounding box representation of a human is a looser bounding polygon. It covers outside of the human in the scene. Therefore, it has much more background points. A simple scaling process might not produce clean and representative human models as for the complex objects in Chapter 4. Small scaling factors of the bounding box causes including more background points, but large scaling factor causes to loose many of the human points in the models.

In addition to using only color images in Chapter 4, the features obtained from the depth image of the scene is going to be incorporated into the human refinement process in this Chapter. These features are going to be the depth, the normal of the points and also point-wise shape descriptors which are specific to the human body.

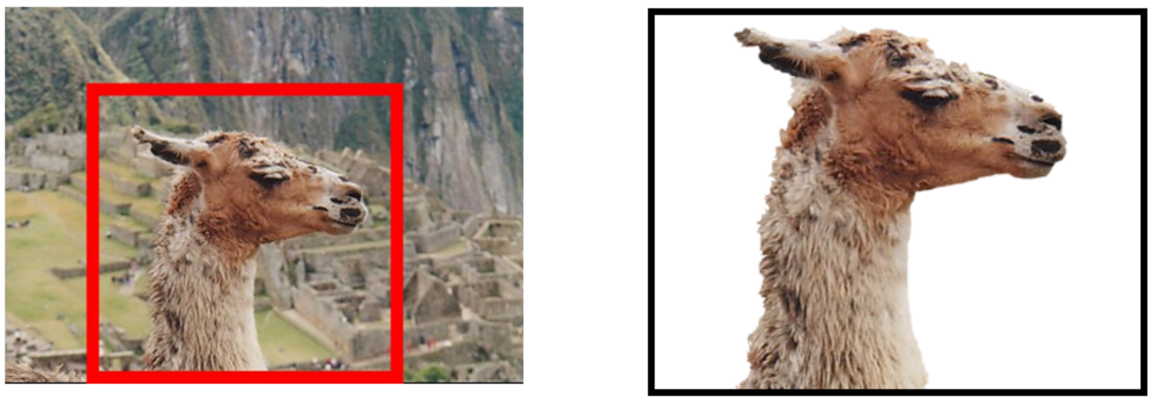


Figure 5.2: In *GrabCut* [157] algorithm, a user initialize a rectangle around the object that he wants to segment. It outputs the point-wise object mask.

5.1 Related Work

To best of my knowledge, currently there is no work exists which directly aims to refine the low dimensional human shape representation to obtain the point-wise mask of the human by using *multi sensor* data. However, *GrabCut* [157] can be considered as a suitable method which uses features obtained only from the color image for this purpose. *GrabCut* is designed as semi-automatic segmentation algorithm. It takes a user drawn rectangle around the object which wants to be segmented from its surrounding background. It builds the Gaussian Mixtures Models for the object and background by using inside and outside of the user given rectangle. Graph cut segmentation method [16] [17] is applied iteratively. At the end of each iteration, the result mask of graph cut is provided to the next iteration to set new fore/background models. Border matting is performed to achieve smooth and more accurate results at the end of each segmentation.

Humanising GrabCut [56] is a specialized version of *GrabCut* method which refines the given low dimensional representation of the human in the image. This method incorporates only color information to its refinement process by training a classifier. Even though it utilizes from the depth data of a Kinect sensor to generate the pixel-wise ground truths of the humans, they do not use the depth information in

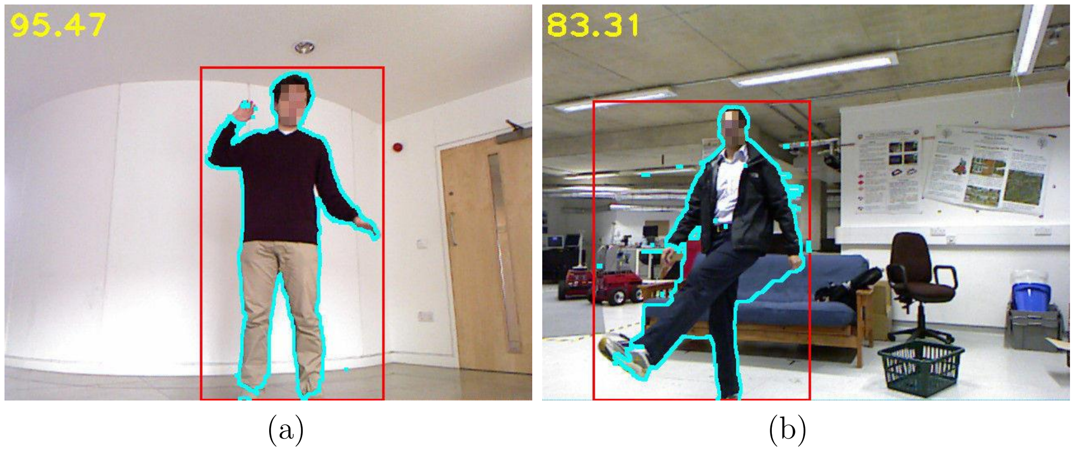


Figure 5.3: Some results of *Humanising GrabCut* [56]. The red bounding box indicates the given low dimensional shape around the human, and the cyan drawn silhouette denotes the refinement result. The numbers on the left top side of the images show the overlapping scores between the result and the ground truth of that image. Since this method utilizes only from the color information, it is obvious to see that the color similarity in the background of (b) diminished the performance.

the refinement of the human. The authors of [56] collected their data by not moving their setup, so they were be able to use a background subtraction method employed in OpenNI library to obtain the ground truths. The background subtraction method extracts the region of interest by only using the depth data of Kinect. [56] registers the color and depth images to achieve the corresponding ground truth in the color image.

Someone can inspire from the depth image based background subtraction method used in [56] also to refine the low dimensional human representation. However, there are differences which make the problem of the human refinement more complicated in our case. The main issue is that the camera position is not assumed as stationary, so background might change from one frame to another. For example, what if the camera setup is mounted on a moving vehicle. In that case, their method will be unable to extract the background due to changes in the scene. Therefore, the approaches which will be explained in the next sections do not assume that the images are provided by a stationary camera setup. Some sample results of this method can be seen in Figure **??**.

[59] and [58] describe a spatio-temporal *GrabCut* based method to segment the humans given in an image. The humans are detected by cascaded HOG classifier [36]. The faces of the humans are detected by Haar-like features [195]. The seed points of *GrabCut* are initialized according to detected humans and their faces. In order to incorporate the spatial coherence, instead of building *k-means* GMM color models, a strategy which uses Mean Shift clustering is chosen to form the GMM models of foreground and background.

The human silhouette is obtained by a bottom up approach inside a human detection window in [115]. The window in which there is a human is divided into blocks. A confidence value which defines that there is a part of human in that block is assigned by HOG classifier [36]. A canny edge detector is run on the window and the strong contour segments are achieved. A graph whose vertices are the contour segments is constructed. The weights of the edges of this graph are set by a energy function which employs the likelihood information coming from the HOG classifier, and the spatial distance between the contours. It is assumed that the silhouette of the human forms a cycle in this graph. The optimal cycle in this graph is found by Dijkstra's shortest path algorithm. Several iterations of this process ensures that optimal cycle which represents the human in the graph is achieved, if some of the silhouette contours are missing.

A local shape feature based human classifier is proposed in [202]. In this method, edgelet features are used to represent the local shape information of the information. First, a classifier detect the the region of the interest of the object. A second classifier classify the foreground pixels around the neighborhood of the edgelet. In this way, a cascaded detection and per-pixel object classification method is achieved. This methods is tested with pedestrian human dataset.

The pedestrians are simultaneously detected and segmented by integrating appearance and motion cues in [161]. The contour features which employs the silhouette information is learned from the training data. This method combines a bottom-up and top-down approaches in a coherent manner. A Markov Random Field is formed to formulate the labeling problem of the contours. The edges between the vertices are assigned according to the similarity of contours to a human contour, the spatial distances between them. The silhouette of the human extracted by a loop closure operation among the contours.

[146] proposes a human segmentation method from the indoor video image sequences. This method can be considered as a variation of background subtraction approach. It forms a texture and color models for each pixel in the image. It removes the shadows to make the algorithm more robust. This work only works for the stationary cameras.

The label discontinuity information of neighbor pixels are not integrated in [165]. [147] describes a human limb segmentation framework based on Random Forest and graph cuts in depth map images of videos. Random forest estimates the probability of the pixels being a human or not. These probabilities, also can be called as confidence scores, are carried into the building of a conditional random field graph. The unary term of the graph is set to the confidence scores. The graph cuts is performed to assign the labels of the pixels spatially and temporally.

A model based human segmentation approach in crowded scenes is described in [210]. This method assumes that the camera which captures the images of the scene is stationary. Various features, such as human shape, human height, and camera model, are all integrated in one Bayesian framework. The solution of the system is obtained by and efficient Markov chain Monte Carlo method.

[192] explains a human segmentation method which uses utilizes from different human body part detectors. This approach constructs a conditional random fields which includes pixel-wise features in the image and also high order informations about the human. The shape prior information is incorporated to bias the segmentation to human body shapes. High order potentials in the formulation of the conditional random fields are defined as the segmented human body parts. Graph cuts is performed to obtain the human segmentation.

A human segmentation algorithm which fuses the information retrieved from color and thermal camera is presented in [208] [209]. In this method, the background subtraction approach is applied in both color camera and the thermal camera images. These two sensors are registered in the same coordinate system. A Gaussian distribution models the temperature of the human. The background model of each pixel in the color image is described by a list of codewords. Equal weights are assigned at the fusion of these two different features to segment the human.

A body part labeling method is described in [165] [166]. This method presents a point-wise descriptor which collects structure information around the interest point. Structure information is obtained from the depth data of the scene. This algorithm runs in Kinect devices nowadays.

5.2 Approaches

Several different approaches can be developed to solve the human refinement problem if a low dimensional representation of the human shape is provided. In this section, the methods which will be discussed take a bounding box representation of the human as the input. It refines the bounding box, *B(x,y,w,h)*, where x and y is the top left point, w is the width, and h the height of the box, to obtain the detailed pixel-wise representation of the human.

There two main types of approaches which can be followed to refine the low dimensional representation of the human. In the first way, only the local cues inside the provided bounding box can be used. No other prior information is incorporated. These types of approaches are explained in Section 5.2.1. In the second way, a classifier can be trained to incorporate some prior information only specific to the human. These prior information can carry general shape, skin color, texture features of a different humans. This type of the methods will be explained in Section 5.2.2.

5.2.1 Refinement Using Only Generic Cues and No-Prior Information

It is possible to develop some pixel-wise human refinement algorithms without incorporating any prior information. In these approaches, the models that carry the cues about human and background are obtained from a single input image. These cues might be the generic cues to refine or segment any object. Some methods can apply a pre-processing step to clean as many as non-human points inside the given bounding box to achieve more reliable foreground models. On the other hand, all points inside the bounding box can be used to form the human model.

5.2.1.1 No Pre-processing: Refinement with Iterative Graph Cuts

The graph cut method explained in Chapter 4.2 and [87] takes a coarse estimate of mask by initialized an user or another method. Then, this method segments the desired object iteratively applying graph cuts.

[87] constructs the fore/background models using just the color images, no features retrieved from 3D point-cloud of the scene are utilized. This work can be extended easily to incorporate any other cues into the refinement process. In order to deploy the other useful features, the size of the feature vector must be increased.

In this method, a feature vector, $v = (v_1, \ldots, v_m)$, is calculated for each pixels in the image, or the points in the point cloud. m indicates the dimension of the feature vector. All feature vectors, v's, are clustered into k different labels by applying k-means algorithm.

A human region feature vector, v, distribution, M_H, is modeled by a histogram $h = (f_1, \ldots, f_k)$ of the label frequencies inside it. The background region model, M_B, is formed in the same way. The model dissimilarity, d_m, between M_B and M_H is measured by histogram distance function which is chi-squared metric $\chi^2(h_i, h_j)$. This dissimilarity value is going be used a confidence measurement to weight another feature, called as distance penalty, which will be employed in the graph cut and explained in the next section.

The regional and boundary term of modified graph cut explained in Section 4.2 are recalled by stating that the object segmented in the formulation is the human body:

$$R_i(''Human'') = -ln(\frac{Pr(l_i|M_H)}{Pr(l_i|M_H) + Pr(l_i|M_B)}) \tag{5.1}$$

$$R_i(\text{"}Background\text{"}) = -ln(\frac{Pr(l_i|M_B)}{Pr(l_i|M_H) + Pr(l_i|M_B)}) \tag{5.2}$$

$$B_{(i,j)} = |R_i(\text{"}Human) - R_j(\text{"}Human\text{"})| \tag{5.3}$$

where i and j are two neighbor pixels, l_i is k-means label of pixel i, M_H and M_B are the histogram label frequencies of the human and the background regions in the image, respectively.

In order to form the human model, M_H, given bounding box, *B(x,y,w,h)* is shrinked by a factor of S_d. All points inside the shrinked bounding box, $B_{sd}(x_1, y_1, w_1, h_1)$, are used to form the human model, M_H. The points to obtain background model, M_B, is calculated first by scaling up *B(x,y,w,h)* and yielding a larger bounding box, B_{Large}. Then all the points which are inside B_{Large}, but not in *B(x,y,w,h)* are used to form M_B. These models in a given image are shown in Figure 5.4.

Distance Penalty Map Construction and Weighting: Globally optimal segmentations are produced by graph cut. In our problem, in addition to color, surface normals, and depth information, it is possible to retrieve a useful information to employ in the graph structure from provided bounding box. It is less likely that the pixels which are far away from the center point, $Mid_B(x, y)$, of the bounding box, *B(x,y,w,h)*, are going to belong to the human body. As explained in Section 4.2.2, this type of penalty can be incorporated into the graph cut by constructing a distance penalty map. The distance map is constructed as: $Map_{dist}(i) = \|i - Mid_B(x, y)\|$, where point i is a pixel in the image space. $\|i - Mid_B(x, y)\|$ is the distance between $Mid_B(x, y)$ and pixel i. Figure 5.5 shows the distance penalty map of a given image.

The dissimilarity value between human and background models, d_m, can be used to weight the distance map, Map_{dist}. Weighting Map_{dist} with d_m biases the graph cut to include far points carefully if the dissimilarity between the models is large. The

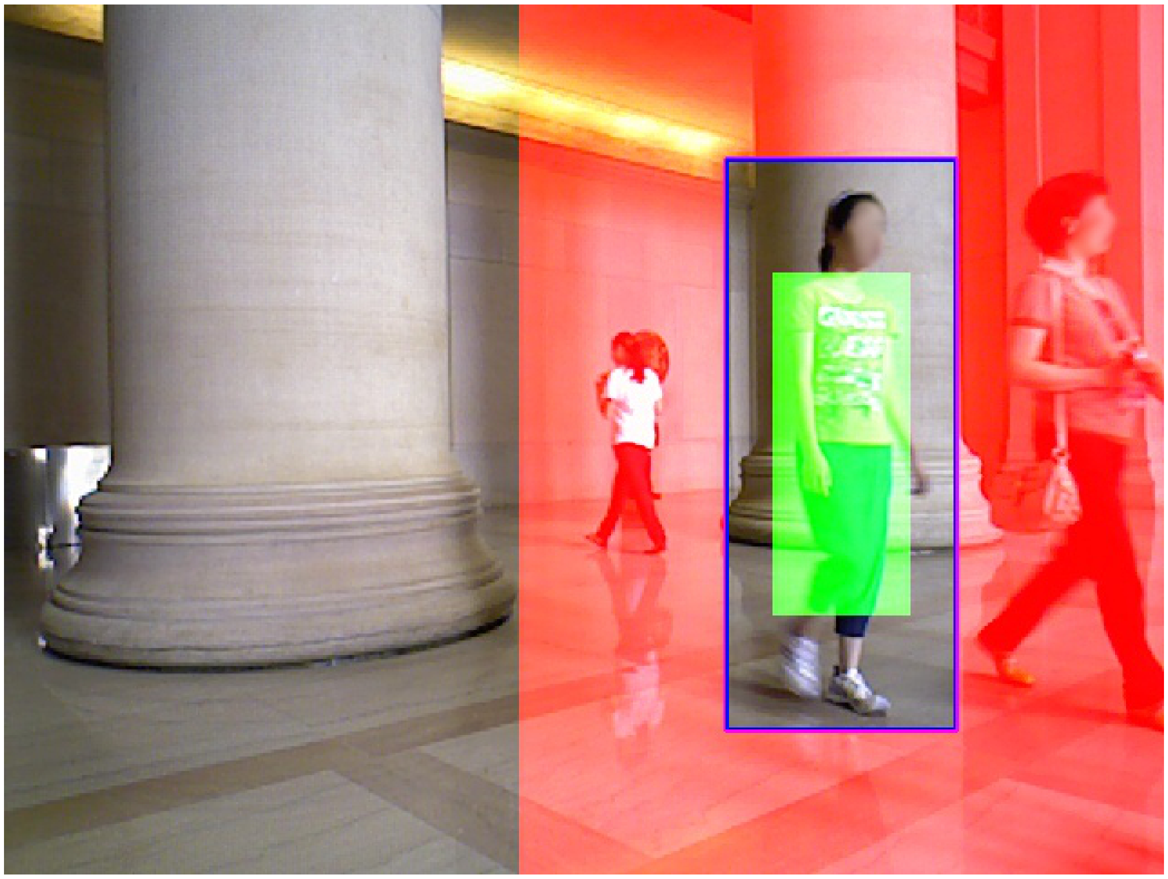


Figure 5.4: The human and background models provided to the modified graph cut. The initial given bounding box is drawn as blue. Green pixels are used to form the human model. Red pixels are for the background model.

distance penalty function formulated in Section 4.2 is brought to add into the *regional term* of the graph cut algorithm :

$$R_i(\text{"}Human\text{"}) = Pr(P_i|O) + \beta(\alpha(d_m)Map_{dist}(i)) \tag{5.4}$$

where $Pr(P_i|O)$ is the penalty of adding pixel i to the human region. β is a constant term to set the relative influence. $\alpha(.)$ is the negative log-likelihood function to weight the distance penalty adaptively. d_m is the dissimilarity value between the models.

The Cues: The feature vector, $v = (v_1, \ldots, v_m)$, can be formed by the cues obtained from the color and depth images of the scene. In this algorithm, 3 different cues are combined in v. The color information is incorporated into the feature vector,

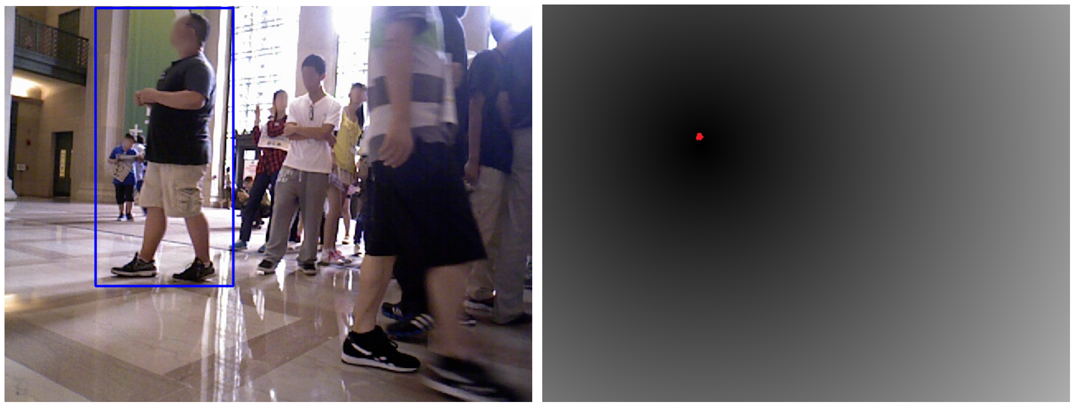


Figure 5.5: Right image shows the distance penalty map of given left image. Bounding box is drawn as blue in the left image. The center point of the bounding box is colored as red in the distance map.

v, simply taking 3 RGB channels of the pixels in the image. The depth value of the corresponding point is added another dimension to v. In addition to color and depth information of a pixel, local surface information of a point can be included in v. In order to put this information, the depth image is first converted to the 3D point cloud. Then, the normal of each point, $p_n = (n_x, n_y, n_z)$, is estimated in the point cloud. The neighborhood search of the points is performed by building a FLANN-based Kd-tree [124] to reduce the computational time. All dimensions of the p_n is added to the final feature vector, v.

Figure 5.6 displays the incorporated cues for some sample images. Figure 5.6(a) shows the original color images, Figure 5.6(b) displays their corresponding depth images. Depth images are colored in blue shade to make more visible and to show invalid depth points as black for the illustration purpose. Dark blue points are more close to the camera location and lighter blue points are far from the camera. The black colored points are the locations where there is no depth information is provided by the infrared camera. One of the reasons is that some points in the scene are out of range of the infrared camera. For example, some far background points do not have depth information. Also, at some points where there are strong sun-light on shinny surface, such as on the left side of the image in the third row of Figure 5.6, the infrared camera is unable to provide depth information. In addition, because of the geometry of the human hair, or type of the clothes which do not reflect well the infrared lights, the depth is not available for some points on the human body.

The normals of the scene points are showed in Figure 5.6(c). In order to illustrate the normals, $p_n = (n_x, n_y, n_z)$, in a colored image format, each dimension of the normal, p_n, whose range is between [-1, 1] is transformed to the range of [0, 255]. In this way, it becomes possible to visualize point normals in the image format. n_x, n_y, and n_z dimensions of a point normal correspond to the red, green and blue channels of the image, respectively. The normals are not computed where there is no depth information is available.

The refinement approach described in this section is called as *GC-Refine* for further references.

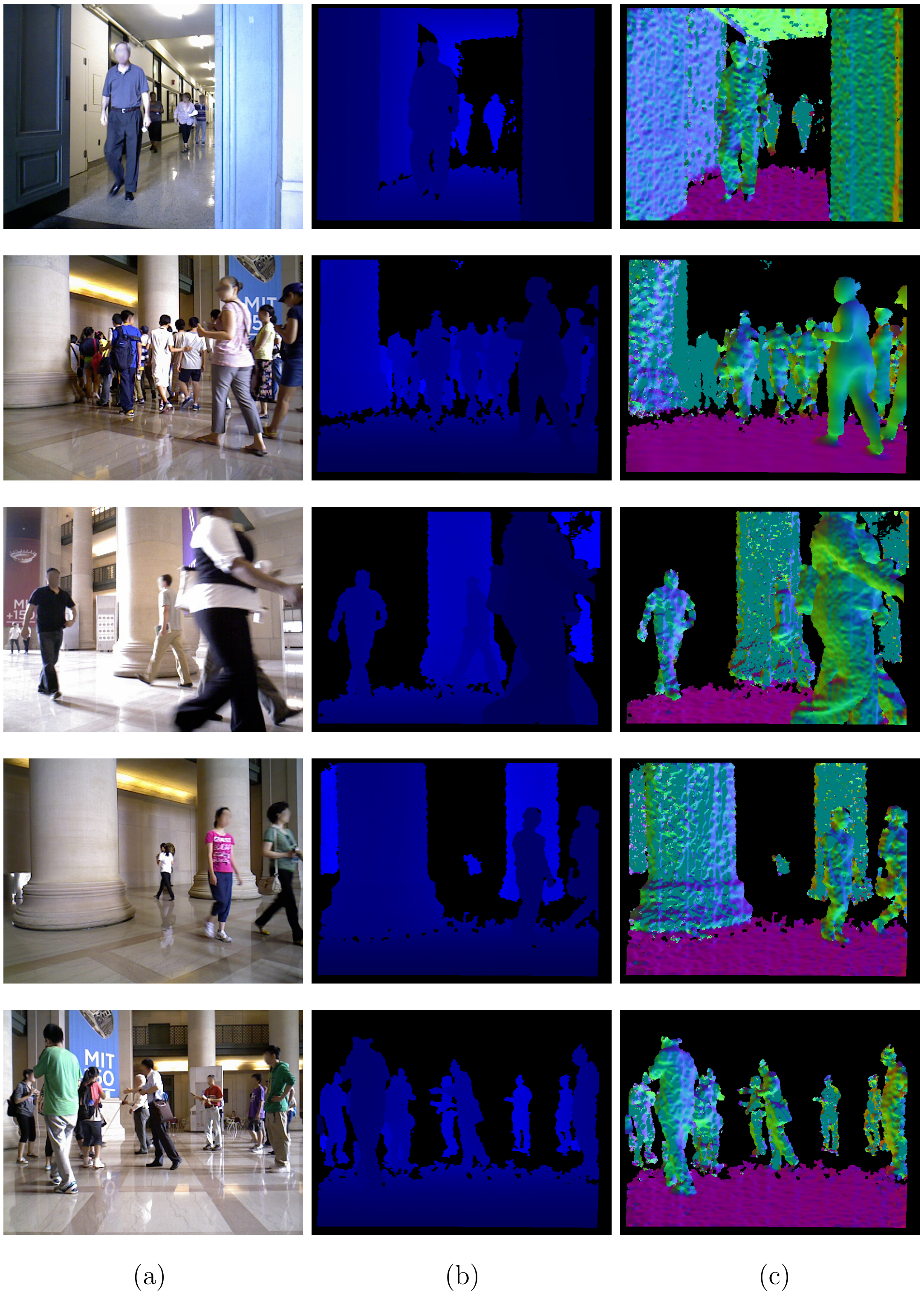


Figure 5.6: The illustration of the cues for sample images. Please see the text, Section 5.2.1.1, for the details.

5.2.1.2 Pre-processing: Ground Plane Fitting and Background Elimination

GC-Refine method described in the previous section builds the human and background models by scaling provided bounding box, *B(x,y,w,h)*. Scaling of this representation is not the best approach to construct robust models. Using a small scale factor causes to include some background points into the human model. Or choosing a large scale factor can eliminate some of the human body points. In order to reduce the number of the background pixels in the human model, some pre-processing steps can be applied.

Removing Ground Plane Points: *B(x,y,w,h)* contains some ground plane points at the foot level of the people in the scene. Refining these points are problematic in several perspectives. They have almost similar spatial distances to the middle point of the bounding box, *B(x,y,w,h)*, as the human foot points, so using the distance map approach explained in the previous section can not entirely solve this problem. Also, the normals of some points on the human foot or shoe, and the ground points can be similar which are sticking up. Also, they have same depth distances. Therefore, segmenting the points at the foot level of the human body requires special attention.

The ground plane can be estimated in the point cloud of the scene by assuming that the ground plane points stays under a certain level of the height, G_h. The points which support a sample consensus plane model under G_h are segmented as the ground plane. Not only the points inside *B(x,y,w,h)*, but all of the points in the scene are used to estimate the ground plane. In this way, more inliers allow better segmentation of the ground plane. The estimated ground plane points are excluded from the bounding box, *B(x,y,w,h)*. For a given image, the estimated ground plane in its 3D point cloud and excluded points from the human model, M_H, are displayed in Figure 5.7.

Extracting region of interest: Elimination of ground plane points from the human model, M_H, cleans non region of interest at the foot level of the humans. However, there might be some background points higher level than ground which are just behind the human. These points can belong to a wall, an object, or another

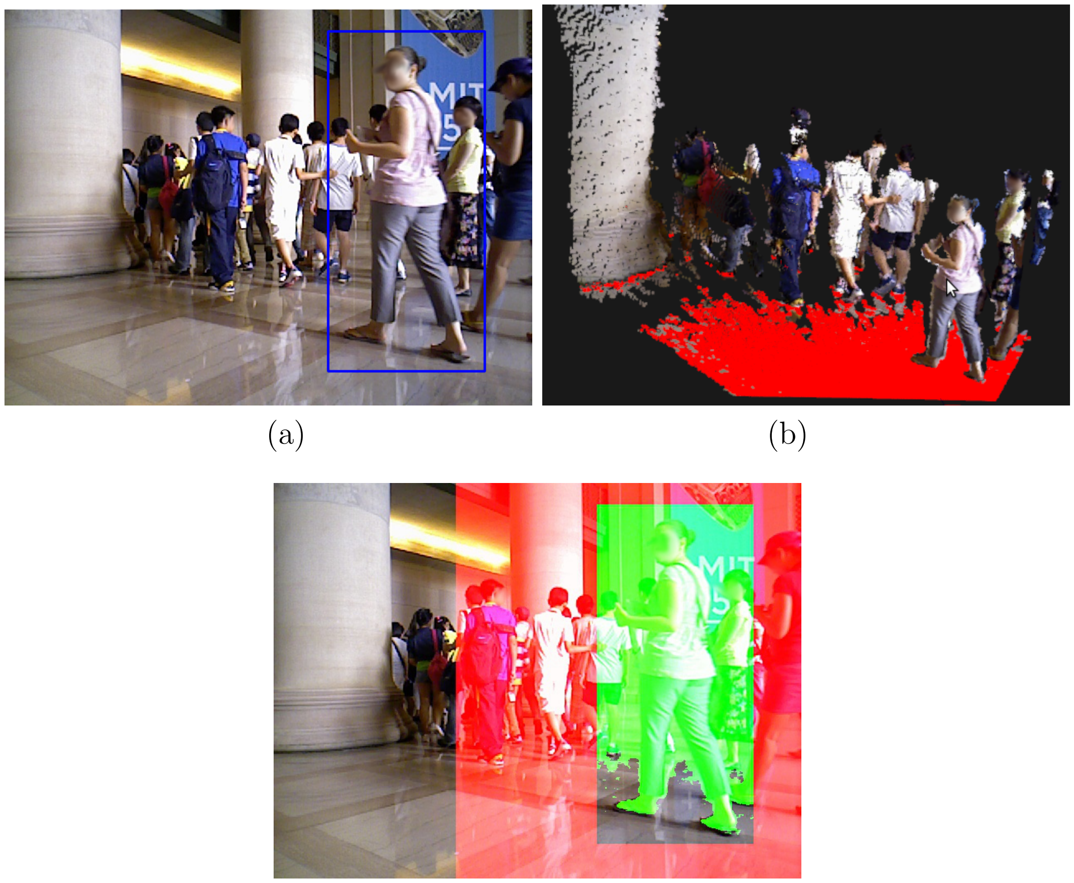


Figure 5.7: Ground Plane subtraction from the models. (a) shows a given image with a bounding box around a human, (b) displays its corresponding 3D point cloud with registered color of the points and the estimated ground plane. An elevated view of the scene from above in PCL viewer shows the estimated ground plane points colored as red. (c) is the final human and background models after excluding the ground plane points. All green colored points forms the human model, M_H, and all the points in red area build the background model, M_B.

human. Therefore, removing these points from the human model helps to achieve more accurate refinement results. To remove these points, first it is assumed that a human has a maximum radius of d_H. A slice of region which is perpendicular to the ground and whose width is d_H is searched to extract the region of interest within *B(x,y,w,h)*. The slice which holds the maximum number of points is selected to form

M_H. These points can be estimated by a Random Sample Consensus procedure as outlined in Algorithm 2.

Algorithm 2 Extracting ROI

```
    Inputs: $Pts = (x_1, y_1, z_1), ..., (x_n, y_n, z_n)$ points in $B(x, y, w, h)$, $d_H$
    Output: $RegionPts = (x_1, y_1, z_1), ..., (x_m, y_m, z_m)$, the inlier points
 1: * Compute $L$, the iteration number of RANSAC
 2: for k=1, ... , $L$
 3: * Choose a random point, $p_k = (x_k, y_k, z_k)$, in $Pts$
 4: * Set, $c_{in} \leftarrow 0$, the number of inliers
 5: * Set, $pts_{in} \leftarrow empty$, the inlier points
 6:   for each $p_i = (x_i, y_i, z_i)$, in $Pts$
 7:   * Calculate the distance, $d_z = |z_k - z_i|$, between $p_k$ and $p_i$ along z axes
 8:     if $d_z < d_H$
 9:     * Add $p_i$ to $pts_{in}$
10:     * $c_{in} \leftarrow c_{in} + 1$, increment the number of the inliers
11:  if $c_{in} > c_{max}$
12:  * $RegionPts \leftarrow pts_{in}$, copy inliers to the return list
13:  * $c_{max} \leftarrow c_{in}$, set max number of the inliers
14:  * $p_{max} \leftarrow i$, set max index
```

Figure 5.8 shows the extracted region of interest from given depth image in which ground plane points are already labeled by Algorithm 2. After removing ground plane points and extracting the region of interest from *B(x,y,w,h)*, the rest of the points inside *B(x,y,w,h)* are used to form the human model, M_H. The background model is formed by the same process described in the previous section. The refinement task is done again by *GC-Refine* method, but in this case applying the pre-processing steps described in this section. Therefore, this approach can be called as *GC-Refine-Pre.*

5.2.2 Point-wise Classifier

All methods explained in the previous section form the human and background models, M_H and M_B by some pre-processing steps based on the bottom-up approaches. These pre-processing steps might cause to build non reliable models that yield poor refinement results. Instead of estimating the models specific to only given image, it is possible to learn some point-wise features of the human body by training a classifier.

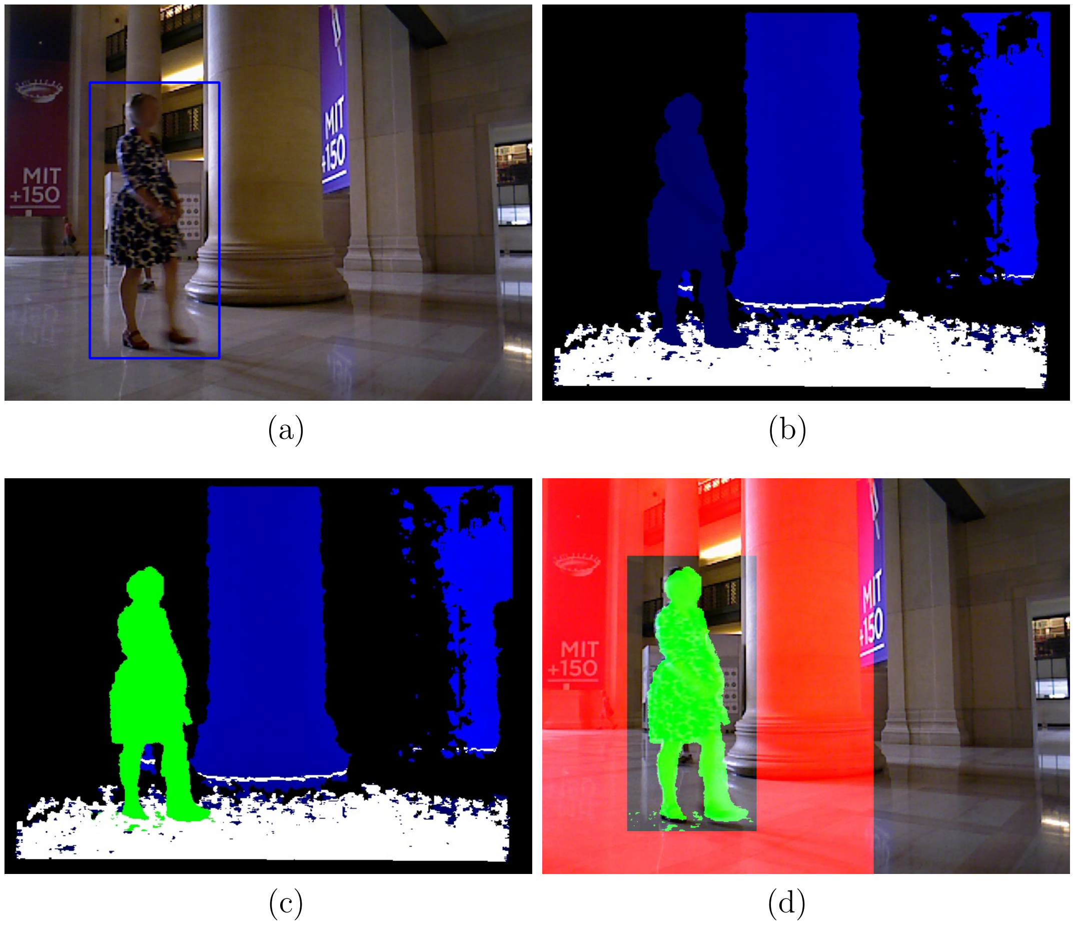


Figure 5.8: Extracting the region of interest to form the human model, H_M. (a) shows a given image with a bounding box around a human, (b) shows the depth image in which the white points correspond to the estimated ground plane. (c) displays the extracted human body points which are colored as green after applying Algorithm 2. Final human model points, colored as green, and background model points, colored as red, can be seen in (d).

These point-wise features are going to be specific to the human rather than a generic spatial cues. One of the main advantage of building such a classifier is that the common cues belong to a human can be incorporated into the refinement procedure. The following sections describe the cues which are obtained to form the point-wise shape descriptor of human body and the details of building the classifier.

5.2.2.1 Point-wise Shape Descriptor

The proposed point-wise shape descriptor, f_s, utilizes from the 3D point cloud data of the scene. It is obtained by the following procedure:

1) Normals: A cue about the local shape information at the surrounding of a point, p_i, can be encoded in the descriptor, f_s, by calculating the normal of p_i, η_i. It is computed for all three dimensions of the point cloud space, so it includes three values for three directions, $\eta_i = (\eta_x, \eta_y, \eta_z)$.

2) Vectorial Spatial Distance: First, the middle point, $mid_B = (mid_x, mid_y, mid_z)$, of the bounding box, *B(x,y,w,h)*, is calculated as formulated in the following equations:

$$mid_B^{2D} = (x + w/2, y + h/2) \tag{5.5}$$

$$mid_B \leftarrow T(mid_B^{2D}) \tag{5.6}$$

where T is the function which gives the corresponding 3D point of a pixel in the image. The vectorial distance relative to the middle point of *B(x,y,w,h)*, $\Delta_v = (\Delta_x, \Delta_y, \Delta_z)$, is computed for the point, $p_i = (p_x, p_y, p_z)$. This computation can be formulated as:

$$\Delta_v = (p_x - mid_x, p_y - mid_y, p_z - mid_z) \tag{5.7}$$

3) Geodesic Distance: As mentioned in [140], geodesic distance between two points on the human body is constant at different poses. This cue is incorporated into our descriptor, f_s. The relative geodesic distance, GD_i, to the middle point, mid_B, of a point, p_i, is computed by Dijkstra's Shortest Path Algorithm. The image of the scene is converted to a graph, $G(V, E)$, where V is the graph nodes, and E is the edges between the nodes. Each point, i, in the image is represented as a node in graph, $G(V, E)$. The neighbors of the each node in the graph is restricted to 4 pixel neighbors from the image. The edge weight, w_{ij}, between two points, i and j, is set to the

Euclidean distance between p_i and p_j in the corresponding point cloud of the scene as in Equation 5.8.

Each point, p_i, in the point cloud is represented as a node in graph, $G(V, E)$. The edge weight, w_{ij}, between two points, p_i and p_j, is set to the Euclidean distance between p_i and p_j as in Equation 5.8.

$$w_{ij} = \sqrt{|p_{i_x} - p_{j_x}|^2 + |p_{i_y} - p_{j_y}|^2 + |p_{i_z} - p_{j_z}|^2} \tag{5.8}$$

If the there is no depth data is available for the neighbor, the edge weight, w_{ij} is assigned to very large distance. Some sample geodesic distance map for the given images can be seen in Figure 5.9. For the illustration purpose, these maps are colored in a way that same intensity values in two different images correspond to the same Euclidean distance value.

As it can be realized that the geodesic distance map contains more accurate distance information than the distance map approach which is computed in 2D image space explained in Section 5.2.1.1 and its example is showed in Figure 5.5. Therefore, it is more representative. Also, it is not possible to see a sphere effect around the middle point of the bounding box in the geodesic distance map. Yet, this can be seen if it is calculated in the image space. However, the advantage of computing the distance map in the image space is that it is computationally so fast. There is no need to run an computationally expensive shortest path algorithm.

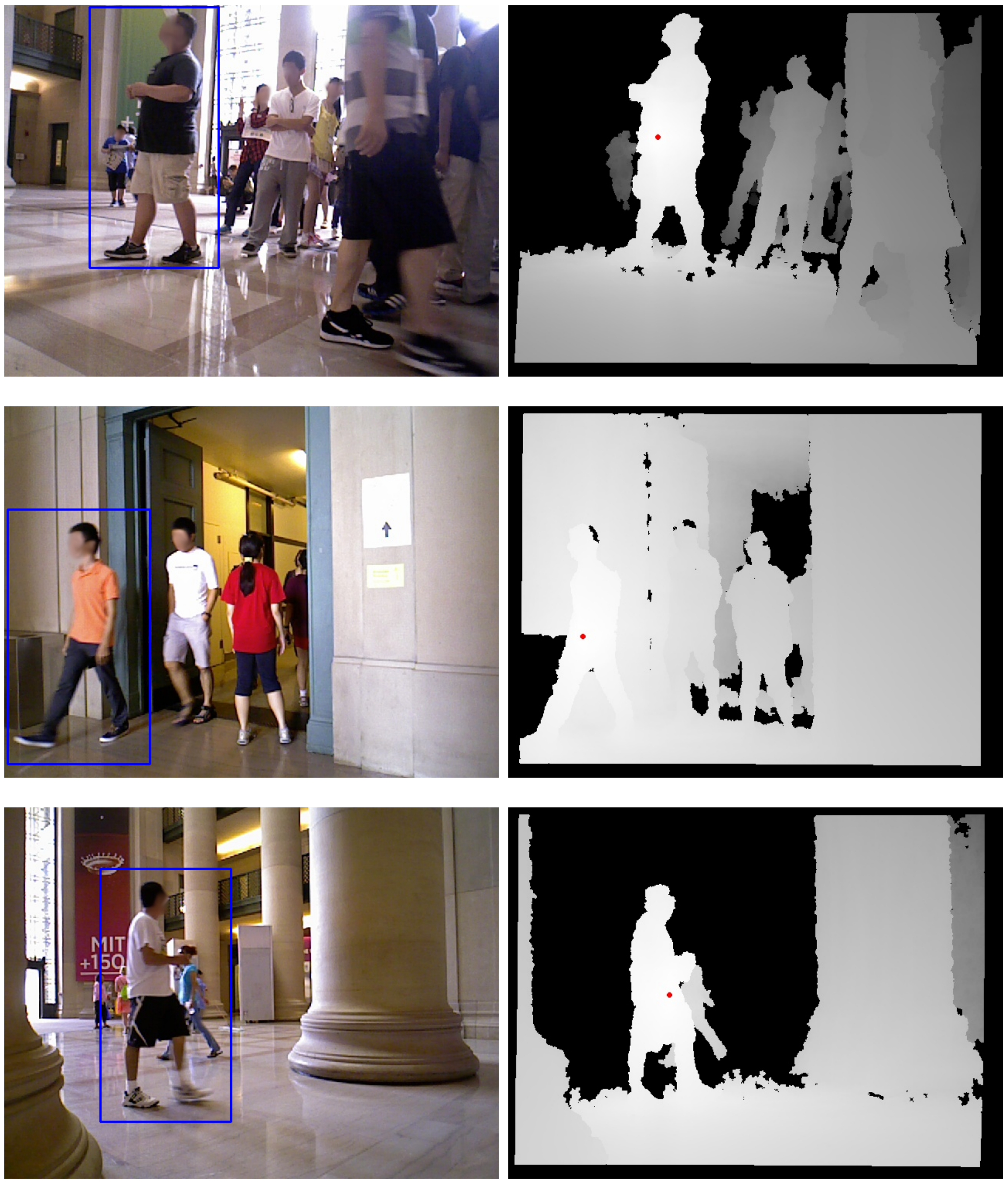


Figure 5.9: Geodesic distance calculation. (a) displays the color image with overlayed bounding box of the object. (b) is the colored geodesic distance map of the given image. Red point corresponds to the middle point of the bounding box. The points which have darker intensity in the map are far from the middle point and lighter points are close to it. Same intensity in two different images correspond to the same euclidean distance.

The proposed point-wise human shape descriptor, f_s, is the combination of the normal, η_i, vectorial distance, Δ_v, and geodesic distance, GD_i of a point p_i. Then, fs becomes:

$$f_s = [\eta_x \; \eta_y \; \eta_z \; \Delta_x \; \Delta_y \; \Delta_z \; GD_i]^T \tag{5.9}$$

5.2.2.2 Training the Classifier

Human refinement task can be considered as 2-label classification problem in the machine learning perspective. Simply, the label of the first class refers to the human, and the other label is for the non-human points, called as background. In order to train, the point-wise human classifier, *H-Classifier*, positive human descriptors, f_s^+, and negative human descriptors, f_s^-, are needed. The samples of f_s^- might be chosen from the non human body points of the scene. f_s^- are computed as same as the positive samples as explained in the previous section.

Randomized Decision Forests is chosen to train *H-Classifier*. It is one of the state of art, fast and effective classifier [145] [22] [3] [163]. It is suitable and applicable for a wide range of different tasks and problems [123] [101] [167]. A Decision Forest consists of some number, T, of decision trees. It is called as randomized because of training each of the decision tree by randomly selected a user determined number of training samples. Each tree in the decision forest includes split and leaf nodes. Each split node consists of an axes, $f_s(x)$, of f_s, and a threshold τ. To classify a given descriptor of a point, f_s, the split nodes of the decision tree are evaluated by starting from the root of the tree. Whenever a leaf node is hit in a tree, t, a decision distribution, $P_t(d|f_s)$, is obtained.

In the case of human refinement problem, $P_t(d|f_s)$ can be considered as 2-bin histogram whose bins refers the label of human or background. The result label of randomized decision forest classifier can be average of all distributions given by the trees in the forest:

$$P(d|f_s) = \frac{1}{T}\sum_{t=1}^{T} P_t(d|f_s) \tag{5.10}$$

Or the result label can be the maximum number of voted label by each decision three, t, in the forest as formulated in the following equations:

$$L(P_t(d|f_s)) = \begin{cases} 1 & \text{if } \; P_t(d_H|f_s) \geq P_t(d_B|f_s) \\ -1 & \text{otherwise} \end{cases} . \tag{5.11}$$

where $L(x)$ is the decision label function of a given decision distribution of a tree, t. d_H and d_B are the bin values of human and background labels in the distribution. The normalized confidence score of a point belonging to the human region becomes:

$$\mathcal{C}_i = 0.5 + \frac{1}{T}\sum_{t=1}^{T} L(P_t(d|f_s)) \tag{5.12}$$

Then, the final decision label of the forest, $L(P(d|f_s))$ becomes:

$$L(P(d|f_s)) = \begin{cases} 1 & \text{if } \; \mathcal{C}_i \geq 0.5 \\ -1 & \text{otherwise} \end{cases} . \tag{5.13}$$

Each tree is trained on a different set of randomly selected positive and negative samples using the following algorithm [101]:

1) Randomly obtain a set of splitting candidates for a tree node, $\Phi = (f_s(x), \tau)$. $f_s(x)$ is an axes of point-wise human shape descriptor, and τ is the split threshold.

2) The set of training points, $S = \{p_i\}$, are divided into two sets, S_l and S_r, for left and right leaf of the node by each Φ:

$$S_l(\Phi) = \{p_i \mid f_s(x) \leq \tau\} \tag{5.14}$$

$$S_r(\Phi) = S \; - \; S_l(\Phi) \tag{5.15}$$

3) Find the best splitting candidate, Φ^*, which produces the largest information gain:

$$\Phi^* = \underset{\Phi}{\operatorname{argmax}}\, G(\Phi) \tag{5.16}$$

$$G(\Phi) = H(S) - \sum_{\psi \in (l,\, r)} \frac{|S_w(\Phi)|}{|S|} H(S_w(\Phi)) \tag{5.17}$$

where $H(S)$ is the Shannon Entropy. It is computed on the normalized distribution of the labels of the points in the set of S as in the following equation:

$$H(S) = -\sum_{i=1}^{n} Pr(l_i|P_L)\; log_2 Pr(l_i|P_L) \tag{5.18}$$

where P_L is the label distribution in the set S, and l_i is the label name.

4) If the current depth of tree is under a maximum threshold, the create left and right children of the current node by using left and right subsets, $S_l(\Phi^*)$ and $S_r(\Phi^*)$.

5.2.2.3 High Level Observations and Color Discontinuity

Graph cut [21] [16] [17] [15] provides a powerful framework to produce globally optimal object segmentation results. Its graph structure enables to combine multiple different kinds of features in one joint framework. In our approach to the refinement problem, graph cut is chosen as the infrastructure to incorporate the cues for a joint final solution.

Point-wise human shape descriptor, f_s, described in the previous section does not include the color or intensity cues of the human body. There are several reasons not to include this information into descriptor. First of all, it is so difficult to learn a generalized the color models of the human skin and the cloths which humans wear. Also, different illumination conditions in the scene, the variance of possible backgrounds, and various fashion style of the people make complicated the construction of a robust human and background color models. Therefore, adding any color related cue into the human descriptor is avoided. However, the human refinement procedure can utilize from the discontinuity of the color between the points in the scene. This can be achieved by employing the color discontinuity in the graph cut framework.

The idea of putting high level observations into the graph cut was first introduced in [135]. It is a point-wise graph cut based object tracker algorithm. In this work, it is assumed that the object pixels at time t should stay connected in a different location at time $t+1$. This could be done by forcing graph cut to put those group of pixels as much as possible into the same label category.

The high level observations determined by some pre-processing steps can be employed as a different layer in the graph. Some of the high level observations which belong to the background can be the estimation of the ground plane, the walls, and the objects which are previously known that they are not a part of the human body. Indeed, there is an important disadvantage of labeling the points which belong to these high level observations as hard background in the graph cut. If the high level observations are retrieved by some estimation process, they might include some actual foreground points. For example, some points which are actually the foot points on the human body are estimated as the ground plane points as can be seen in Figure 5.7. In the graph cut, it is desirable to state two attributes of these points. First, they all together define an observation. Second, yet some of the points within this observation might be misclassified by the estimator and those are subject to change their labels.

Standard graph cut technique [15], contains only single layer graph structure in which only the point level observations can be employed, such as the intensity, color, disparity, and etc. Single layer structure can be extended to the multi level architecture to include high level observations, such as estimated ground plane in the human refinement problem. In order to incorporate the high level observations, a second layer of the nodes are added in addition to the standard first layer of the nodes. Each node in the second layer represents a high level observation defined by a group of the points in the first layer. In this case, a node is expected to be added to the second layer to represent the estimated ground plane points. The interaction between first and second layers are established in a way that a second layer node is connected to the some of the nodes in the first level which all together define that high level observation.

Multi-layer graph cut allow to combine low and high level observation in a joint way. The confidence score of the classifier which is trained using point-wise human shape descriptors, f_s, and the color discontinuity between neighbor points are employed as the low level observations into the multi-layer graph. The estimation of the ground plane is incorporated as a high level observation and employed in the second layer of the graph structure.

Multi-Layer Graph: The multi-layer undirected graph, $G_{Multi} = (V, \mathcal{E})$, is defined by the set of the nodes, V, and the set of the edges, $\mathcal{E}$. The set of nodes consists of two subsets. The first subset of the nodes, V_L, are the first layer nodes which represent the low level observations. Each point in the scene is defined by a node, n_i, where $n_i \in V_L$. The second subset of the nodes, V_H, represent the high level observations employed in the second layer of the graph G. In this case, V_H consists of a single node, n_H, which is for the estimated ground plane. Thus, the set of nodes, V, becomes $V = \{n_1, \dots, n_k\} \cup \{n_H\}$, where k is the number of the points in given scene.

The set of edges, $\mathcal{E}$, consists of two types of the edges. The low level interactions between the points in the first layer are formed by the set of the edges, $\mathcal{E}_L$. $\mathcal{E}_L$ consists of the edges, $e_{i,j}$, between two neighbor nodes, n_i and n_j, in V_L. The connections between the low and high level observations are established by the edges, $\mathcal{E}_H$. Each node, $n_i \in V_L$, in the low level, is connected to the node, $n_H \in V_H$, in the second level by the edges, $e_{i,H}$. Thus, $\mathcal{E}$ becomes $\mathcal{E} = \{e_{1,2}, \dots, e_{i,j}, \dots, e_{k-1,k}\} \cup \{e_{1,H}, \dots, e_{k,H}\}$.

The segmentation of the multi-layer graph, $G_{Multi} = (V, \mathcal{E})$, is assigning a label l_i, from a set of labels, $\{l_H, l_B\}$, to each node, n_i, in V. In this case, l_B refers to the background, and l_H refers to the human point. The set of all labels assigned to nodes in V is denoted as $\widetilde{L}$. The final mask of the human is formed by the points whose labels are assigned to l_H by graph cut.

Graph Cut Energy Functions: The energy function of the multi-layer graph cut consists of two terms, namely Regional term, R, and the boundary term, B, as in the standard graph cut energy equation:

$$E(\widetilde{L}) = \sum_{i \in V} R(l_i) + \sum_{\{i,j\} \in V} B_{i,j}(l_i, l_j) \tag{5.19}$$

where i and j are the nodes of any edge, $e_{i,j}$, in $\mathcal{E}$.

Regional term of the multi-layer graph cut employs the confidence score of the $H-Classifier$, $\mathcal{C}_i$, for each point as defined in the Equation 5.12. Also, The high level observation which is the ground plane estimation in this case is incorporated into the regional term. More precisely, the regional term of the multi-layer graph cut energy function becomes:

$$\sum_{i \in V} R(l_i) = \alpha_1 \sum_{i \in V_L} -\ln(p_c(i, l_i)) + \alpha_2(\mathcal{H}_L(n_H, l_H)) \tag{5.20}$$

The first term in the above equation describes the confidence score of $H-Classifier$. Second term defines the feature of the high level observation in the system. α_1 and α_2 is the relative influence between two terms.

The shape likelihood of a point, p_c, is formulated using the confidence score, $\mathcal{C}_i$, produced by $H-Classifier$ as following:

$$p_c(i, l_i) = \begin{cases} \mathcal{C}_i & \text{if } l = \text{"}foreground\text{"} \\ 1 - \mathcal{C}_i & \text{if } l = \text{"}background\text{"} \end{cases} . \tag{5.21}$$

H_L defines the similarity function of of the observation for the high level observation node and its given label, L_H. Since, the estimated ground plane is considered as the background and there is no prior information available to measure the confidence of the ground estimation process, $\mathcal{H}_L(n_H, l_H)$ is set to 1 if l_H is *background*. Simple, it is set to "0", if l_H is "foreground".

The color discontinuity and the interactions between low and high level observations are defined in the boundary term of the energy function, $E(\widetilde{L})$. As in [14], the color discontinuity between neighbor points in the first layer of the graph can be formed as the following:

$$B-Color_{i,j \in V_L} = \lambda_1 \frac{1}{dist(i,j)} e^{-(||c_i - c_j||^2)/2\sigma^2} \tag{5.22}$$

where c_i and c_j are the color of the point i and j, $dist(i, j)$ is the standard L_2 Euclidean norm yielding point distance, σ^2 is the average squared norm in the image.

The graph edges between one first layer node and high level observation node depends on the distance between the normal of a point, $p_n = (n_x, n_y, n_z)$, in the estimated ground plane and the estimated normal of the ground plane, $\overline{n_H}$. It can be formed as:

$$B - High_{i \in V_L, j \in V_H} = \lambda_2 e^{-(||p_n - \overline{n_H}||^2)/2\sigma_H{}^2} \tag{5.23}$$

where σ_H is the averaged squared distance between the normal of each ground plane point, p_n, and the estimated normal of the ground plane, $\overline{n_H}$. λ_1 and λ_2 are to weight these two boundary terms.

The final labeling, $\widetilde{L_F}$, can be achieved by minimizing the energy function in Equation 5.19 by the graph cut algorithm explained in [15]:

$$\widetilde{L_F} = \underset{\widetilde{L}}{\operatorname{argmin}}\, E(\widetilde{L}) \tag{5.24}$$

The proposed refinement process which utilizes from the multi-layer graph cuts and $H - Classifier$ is called as $H - Classifier_{Multi-GC}$ after this point.

5.3 Experimental Results

Several experiments were conducted to quantify to analyze the performance of the proposed algorithms. In order to use in the experiments, a subset dataset of *DontHitMe*, called as *DontHitMe-Refine*, which includes the manually annotated ground truths of 103 images are used. In order to diversify this dataset, it is ensured that the same person does not appear in multiple images of *DontHitMe-Refine*. Also, as possible as background variation tried to be included into the dataset in the creation of *DontHitMe-Refine*. Ground truths are generated so carefully with the help of an software which is implemented just for this purpose. The software allows the user to label the human body pixels in detail.

Two different set of experiments are performed. The first set of the experiments aim to measure the performance of the methods which do not utilize from the prior information, but only uses generic cues, described in Section 5.2.1. The second set of the experiments analyze the performance of the point-wise classifiers explained in Section 5.2.2.

5.3.1 Refinement using generic cues and no prior information

Different tests which incorporate different combination of the generic cues were performed to see the outcome of *GC-Refine* explained in Section 5.2.1.1. Seven different combinations of three generic cues which are the color, depth and normal are used in this experiment. *GC-Refine* is tested with each of the combination of the cues separately. All of the images of *DontHitMe-Refine* dataset are refined in these tests. The same formula in Equation 4.2 is used to measure the overlap between the results of *GC-Refine* and the ground-truths. The median overlap scores of these experiments can be seen in Table 5.1. The highest performance is achieved by incorporating only the depth of the points into the feature vector of *GC-Refine* whose median overlapping score is 0.55. The combination of the depth and normal cues produces second best results. The test case in which only the normal of the points are used in *GC-Refine* gives the third best performance.

Figure 5.10 displays the results of this experiment by using the same test image for the different combination of the cues. Only the best three results of the tests are illustrated. Figure 5.10 (a), (b) and (c) shows the outputs of the tests in which only depth, the combination of the depth and normal, and only the normal cues are used, respectively. It can been seen that a simple scaling of the bounding box, *B(x,y,w,h)*, to obtain the human model, M_H, is not sufficient. In this case, M_H contains some points which are originally background points and belong to the human far behind and left of the target model. Hence, all those points are segmented as the human by three tests regardless of the combined cues. If it is looked at the foot level of the human in Figure 5.10 (a), it can be noticed that some ground points are included to the refined region. This is a natural outcome of chosen the depth as the only cue in the process. Those ground points have the similar depth values as the human model, M_H, so they are added into the output mask. Combining the normal and depth of the points did not help to improve the results, but it caused to exclude some points at the upper body level. It is mainly because the normal of those part of the body mostly exist in the background.

A separate set of tests were performed to see the effects of adding distance penalty as another cue in addition to the combination of the cues in the previous experiment. The median overlapping scores of this experiment can be seen in Table 5.2. These top three results are achieved by using only depth, the combination of the depth and normal, and lastly by combining all cues which are rgb color, depth and the normal. The distance cue helps to improve the results for each tests. For example, the best overlapping score which is obtained by only depth cue in the previous experiment increase to 0.61 from 0.57 after utilizing from the distance. The best three results of different possible cue combinations are displayed for the same image in Figure 5.11. The results in this figure are shown according to descending order of the overlapping scores of the cue combinations. An obvious effect of incorporating the distance cue can be seen in 5.11 (a) where only depth data is used in the refinement. It removes the ground points at the foot level which are assigned to large distance penalty from the human region and yields better result. However, a drawback of it can be observed in Figure 5.11 (b) and (c). Some of the points on the head of the human eliminated from the segmented region. This is due to far relative distance of these points to the middle of the body.

In order to analyze the performance of extracting the non-region of interest from the object model, M_H, *GC-Refine-Pre* refinement method is experimented in the same way. As in the previous tests of *GC-Refine*, all different combinations of the cues are analyzed. As it can be foreseen, provided more clean models yield better refinement results. The median overlapping scores of this experiment are summarized in Table 5.3. The best score is achieved if only depth data is used. Its score goes up to 0.66 from 0.61 which was achieved by *GC-Refine* with distance cue in Table 5.2. The results of best three combinations of the cues are shown for the same image in Figure 5.12.

Used Cues	*Median Overlap Score*
Only RGB	0.36
Only Depth	0.55
Only Normal	0.46
RGB+Depth	0.37
RGB+Normal	0.40
Depth+Normal	0.52
RGB+Depth+Normal	0.43

Table 5.1: Median overlap scores of *GC-Refine* method for *DontHitMe-Refine* data set. Each row specifies an experiment in which different combinations of the cues used in the refinement process by *GC-Refine* approach. For example, RGB+Depth means that the combination of RGB color and depth image data are incorporated into *GC-Refine* for that experiment.

The outcome of only depth is displayed in 5.11 (a), the combination of the depth and normal in 5.11 (b), and the combination of rgb color, depth and normal in 5.11 (c). The main improvement in the results of *GC-Refine-Pre* is that they have much less or no background points. Especially, the points of the human who is on the left far side is removed from the final refinement. Those background points were problematic for *GC-Refine* as can be seen in Figure 5.10 and 5.11.

A result gallery of the methods can be seen in Figure 5.15.

5.3.2 Point-wise Classifier

Several experiments were conducted to measure the performance of point-wise classifier, $H - Classifier$, and its extended version, $H - Classifier_{Multi-GC}$, explained in Section 5.2.2. In order to train the classifiers in each of the experiments of this section, *DontHitMe-Refine* is divided into two subsets, *DontHitMe-Refine-Train* and *DontHitMe-Refine-Test*. No same human appears in both of these two datasets. *DontHitMe-Refine-Train* contains 51 images and its ground truths were used the train the classifiers. *DontHitMe-Refine-Test* includes 52 images and used for the testing purpose.

Cues (In addition to Distance Map)	*Median Overlap Score*
Only RGB	0.43
Only Depth	0.61
Only Normal	0.47
RGB+Depth	0.48
RGB+Normal	0.45
Depth+Normal	0.57
RGB+Depth+Normal	0.51

Table 5.2: Median overlap scores of *GC-Refine* method for *DontHitMe-Refine* data set. In this case, in addition to other cues, distance map is incorporated. As it is defined in Table 5.1, each row specifies different combinations of the cues used in the refinement process by *GC-Refine* approach.

Method	*Median Overlap Score*
Only RGB	0.48
Only Depth	0.66
Only Normal	0.52
RGB+Depth	0.51
RGB+Normal	0.53
Depth+Normal	0.61
RGB+Depth+Normal	0.58

Table 5.3: Median overlap scores of *GC-Refine-Pre* method for *DontHitMe-Refine* data set. As it is defined in Table 5.1, each row specifies different combinations of the cues used in the refinement process by *GC-Refine-Pre* approach. The distance map cue is added by default into all experiments.

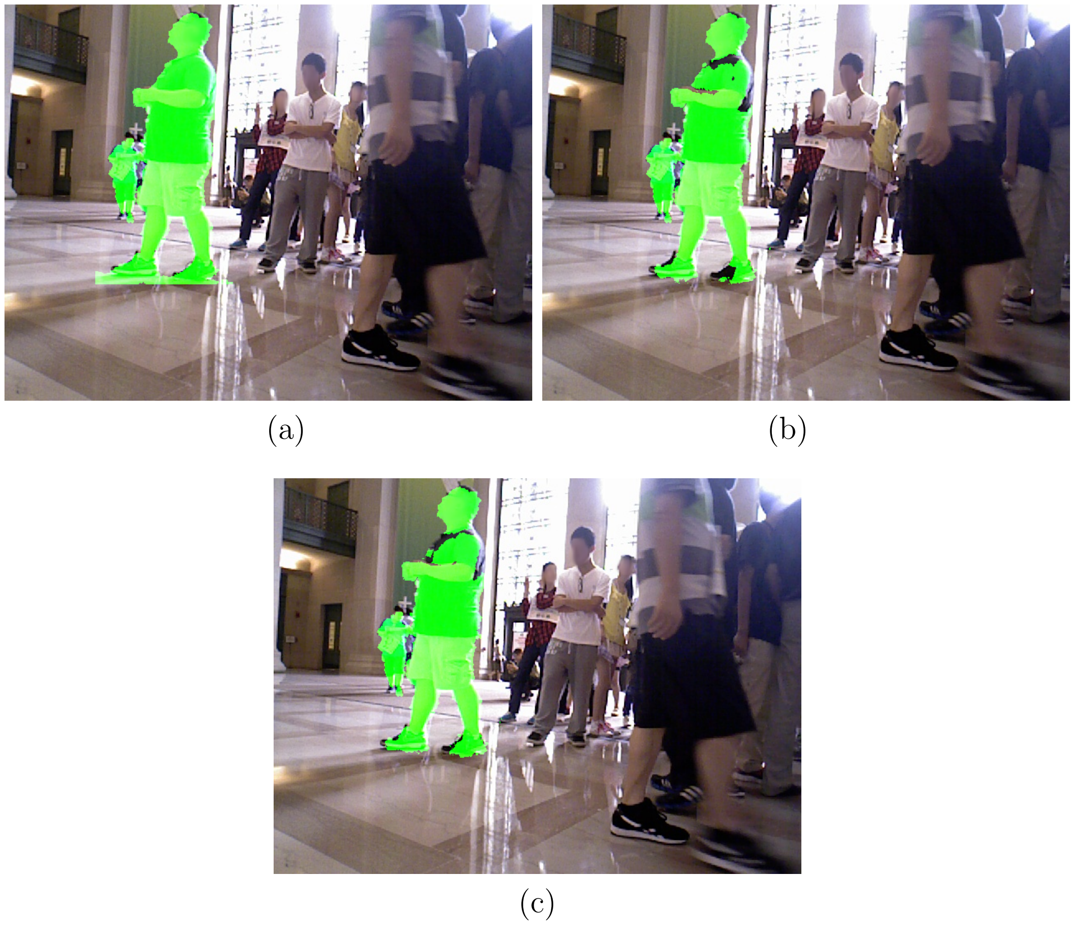


Figure 5.10: The results of *GC-Refine* for different combinations of the cues in a given same input image . (a) shows the refinement result, if only the depth data is used, (b) displays the result of combining the normal and the depth of the points, (c) shows the refinement produced by using only the normal cue. It can be realized that these combinations of the cues produced best three overlapping scores in Table 5.1.

The positive samples are obtained from the ground truth of the human masks in *DontHitMe-Refine-Train* dataset to train the classifiers. All the points inside a ground truth mask are used to generate the positive samples. However, negative samples are not computed for all points which stay outside of the ground truth. Only one of every two pixels in a row of the image is added to the set of the negative samples. In this way, the training time of the classifiers is aimed to be reduced. Also, no points which

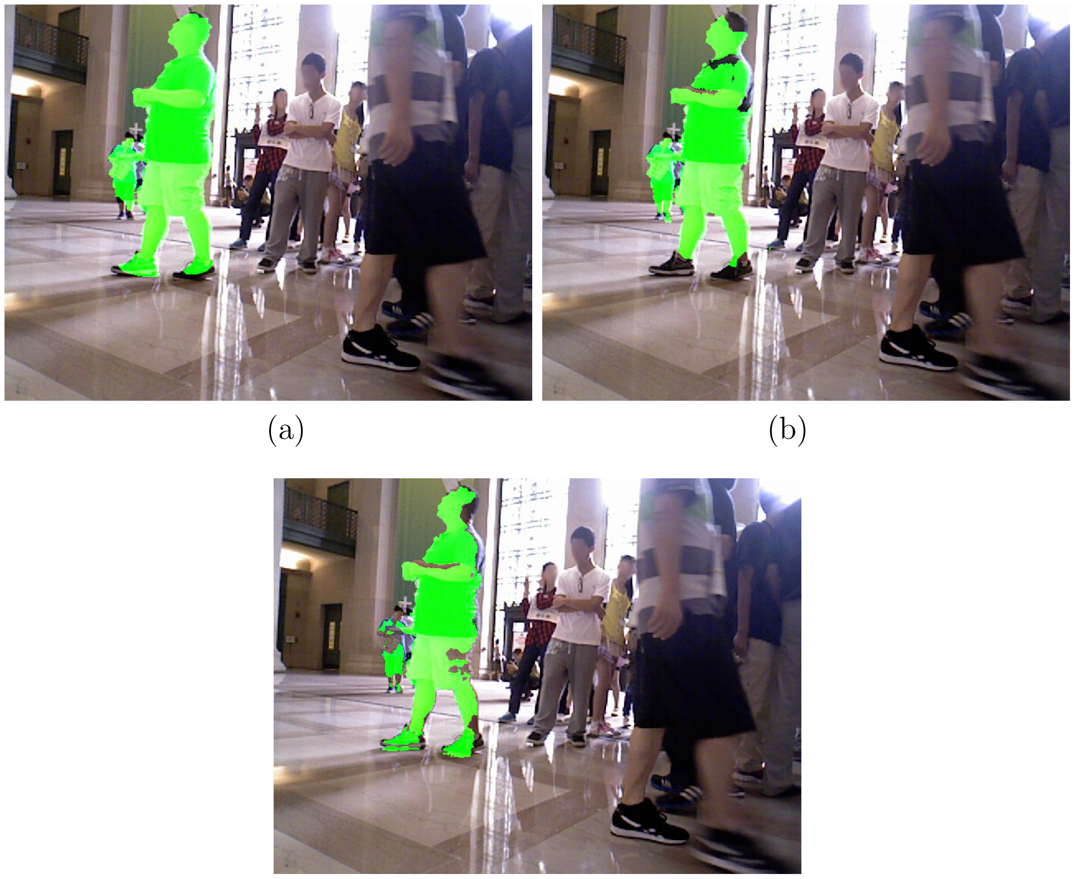

(a) (b) (c)

Figure 5.11: The results of *GC-Refine* for a given same input image. In this case, the distance cue is incorporated into all different combinations of the cues by default. (a) shows the refinement result, if only the depth data is used, (b) displays the result of combining the normal and the depth of the points, (c) shows the refinement produced by combining RGB color, the normal, and the depth of the points. It can be realized that these combinations of the cues produced best three overlapping scores in Table 5.2.

do not have valid depth data are not included in the training set. 5 decision trees were trained for each of the classifiers.

The first set of tests were performed to analyze the performance of $H{-}Classifier$ when it is trained with the point-wise shape descriptors, f_s, which includes different combination of the cues. The normal, η, vectorial spatial distance, Δ_v, and the geodesic

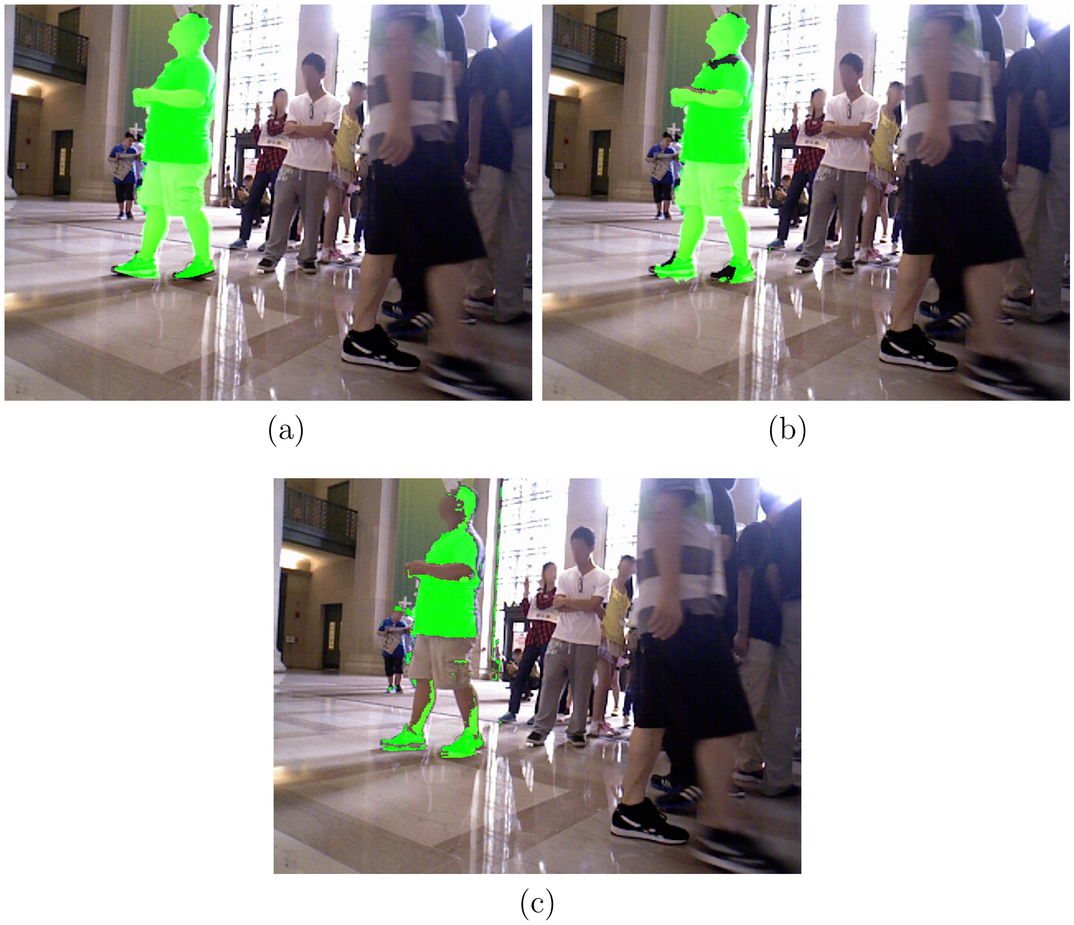


Figure 5.12: The results of *GC-Refine-Pre* for a given same input image. (a) shows the refinement result, if only the depth data is used, (b) displays the result of combining the normal and the depth of the points, (c) shows the refinement produced by combining RGB color, the normal, and the depth of the points. It can be realized that these combinations of the cues produced best three overlapping scores in Table 5.3.

distance, GD of the points are combined in f_s in 7 different possible way. A different $H-Classifier$ is trained for each of these combinations using the same *DontHitMe-Refine-Train* dataset. Then, the points in the images of *DontHitMe-Refine-Test* are labeled as the human or background by these classifiers. Median overlapping scores are computed for each tests. They can be seen in Table 5.4. The best performance, the median overlapping score is 0.79, is achieved when all three of the cues are included

Method	*Median Overlap Score*
Only Δ_v	0.75
Only η	0.58
Only GD	0.78
$\Delta_v + \eta$	0.785
$\Delta_v + GD$	0.77
$\eta + GD$	0.78
$\Delta_v + \eta + GD$	0.79
$\Delta_v + \eta + GD$ + RGB	0.78
$\Delta_v + \eta + GD$ + LAB	0.78

Table 5.4: Median Overlap scores of $H - Classifier$ for different cue combinations in its descriptor, f_s. For example, $\Delta_v + \eta$ means the vectorial spatial distance and the normal of the points are used in the descriptor, f_s.

in f_s. In the case of missing one of the cues in f_s, the scores dropped down. Also, the classifier which utilizes from the normals, η, was unable to distinguish between background and human points. Thus, the normal, η, is not capable to represent the human body points alone. However, when it is associated with the vectorial spatial distance, Δ_v, they both performed well by hitting the score of 0.785.

In addition to three cues related to the shape, 2 more tests were conducted to see the effect of incorporating the color of the points into the point-wise human descriptor, f_s. Simply, three channels of RGB color of the points are included in f_s as three additional dimensions. Same test is also performed for LAB color space. In this case, LAB color of the points are put in f_s. The overlapping scores of these two classifiers which include the color information are shown in the last two rows of Table 5.4. As it is expected, adding the color reduced the performance of the classifier from 0.79 to 0.78. This is mainly because of the color variation of the human skin, clothes, background and different illumination conditions. No difference between different color spaces is observed. They both dropped the overlapping scores in the same amount.

The best four classifiers are picked up to run on the same image. The results of these four classifiers for a given image are illustrated in Figure 5.13. Figure 5.13

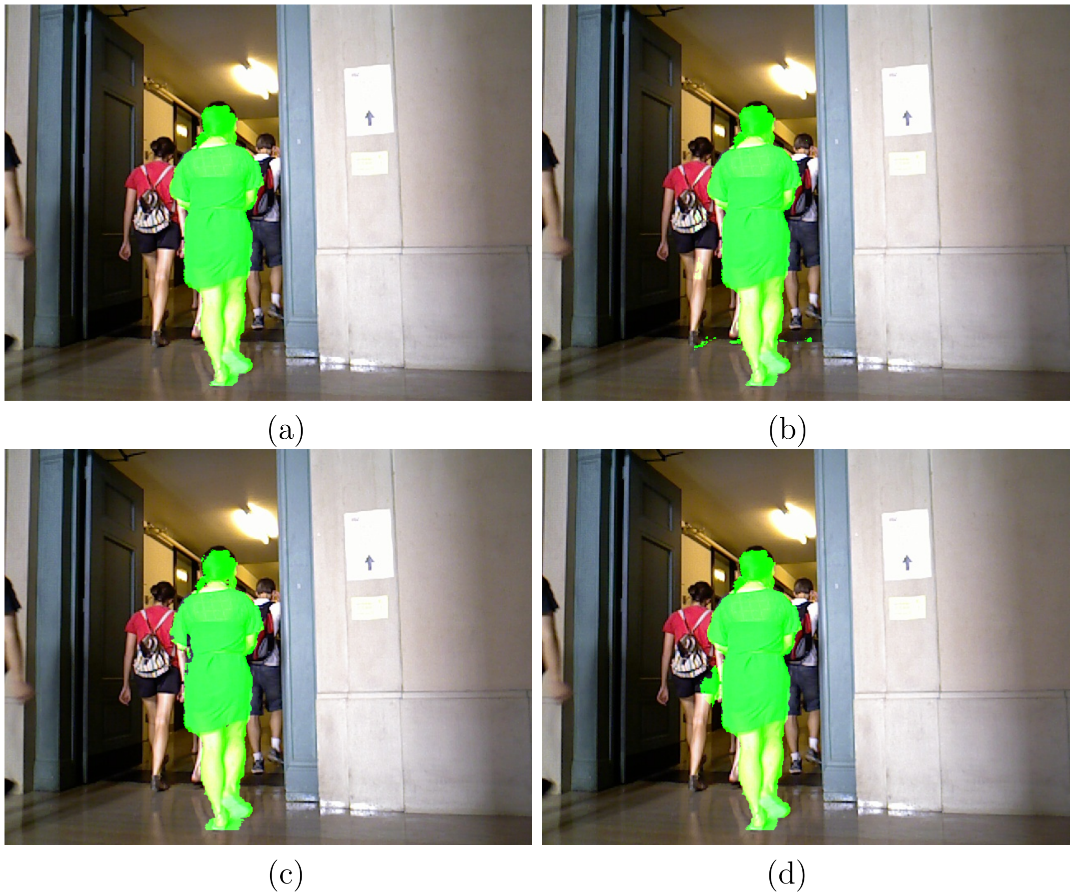


Figure 5.13: The results of $H - Classifier$ for a given same input image to classify its points. (a) shows the classification result, if all three shape cues, Δ_v, η, and GD are incorporated in the descriptor f_s, (b) displays the result of combining all three shape cues and RGB color in f_s, (c) shows the refinement produced by combining Δ_v and η, (d) displays the result of η and GD in f_s. It can be realized that these combinations of the cues produced best four overlapping scores in Table 5.4.

(a) shows the result of the test in which all three shape cues are used, (b) displays the combination of the shape cues and RGB color, (c) the combination of the vectorial spatial distance, Δ_v, and the normal, η, (d) the combination of the geodesic distance, GD, and the normal η. As it can be seen in (b) in which the color is used an extra cue, some points in the background are classified as the human. It is easy to guess that the classifier learned the similar color models that these points have, so they are labeled as the human. Not utilizing from the geodesic distance caused mislabeling of the points at the head level of the image (c). Since the normal computation might be erroneous at the points where the depth information is not so accurate at the sharp edges, in the hair of the human, those points might be labeled as the background even they stay inside the human body. It is possible to some of those points in (c). The negative effect of not using the vectorial spatial distance, Δ_v, can be observed in Figure 5.13 (d). In this image, some of the points which belong to the human on the left side are put into the classified region. Their spatial distance in z direction, Δ_z is expected to be large and labeled as background. Since this information is not included into f_s, this result becomes usual.

Also, the images of *DontHitMe-Refine-Test* is refined by $H - Classifier_{Multi-GC}$ method which is explained in Section 5.2.2.3. The confidence scores of $H - Classifier$ which produced the best median overlapping score in Table 5.4 by combining three shape related cues, Δ_v, η, and GD, in f_s are used in the multi-layer graph cut framework. This process can be considered as second stage in the refinement process. The median overlapping scores of $H - Classifier_{Multi-GC}$ is listed in Table 5.5. A remarkable improvement was achieved by this method. Incorporating the estimated ground plane as a high level observation, utilizing from the shape confidence score of $H - Classifier$, and employing the color information jointly in a multi-layer took the median overlapping score from 0.79 to 0.92. Figure 5.14 displays some examples refined by $H - Classifier$ and $H - Classifier_{Multi-GC}$. (a) and (c) of this figure shows the results of $H - Classifier$. (b) and (d) displays the refinements results of $H - Classifier_{Multi-GC}$ for the same images. In both of the results, proposed multi-layer approach helped to remove some ground points and also background points classified as the human by $H - Classifier$. The color discontinuity term in the graph cut provided to segment the points where there is no depth data is available. For example, some of the hair points on the left side of the neck of the woman in Figure Figure 5.14 (a) do not have valid depth information. Hence, $H - Classifier$ was unable to classify hose points. Yet, $H - Classifier_{Multi-GC}$ included them into the final refined region

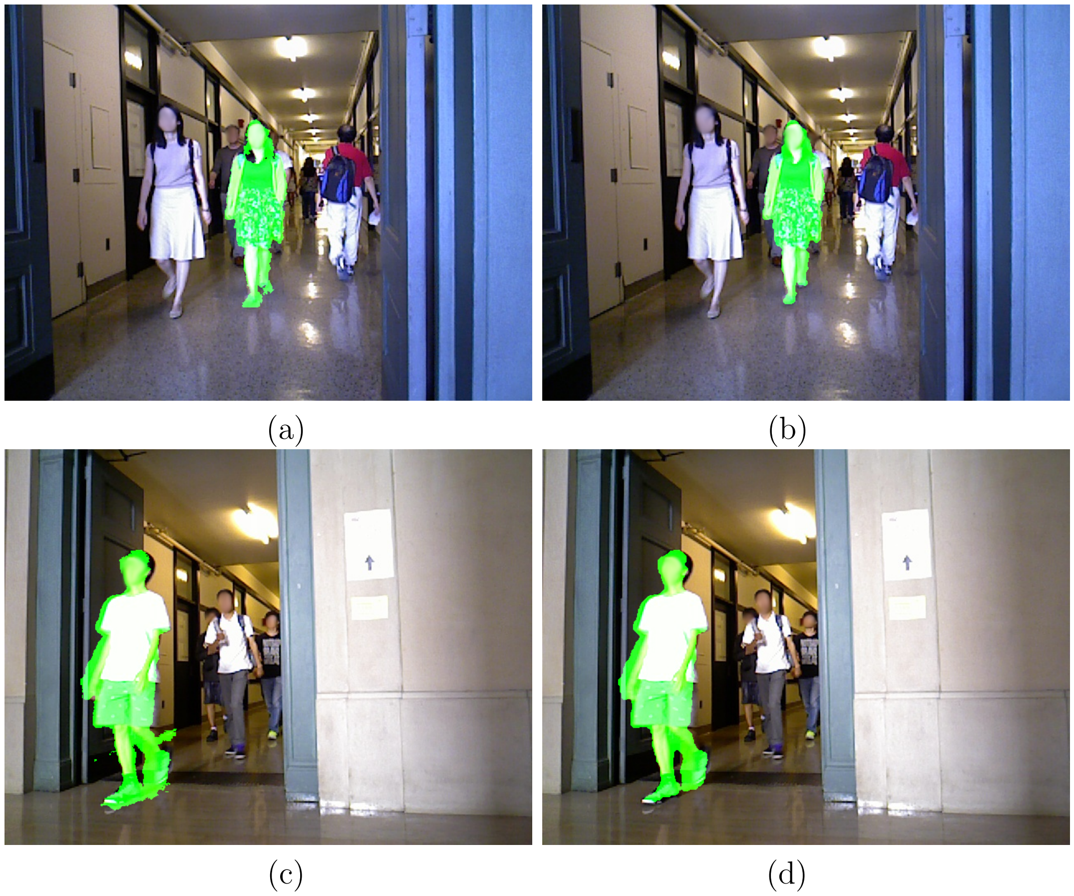


Figure 5.14: Comparison between the results of $H - Classifier$ and $H - Classifier_{Multi-GC}$. Left column images, (a) and (c), display classification results of $H - Classifier$. Right column images show the refinement outputs of $H - Classifier_{Multi-GC}$ for the same image set. $H - Classifier_{Multi-GC}$ improves the results by eliminating more background points at the foot level of the human. Also, its color continuity term helped to classify some points correctly on the human body which have no valid depth data, such as in the hair of the woman in (a) and (b).

as shown in Figure 5.14 (b), because of the color similarity between other hair points which have valid depth.

The proposed $H - Classifier$ and $H - Classifier_{Multi-GC}$ are compared to *GrabCut* [157], and the body part labeling method explained in [165] [166] which runs in the Kinect devices. The same set of images *DontHitMe-Refine-Train* and samples

Method Name	*Median Overlap Score*
$H-Classifier_{Multi-GC}$	0.92
H-Classifier	0.79
H-Classifier-SVMs (trained by SVMs)	0.67
GC-Refine-Pre	0.66
GC-Refine	0.61
GrabCut [157]	0.57
Depth difference features from [165] [166]	0.53

Table 5.5: The comparison of the best performed methods in *DontHitMe-Refine* data set. Only the result of cue combinations which performed best are taken into this table for $H-Classifier_{Multi-GC}$, $H-Classifier$, *GC-Refine-Pre*, and *GC-Refine*. For example, *GC-Refine* in this table uses the depth and distance map cues in its refinement process as its best overlapping score can be seen in Table 5.2.

are used to train their classifier. In this case, human body part labeling problem is reduced to a simple human/background problem. It did not perform well in the test dataset, *DontHitMe-Refine-Test*, and got the overlapping score of 0.53 as listed in Table 5.5. Their descriptor was unable to distinguish the human and background points. Also, $H-Classifier$ and $H-Classifier_{Multi-GC}$ outperformed significantly the results of *GrabCut*. Figure 5.15 displays a gallery of the images which shows the results of different methods. Because of the space limitation, for a given image and the bounding box around the human which is going to be refined, only the best result of *GC-Refine-Pre* obtained by using only depth data, $H-Classifier$ which combines all three shape cues, $H-Classifier_{Multi-GC}$, and *Body Part Classifier* [165] [166] are illustrated.

Random Decision Forests vs Support Vector Machines (SVMs): In order to analyze the performance difference between the Random Decision Forests and SVMs, another point-wise human classifier, $H-Classifier-SVMs$, was trained using SVMs [27]. Radial Basis Function (RBF) was used as the kernel function of SVM. The same human shape descriptors, f_s, to build $H-Classifier$, were used to train $H-Classifier-SVMs$. Also, to assign the same weight to the different cues in f_s, all dimensions of f_s are normalized to same range. In this way, same conditions were established to compare the performance difference between them. SVMs performed worse than Random Forests in the experiments. As it can be seen in Table 5.5, the overlapping score of $H - Classifier$ which was trained by Random Forests is 0.79 whereas the performance of $H - Classifier - SVMs$ dropped to 0.67. The worse performance of SVMs is connected to its decision function explained in Equation 2.8. Basically, the distance to the hyperplane which is the decision margin of the space is measured. In this decision function, all the cues within the descriptor, f_s, take the same amount of importance. However, our descriptor includes different kinds of the cues which may require different weights in the training and testing phases of the classifiers. SVMs does not contain this internal mechanism to balance the weights between different cues. Fortunately, Random Decision Forests supplies this good feature. In the training phase, it separates the data according to maximum possible entropy among different labeled sets. Computing the entropy among the sets does not depend on weighting the each dimension of the descriptors, f_s, differently. Random Forests can handle it because of its this character. Therefore, Random Forest based $H - Classifier$ outperformed the SVMs based classifier. Some sample results of $H - Classifier - SVMs$ can be seen in Figure 5.17.

5.4 Conclusion

In this work, we tackle with the problem of refining the low dimensional representation of the human shape which is a bounding box in this case to obtain more accurate point-wise mask of the human which is more beneficial some type of the applications, such as interactive games, human behavior understanding, and the pose estimation. Two types of the methods are proposed.

The first type of the proposed methods utilizes from only the generic cues, such as the color, depth, and the normals of the points. They do not use any kind of the prior information, but they only extract the necessary cues from the given single scene. These methods are built on the graph cut framework which enables to incorporate the multiple cues and point-wise fast segmentation. Namely, *GC-Refine* and its extended version *GC-Refine-Pre* which obtain more accurate models to feed graph-cut by estimating and the ground plane and extracting cleaner region of interest are developed.

A subset dataset of *DontHitMe*, called as *DontHitMe-Refine*, which contains the point-wise ground truths of 103 images is generated. The performance of *GC-Refine* and *GC-Refine-Pre* are measured by refining the images in *DontHitMe-Refine*. Both of these proposed methods performed better than *GrabCut* [157]. Providing cleaner models to *GC-Refine-Pre* improved the performance from 0.61 to 0.66.

In order to incorporate the prior information into the refinement process, a novel point-wise human shape descriptor is developed. A point-wise human classifier, $H-Classifier$, which is trained by the human shape descriptor is proposed. $H-Classifier$ outperformed the methods which uses only generic information, *GC-Refine-Pre*, *GC-Refine*, and *GrabCut*. Also, multi layer graph cut is proposed to combine the low level and high level observations in the refinement process. The confidence scores achieved from $H-Classifier$, the color information of the points, and the ground plane estimation as a high level observation are all employed in the multi layer graph framework for a joint segmentation. This post-processing step allowed better results and improved the overlapping score from 0.79 to 0.92.

Also, the proposed methods are compared to the famous body part labeling method run in the consumer Kinect devices [165]. In this case, their descriptors were trained only for two sets, the human and background. The low performance of this method showed that their method is not really suitable for the refinement of low dimensional human shapes.

As the future work, the performance of the proposed methods can be analyzed for the cases if some part of the human body is occluded. The power of the human shape classifier can be measured for different human poses, such as sitting humans on the chair, and laying humans on the ground. Another dataset can be recorded for this kind of test. Also, the proposed shape descriptor, f_s, can be extended to generic object shape descriptor to refine any kind of object if its low dimensional representation is given. As a start point of this extension, its given low dimensional representation might be added into the descriptor as additional dimensions. In this way, different associations between the low dimensional representation of the object and its shape can be learned by a decision tree type classifiers. It can be observed that this type of association can not be included into the classifier by only normalizing the axes of the descriptor by using the low dimensional representation of the object.

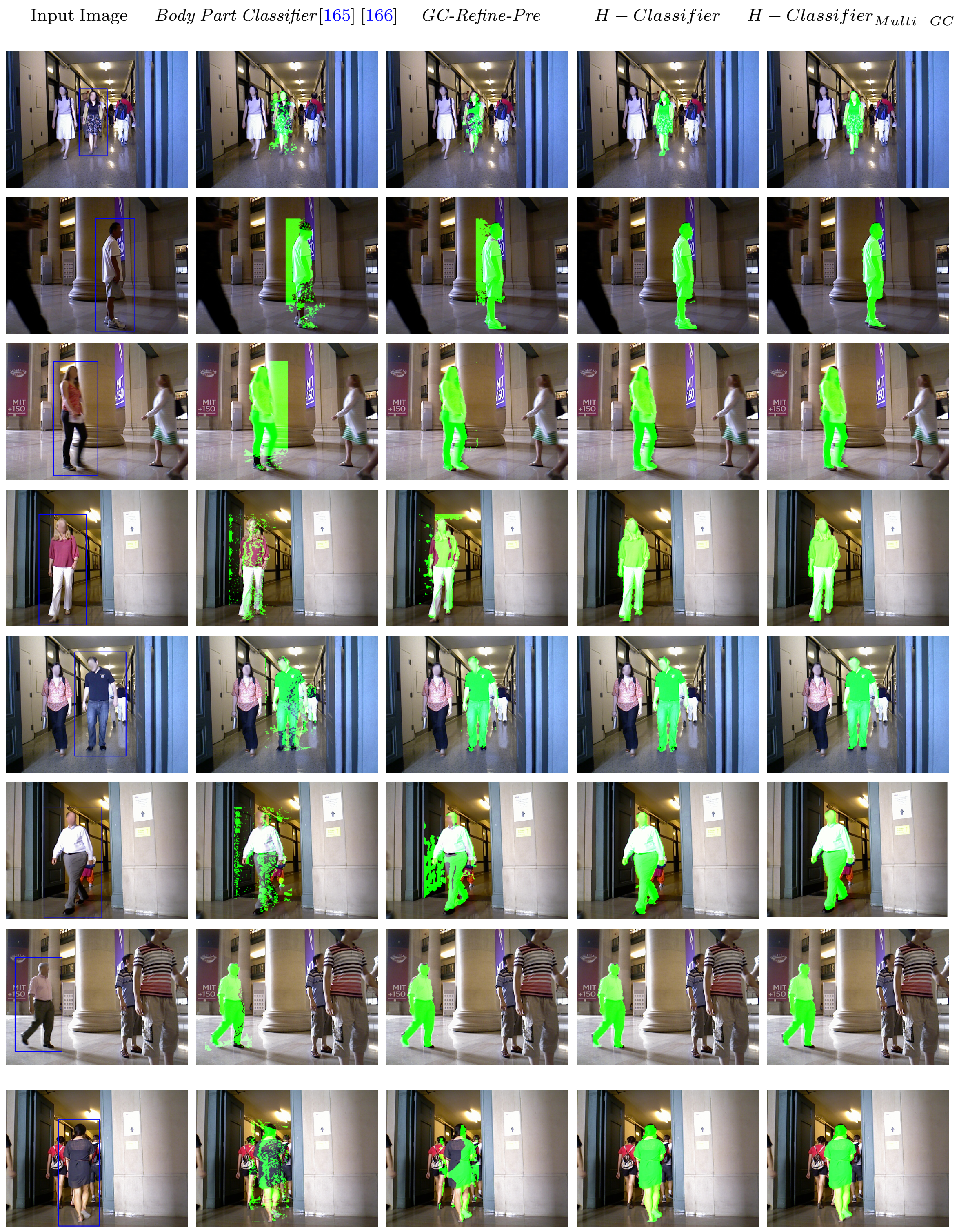

Input Image
Body Part Classifier[165] [166]
GC-Refine-Pre
$H - Classifier$
$H - Classifier_{Multi-GC}$

Figure 5.15: Sample results of different refinement methods. Column headings show the name of the method.Only the results of the cue combinations which produced the best overlapping scores are illustrated for *GC-Refine-Pre* and $H - Classifier$.

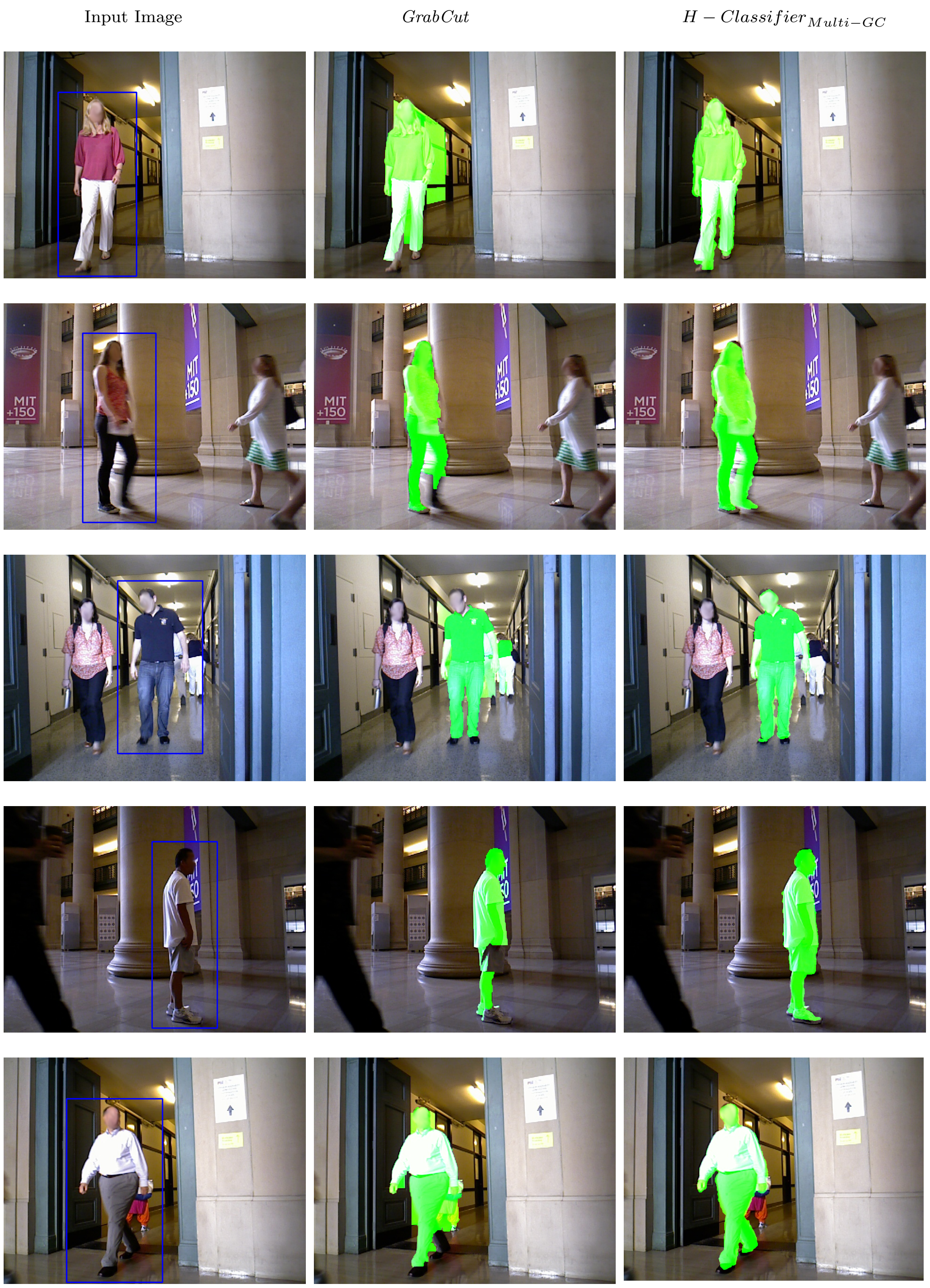


Figure 5.16: Sample results of $GrabCut$ and $H - Classifier_{Multi-GC}$. Column headings show the name of the method.

$H - Classifier$ $H - Classifier - SVMs$

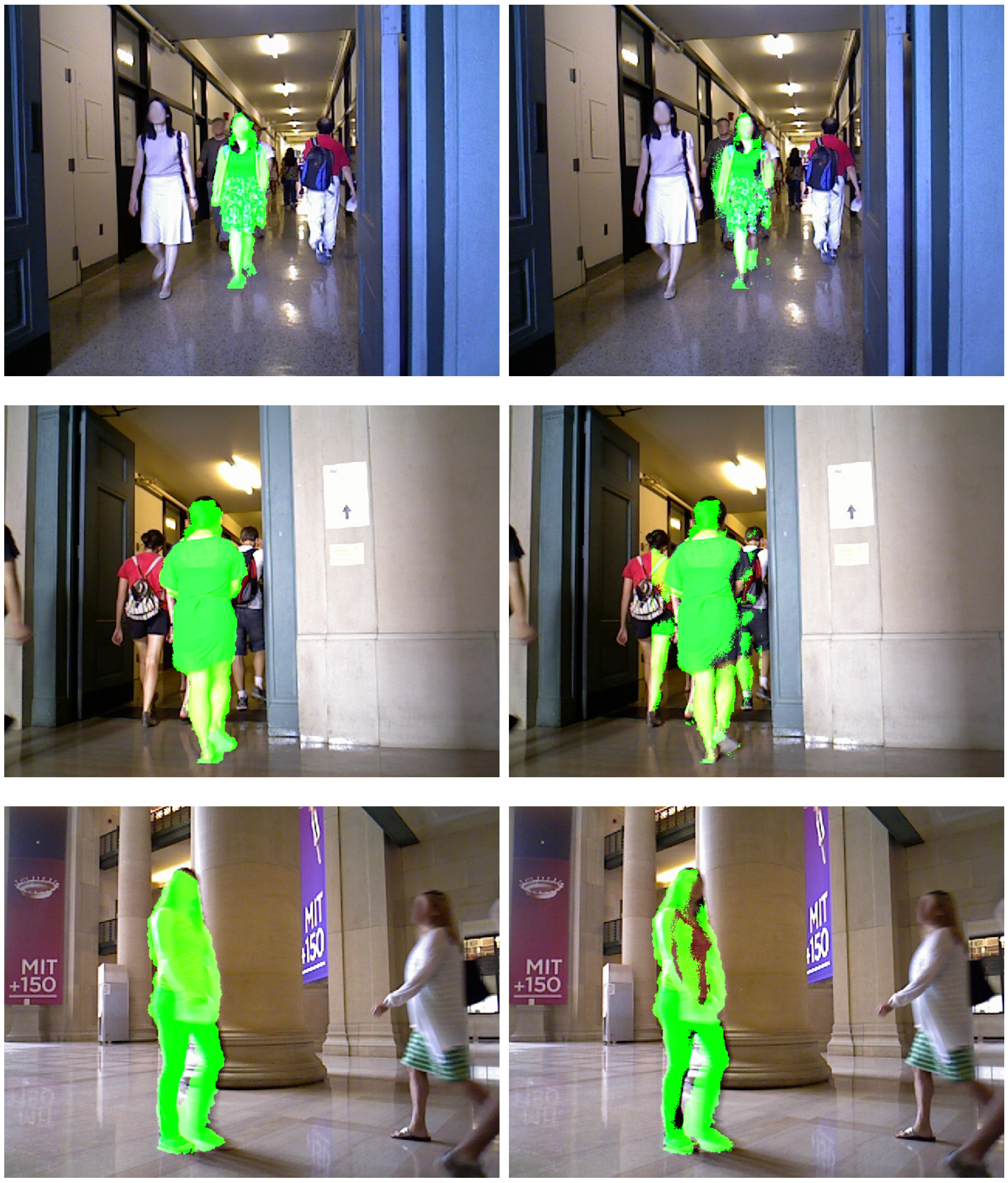


Figure 5.17: Sample results of $H - Classifier - SVMs$ and $H - Classifier$. Column headings show the name of the method.

Chapter 6
POINT-WISE OBJECT TRACKING

By definition, object tracking means to locate the position and orientation of an (or multiple) object over a sequence of sensory information. In other words, an object tracker distinguishes the desired object from its surrounding environment and follows it over time. The tracked object might be someone's face, hand, a vehicle on a road, an aircraft, a fish in the sea, a walking person, a flying bird or anything in the universe which has some features which can be captured by a sensor. The most common used sensors for this purpose are cameras(color and IR), radar and Lidar. A tracking algorithm processes the data coming from one or multiple of the sensors finds object of interest and keeps track of its position, movements and orientation, called as state of the object, in the specific domain.

Object tracking generally is a difficult task. Possible illumination changes in the scene, large object movements, noise in the sensor data, shape or appearance changes of the object or scene, partial occlusion or disappearance of object, sensor movement, loss of information projection from object space to tracking space or need of real time computational power requirement can make the tracking process hard. There is currently no algorithm which overcomes each of these problems and provides fully accurate object tracking. So tracking problem is usually constrained by some assumptions to simplify the tracking process. For example, one can assume that there is no sudden changes in the appearance or motion of the object. Or one can set the sensor as stationary and the object is not subject to occlusion and disappearance. Moreover, the prior information about the object can be provided, such as possible trajectory, appearance, and illumination conditions.

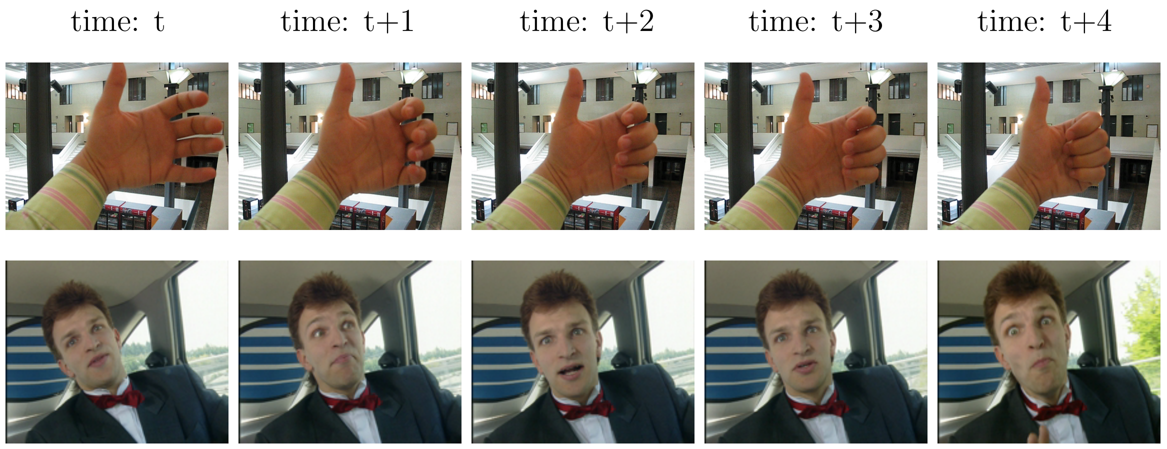


Figure 6.1: Sample two image sequences for tracking the objects in them [87]. First rows images belong to a video in which the hand of a person supposed to be tracked. The second set of the images contain a human face which needs to be tracked over time. At the top of each column, their timestamps are given.

The sample image sequences taken from two different videos to track the hand of a person and the face of someone can be seen in Figure 6.1 [87]. Both of the image sequences have their timestamps changing from t to t+4. In the image sequences of the first row, the hand moves and make some gestures. As it can be seen that the hand is deforming. The tracking process is supposed to locate the hand points in each of these images changing over time. In the other image sequences which can be seen in the second row, the head of the person moves and he mimics. In this case, tracking the face means that locating all the face points in the images over time.

Object tracking algorithms have a variety of uses, some of the applications are:

(1) Robot vision: The robots need to know its surrounding environment, so tracking the objects around itself is so important to accomplish its task. For example, an unmanned ground vehicle needs to track the road to drive by itself [35] [152]. Or a humanoid robot needs to track the face [66] [108] and hand [199] [68] of a human to interact to a person.

(2) Video compression and indexing: Object detecting and tracking is a part of

some video compression algorithms and multimedia retrieving process from databases [131] [1] [162].

(3) Automated surveillance: The tracking system is used to monitor the object movements, people and events in the public or private places, such as in front of a building, a shopping mall, a road, car parks, etc. The system needs to distinguish the interest of the object and track it [75] [184] [98] [42].

(4) State/Trajectory prediction of the vehicles: Aircrafts and missiles process their sensor data to track their positions over time [8] [178] [201].

6.1 Related Work

The problem of tracking objects in the environment has taken attention of the computer scientists for last five decades. Numerous object tracking algorithm exist in the literature. Each of them works well under different constraints about the environment and the object. One who searches for an object tracker should first determine his/her needs to achieve high accuracy and accomplishment of tracking task. Listing the features of the object, environment, and the motion simplifies to find the most proper tracker. This list defines the scenario of a task specific problem. For example, in one scenario, the object may have non-linear motion, the object shape might be non-rigid shape and the environment is not complex. In another scenario, object appearance might change during the tracking process, move at constant velocity and requires real time tracking. Each of these scenarios requires different type trackers. Also, the researchers design the algorithms by first setting the tracking problem to a scenario.

The first known object location estimation mechanism was developed by a Hungarian scientist, Rudolf Kalman, in early 1960s. His method, called as *Kalman Filter*, is described in [79]. *Kalman Filter* became the main object tracking principle for years. Engineers discovered the application of it in different areas. It has been used in the satellite navigation, aircraft autopilots, positioning systems of the ships, radar tracking, and visual tracking systems. *Kalman Filter* involves linear Gaussian models and not capable of non-linear and/or non-Gaussian systems type problems, so it was extended to *Extended Kalman Filter* and *Unscented Kalman Filter* [78] [77] [181].

Kalman filter is weak at the problems in which the posterior of the tracked object needs to be multi modal. *Particle Filter* is designed to handle this kind of problems. *Particle Filters* was born at early 1950s. Its first mathematical methods was described in [113]. It became one of the most popular object tracking methods in last two decades. *Particle Filter* has been highly started to applied in computer vision problems after it was showed in [9] [72] that it can be used to track objects robustly in the image sequences.

Particle Filter has been accepted as one of the most powerful approach in the computer vision community to solve the problems in which the tracked object can be represented by a low dimensional modal. Rough borders of the object can not be enough in some cases as the result of *Particle Filter* tracking process, more detailed border representation might be necessary. Active contours model was introduced in [81] and applied intensely in the tracking problems which requires exact representation of the object borders. This method, called as *Snakes*, attempts to adjust the initialized contours on the borders of the object in an iterative framework. An energy function is constructed and associated to the contour. Minimum value of the energy function represents the object borders. This energy function is minimized iteratively in the *Snakes* framework. The final position of the contour draws the object border.

There is an extensive of work on object tracking in the field. Its crucial need for some applications make the scientist to focus and develop many different approaches. The tracking methods differ according to object representation types. Different object representations require different solutions. A categorization of object representations was made recently by Yilmaz in [205]. One can represent a tracked object as a point and propose a method as described in [160]. To track a non-rigid object, it can be formed from primitive geometric shapes as in [31]. If the object has a complex shape and its exact detailed border have to be output, an active contour based method [206] can be the solution.

Tracking the objects by representing them as point-wise is also possible. Optical flow is one method to estimate the motion of the points in given image sequences [44] [106] [11] [107] . This method assumes that the brightness of the point is constant in consecutive images. Optical flow method can be extended to estimate the movement of all points in the images [44]. In this way, all object points in the scene can be tracked.

Graph cut based methods provide point-wise object tracking. These kind of approaches is introduced first in [24] [135]. A distance penalty method was explained in [110] to eliminate non-object regions by penalizing them in the graph cut energy. This was a novel approach to convert the graph cut to a tracker. However, the future work of that approach did not come to the field.

The methods in [24] [135] introduced a novel way to add the locations of the detected objects as high-level observations. Multi-layer graph cut construction for object tracking is described in these works. Background subtraction is performed and a set of candidate object blob set obtained in the preprocessing step. Special high level observation nodes are added as a layer in the graph. These nodes provides temporal and spatial consistency for the object tracking between the frames. The pixel displacements between the frames are estimated by Lucas-Kanade method. The color information is carried only from the previous frame. Complete occlusions are handled and new objects are created after some time later if they were occluded. A second stage post-processing, a second multi label graph cut with alpha expansion algorithm, is performed for the merged objects. In the graph cut based tracker explained in [135], occluded parts of the objects are tracked over time by adding some new terms into the graph cut. The object pixels which don't move close to average displacement of the object in last frame are penalized, so that they are most likely disappeared or occluded in next frames.

A straightforward graph cut tracking method is described in [130]. In this tracker, RGB color histogram models are carried from previous frames to the next. A new model is combined with the existing one coming from previous frame. A simple location prediction mechanism is employed. To speed up the graph cut, some preprocessing is applied in the constructed graph, then the graph is cut.

A pixel-wise graph cut and optical flow based object tracking algorithm for augmented reality is presented in [122]. Since augmented reality applications are so depended on the computational time of the methods, this algorithm works in real-time. It does not rely on the preexisting 3D models, but the locations of the objects points are calculated with the help of the optical flow.

In live video object tracking tasks, the frames cannot be layered to construct a 3D graph and segment the object in this graph structure. Therefore, the locations of the pixels must be carried from one frame to the next one. [51] describes a method to construct a location probability map of the pixels for the next frames. This approach utilizes from the location map by employing it into the graph cut method to track the objects in live videos. A method which combines optical flow and graph cut to track the people only using color images is explained in [169]. In this approach, the people are modeled as a ellipsoid shape and the location of a person is estimated by the optical flow type method. Graph cut is performed at each frame based on the predicted models.

6.2 Approaches

Point-wise object tracking is possible in two main ways. In first way, the location of the object is estimated in next frames. This estimated location is somehow incorporated into the tracking procedure. In the second way, the object is tracked by not estimating the location of it. In this approach, the location obtained from the last frame is carried to the next frame. These two types of developed methods are explained in the next sections.

6.2.1 Tracking Without Location Estimation

In this work, single image refinement method explained in Chapter 4.2 is extended to work as a tracker over video sequences of the moving objects. Graph cut tracker is initialized with the ground truth positions of the object in the first frame. Graph cut segmentation algorithm explained in Chapter 4.2 is performed in each frame of the tracking process. The desired object region is retrieved as a mask from the graph cut. However, the raw object mask generated by graph cut technique might contain some noisy, small and weakly-connected foreground regions, because the images contain a non-homogeneous color distribution inside the object regions. In order to clean up those regions, morphological opening and closing are done and found the connected components. The largest region is taken as the final object region, M_t. This object region is carried to the next frame without changing its location in the image space to form the fore/background models of the graph cut method. The carried object region in next frame is called as, M_{t+1}. Simply, the foreground, M_F, and background, M_B, models are obtained by dilation and erosion processes of the carried object mask, M_{t+1}, in next frame. The process is illustrated in Figure 6.2.

6.2.1.1 Results

For the comparison, in the way of how graph cut segmentation algorithm is extended for tracking, the SVM object segmentation is also extended to *SVM-Tracker*, as described in [87]. After an initialization, the object model is updated based on the classification results from the previous n frames by adding the newly predicted positive and negative examples and throwing away the oldest examples, like a sliding window scheme. However, the training data for SVM is no longer ground truth but the classified results from the previous n frames. The graph cut tracking algorithm which only has color information is called as *Color GC-Tracker*, the one which also includes distance penalties is called *ColorD GC-Tracker*, and the SVM-based tracker is *SVM-Tracker*. *ColorD GC-Tracker* involves color and distance penalty information.

In this experiment, the accuracies of the trackers are measured and analyzed. The trackers are started with the ground-truth image in the first frame and run until the frame which has a ground-truth. The generated object mask in the last frame which has a ground-truth is saved. Then, trackers are re-started by setting the object

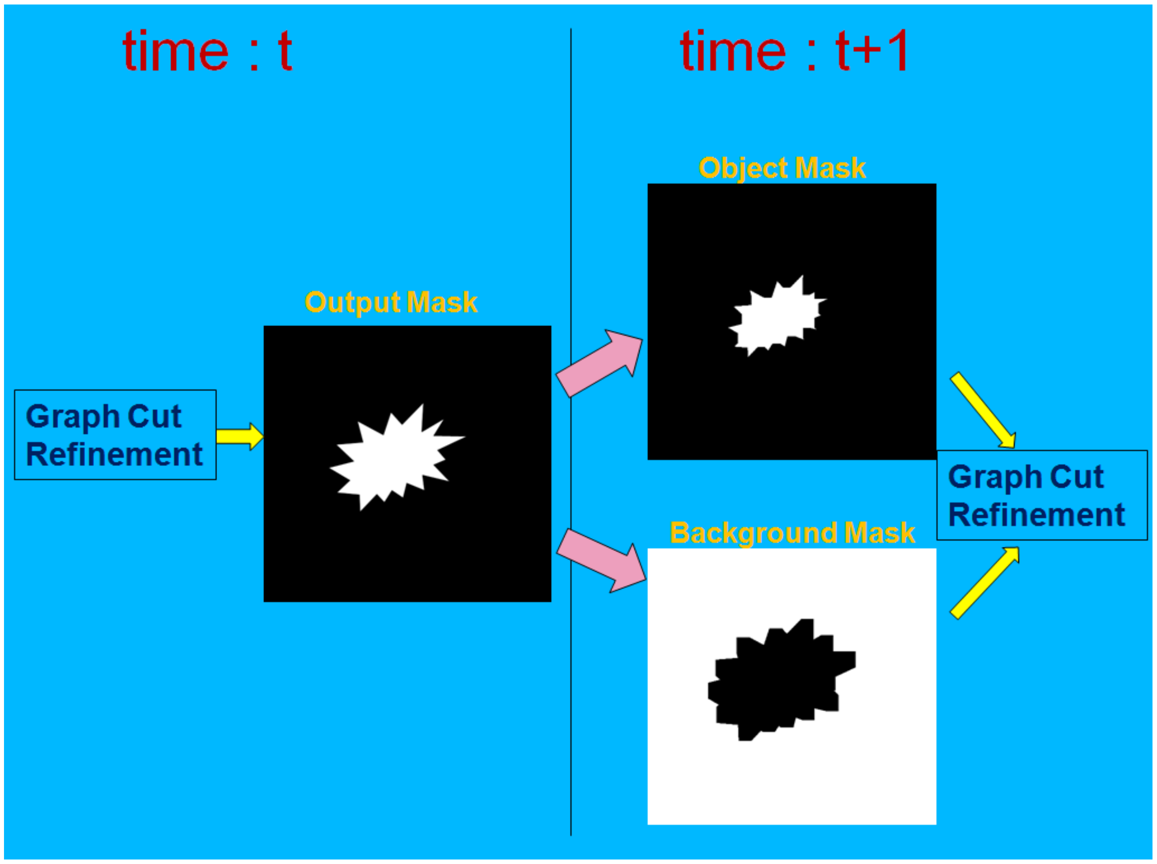


Figure 6.2: Calculation of foreground and background models of the single frame graph cut tracker. The output mask of the object at time t is carried to the next frame. The fore/background masks are obtained by dilation and erosion operations. The white colored points display the masks of the models.

region to the ground-truth. Total 502 results, the number of the images in the ground-truth sets, are produced in this way for *Trail*, *Hand*, and *Head* data sets. The details of these datasets are explained in Section 4.1.4. The same area overlapping formula in Section 4.1.4 is used to measure the accuracies of the trackers. Table 6.1 shows the median overlap scores of this experiment. Some results of the trackers from the trail, hand and head datasets can be shown in Figure 6.3. These results belong to the frames which have ground-truths in the datasets.

Overall accuracy of *ColorD GC-Tracker* is comparable with *SVM-Tracker* as can

Method Name	*Trail*	*Hand*	*Head*
CC-Tracker [150]	0.72	—	—
Color GC-Tracker	0.46	0.92	0.27
ColorD GC-Tracker	0.77	0.96	0.82
SVM-Tracker	0.74	0.95	0.68

Table 6.1: Median overlap scores of the trackers for the data sets. Since *CC-Tracker* is specific to tracking trails, it does not have overlap scores for the *Hand* and *Head* data sets. The details of these datasets exist in Section 4.1.4.

be seen in Table 6.1. *ColorD GC-Tracker* performs better than *SVM-Tracker* in this test. The big overlap score difference between *SVM-Tracker* and *ColorD GC-Tracker* for *Head* set test is caused by similar colored background and foreground regions in the images. Distance penalty information incorporated in *ColorD GC-Tracker* helped it for better tracking.

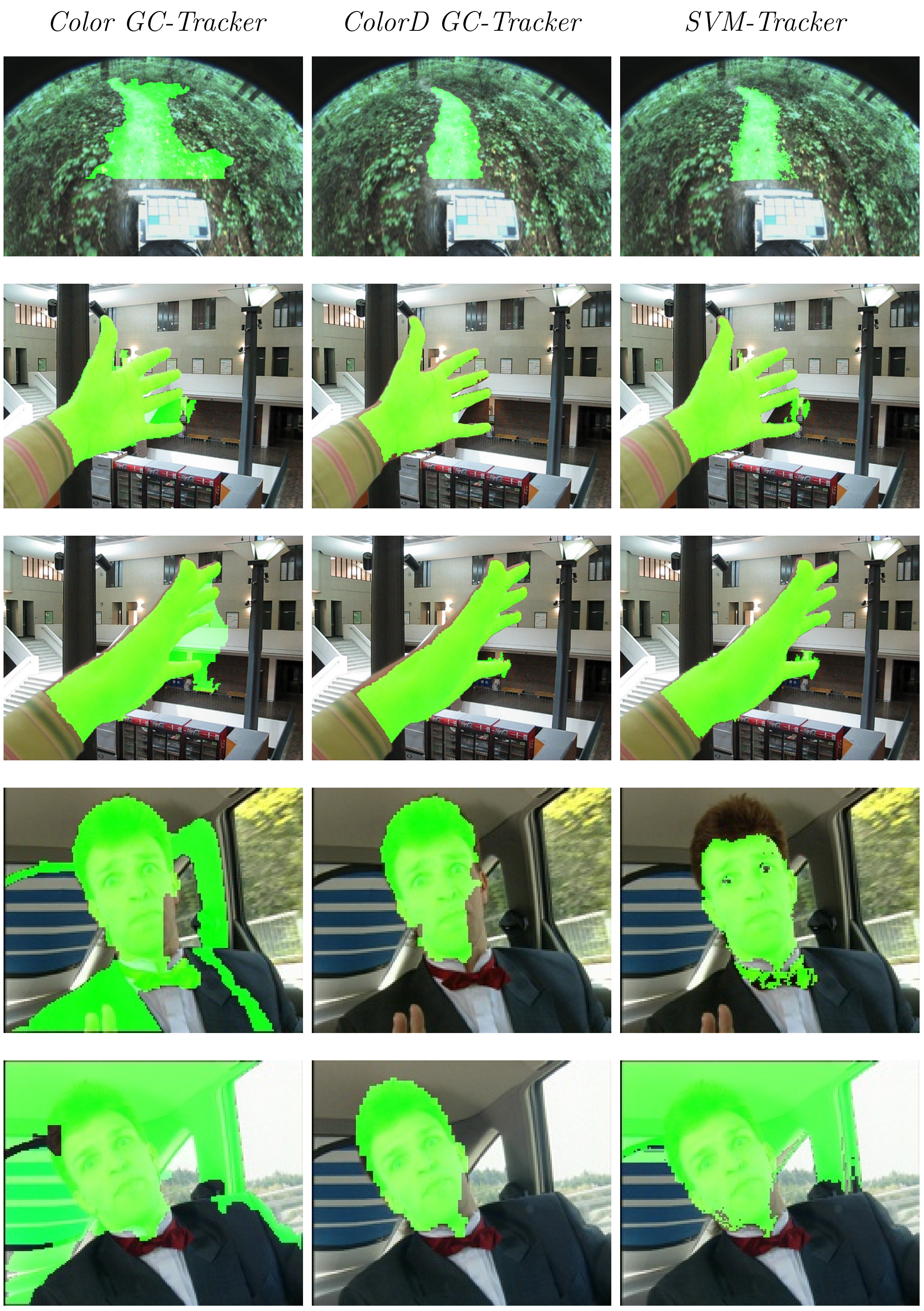


Figure 6.3: Sample results of the trackers. First row results are from *Trail*, next two rows are from *Head* and last two rows show *Hand* dataset results. The results of each row belong to the same ground-truth image.

6.2.2 Tracking by Estimating the Location of the Object

The method, *ColorD GC-Tracker*, explained in the previous section does not estimate the location of the object, but it assumes that the object makes no movement in the next frame. However, in real life scenarios, either the object or the camera might move in the scene. It sometimes is possible that both of the camera and the object might move. In order to deal with these conditions and build more robust trackers, it is better to incorporate the estimated location of the object to the point-wise tracker. Therefore, a point-wise object tracker which incorporates SIFT feature [104] [105] based location estimation mechanism is developed. Also, *ColorD GC-Tracker* utilizes only from the color images. In order to make the tracking process more robust, the newly developed tracker, called as $MM-Tracker$, fuses the features which are obtained from the images of the depth and color cameras. A diagram of $MM-Tracker$ is given in Figure 6.6.

6.2.2.1 Close Look at the Keypoints

A keypoint in a scene can be defined as an interesting point in its surrounding area. Keypoints are usually found in high contrast regions, such as at the corners, and on the edges of the objects. They are supposed to be invariant to the illumination changes, orientation, scaling and distortion. A set of the keypoints obtained from an object can describe that object. These set of the keypoints can be called as the feature descriptors of the objects. The keypoints are useful in wide range of tasks in the computer vision field, such as object recognition, tracking, localization and mapping of the intelligent systems, 3D reconstruction, camera calibration, and building panorama images.

An example usage of the keypoints is the robot localization. In order to localize a robot in a map, one of the ways is to find the keypoints around the robot and match them with the keypoints previously obtained from the map. Keypoint computing can be made in any sensor space, such as in the color images and/or LIDAR data. In the localization method of an indoor robot moving inside a museum, this kind of approach is used [182]. The color camera pointing toward the ceiling of the museum captures the ceiling mosaics. The keypoints extracted from the mosaics are matched in the ceiling map of the museum. Thus, the location and pose of the robot is estimated.

Harris corner detector [57] is one of the first proposed eigenvalues based corner detector. It is not scale invariant. [117] extended this work by using the determinant of the Hessian matrix, and the Laplacian to make Harris corner scale invariant. [164] made small modification to Harris corner detector by changing its scoring function and used this feature for the object tracking.

Later, Scale Invariant Feature Transform (SIFT) is introduced in [104] [105]. It computes the keypoints as the maxima or minima of the difference of the Gaussian convolved images at different scales. Difference of the Gaussians, called as DOG, is obtained by convolving the image with two different Gaussian filter at different scales and taking the difference of the responses of these two convolutions. Some unstable or low-contrast keypoints, and edge responses are eliminated. The final descriptor includes local orientation information. It is distinctive and invariant to scaling, orientation, and partially invariant to illumination changes and affine distortion.

A variation of SIFT descriptor is proposed in [82]. It applies Principal Component Analysis (PCA) to the normalized gradient patch, instead of computing smoothed weighted histograms. This method is called as PCA-SIFT. The performance of the local feature descriptors are analyzed and compared in [118]. It was showed that SIFT outperforms the other descriptors in most cases. According to this study [118], SIFT is more distinctive than PCA-SIFT. Also, another variant of SIFT, called as GLOH, is introduced in this publication. It is more robust than SIFT. However, it is computationally more expensive. Speed Up Robust Features (SURF) is keypoint descriptor based on Haar wavelet responses [6]. It is faster than SIFT, because the image convolutions rely on the integral images. In some cases, it is as robust as SIFT descriptor.

Keypoints can not only be computed in the 2D images, but also they can be obtained in 3D point clouds. [99] explains a generalized version of SIFT, called as RIFT, which can produce SIFT keypoints in a 3D Point cloud.

A descriptor which relies on the principal curvatures of the point is proposed in [30]. This method is called as Local surface patches (LSP). The quality of the point is described by a Shape Index (SI). SI is considered as maximum and minimum principal curvatures at that point. Also, SI frequencies of the neighbor points are accumulated in a histogram. The point is declared as a keypoint, if its SI is outlier in its neighborhood.

The method explained in [211], called as ISS, computes eigenvectors in the covariance matrix of the neighbor points centered at a point. Centroid is aligned with the principal axes. The means of the curvatures are obtained from the surface which is fitted to the aligned data. The quality of the keypoint is defined by this mean. [114] follows a similar way to compute the keypoint quality. However, it only considers the magnitude of the smallest eigenvalue, and the ratio between the eigenvalues.

The approach described in [207], called as meshDOG, is a scale invariant keypoint detector method. It computes the extrema points in the Laplacian response images after applying a set of Gaussian Filters. A certain percentage of the extrema points are thresholded by sorthing them according to the magnitude of the responses. Non-stable responses are eliminated by using the Hessian operator. In this way, strong corner points are selected as the keypoints.

SURF [6] is extended for 3D point clouds in [85], called as 3D SURF. The shape is first voxelized in a volumetric cube. The quality of the keypoints are determined by the determinant of the Hessian matrix of Gaussian second-order derivate which are calculated by the box filters. [183] proposes a descriptor to encode the local features of a 3D keypoint, called as Signature of Histograms of OrienTations (SHOT). The histogram is built by the first-order differential of the normals of the points inside the grid. The angle, between the feature point and each point within a grid is computed and histogrammed.

[12] [158] [47] [2] compare and analyze the performance of 3D keypoint detectors for the object recognition and retrieval tasks. The experiments are made by using several synthetic and real datasets collected by different sensors, such as Laser Scanner, and the stereo cameras. Their overall discussions show that [114] with its scale invariant

version, called as KPQ-SI, MeshDOG [207], and ISS [211] gives the best performance in terms of the repeatability. ISS is the most efficient keypoint detector. MeshDOG outperforms the other methods when the scale is the issue. KPQ performs best among the fixed scale detectors. Also, it is robust to noise in the data. Furthermore, MeshDOG suffers if the data is not dense, for example on the Laser Scanner datasets.

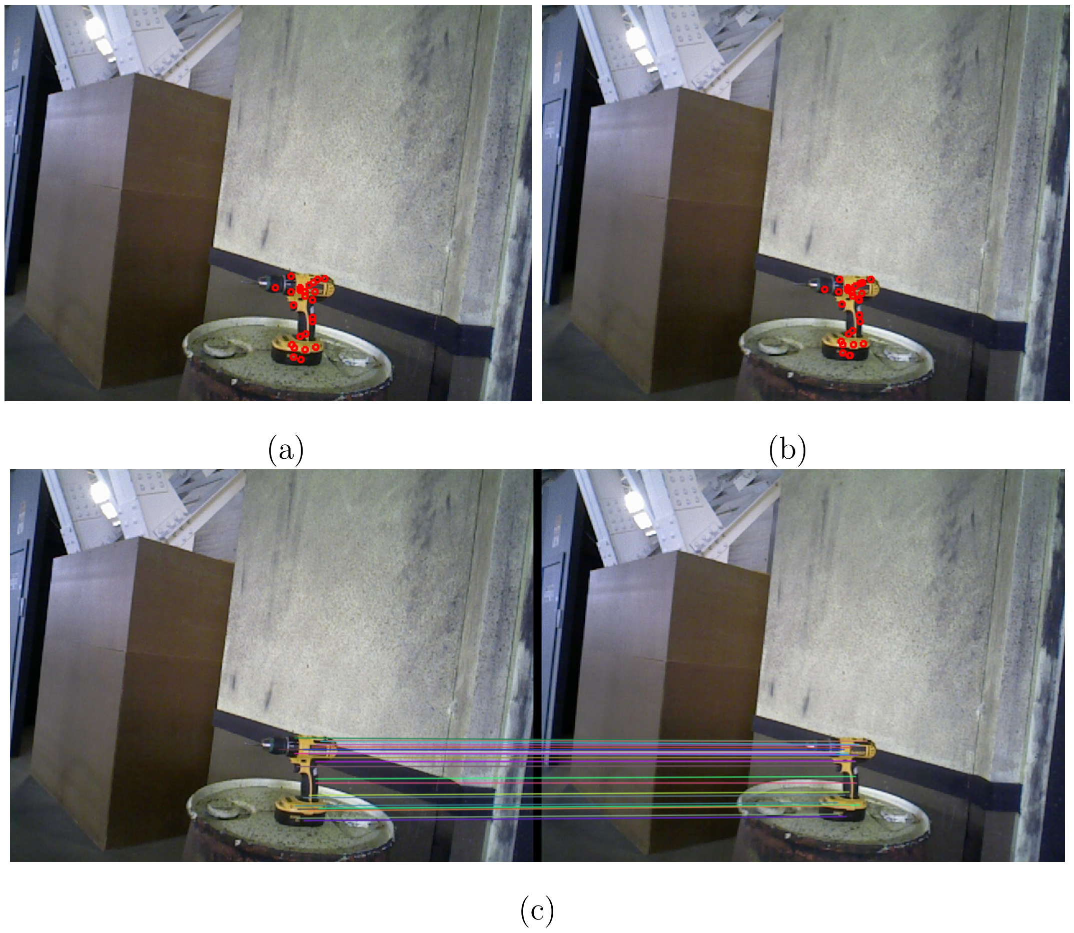


Figure 6.4: Matching the keypoints inside the object (it is a drill in this case) between two frames. (a) displays the image at *time t*, (b) shows the color image at *time t+1* of the same image sequence. Detected SIFT keypoints are drawn as red in both of the images. (c) illustrates the matched keypoints between the images.

6.2.2.2 SIFT based location estimation

Detecting the keypoints in a 3D point cloud is computationally more expensive than detecting them in a color image. Hence, $MM - Tracker$ prefers to compute them in the color images instead of in the corresponding point clouds. SIFT feature descriptor is used to estimate roughly the displacement of the object from time t to time $t + 1$. There are several reasons why SIFT is chosen for this purpose. As mentioned in the previous section, SIFT feature descriptor is invariant to uniform scaling and orientation. Also, it is partially invariant to the distortion and illumination changes. In addition, it is fast to compute, so it is suitable for $MM - Tracker$. However, corresponding 3D point cloud of the image obtained from the depth image is used for projecting keypoints to 3D space in $MM-Tracker$. The displacement of the keypoints between the frames are computed in this 3D space to achieve more accurate estimation. If the displacement of the keypoints would be calculated in the color image space, it would be so sensitive to the scale changes between the frames. Hence, the keypoints are detected in the image space, but the displacement of them is computed in 3D point cloud space. The location of the object at time $t + 1$ is estimated by the following steps:

1) For given two color images, I_t and I_{t+1}, whose time stamps are t and $t + 1$ respectively in the image sequences, their SIFT keypoints are computed:

$$K_t = \mathscr{S}(I_t) \ \ , \ \ K_{t+1} = \mathscr{S}(I_{t+1}) \tag{6.1}$$

where K_t is the set of the keypoints, and $\mathscr{S}$ is the operator to obtain SIFT keypoints from the image I_t.

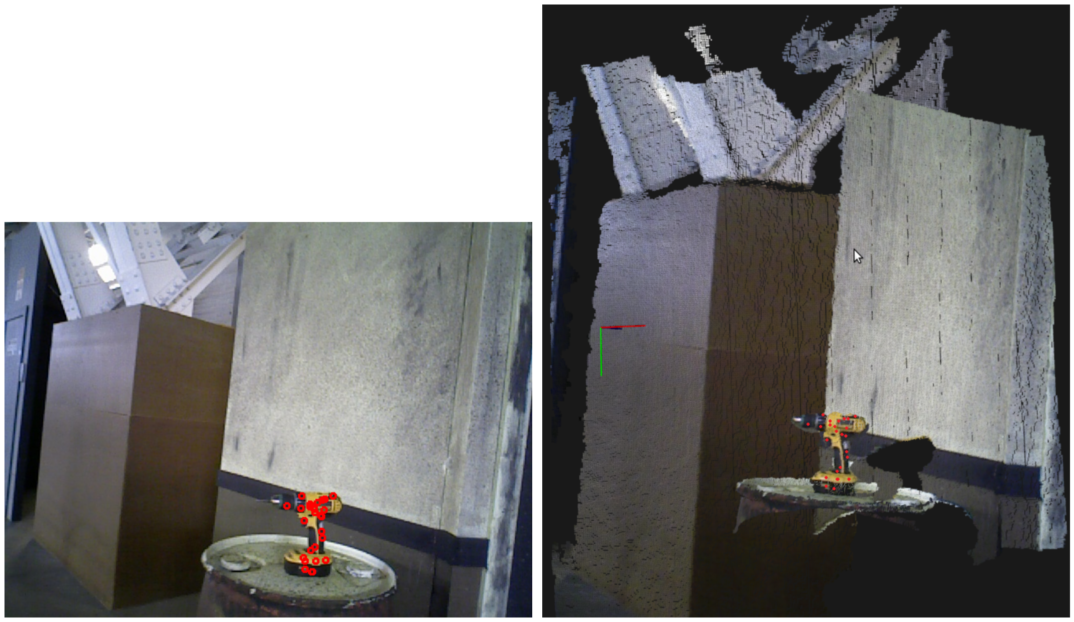

(a) (b)

Figure 6.5: Taking the detected keypoints inside the object from the color image space to 3D point cloud space. (a) displays the image at *time t* with the keypoints drawn as red, (b) shows the corresponding point cloud and the keypoints in it. The scene is rotated in the Point Cloud Viewer for better display.

2) The keypoints which are outside of the object region are excluded only from K_t:

$$K_t^- = \{k(x,y) \in K_t \ : \ k(x,y) \notin M_t\} \tag{6.2}$$

$$K_t = K_t - K_t^- \tag{6.3}$$

where $k(x,y)$ is a keypoint in the image, M_t is the object region in the image.

3) The keypoints in K_t and K_{t+1} are matched:

$$K_{t,\,t+1} = \mathscr{M}(K_t,\ K_{t+1}) \tag{6.4}$$

$$K_{t,\,t+1} = \{k_t(x_1, y_1), k_{t+1}(x_1, y_1)\}, ..., \ \{k_t(x_n, y_n), k_{t+1}(x_n, y_n)\} \tag{6.5}$$

where $K_{t,\,t+1}$ is the set of pairs of keypoints matched from K_t to K_{t+1}, and $\mathscr{M}$ is the keypoint matching function. The detected keypoint inside an object for given two consecutive images, and matched keypoints between them can be seen in Figure 6.4. In this case, the object is considered as a drill.

4) Convert the depth images to 3D point clouds for timestamps of t and $t+1$ in the image sequences. Then, compute the corresponding 3D points of all keypoints in $K_{t,\,t+1}$:

$$k^{3D}(x, y, z) = T(\mathscr{P},\ k(x, y)) \tag{6.6}$$

$$K^{3D}_{t,\,t+1} = \{k^{3D}_t(x_1, y_1, z_1), k^{3D}_{t+1}(x_1, y_1, z_1)\}, ..., \ \{k^{3D}_t(x_n, y_n, z_n), k^{3D}_{t+1}(x_n, y_n, z_n)\} \tag{6.7}$$

where T is the function to compute the corresponding point in the point cloud $\mathscr{P}$ for a given keypoint $k(x, y)$ in the image. As an example, Figure 6.4 shows the keypoints inside the object in both color image and its corresponding 3D point cloud after this step is applied.

5) The displacement of the object from the time t to the time $t+1$ is calculated as the average displacement of the paired keypoints:

$$p^{3D}_{dis} = \frac{1}{n}\sum_{i=1}^{n} k^{3D}_{t+1}(x_i, y_i, z_i) - k^{3D}_t(x_i, y_i, z_i) \tag{6.8}$$

where p^{3D}_{dis} is the displacement of the object.

6) All object points, M_t, at time t are converted to its 3D representation, M^{3D}_t, by using function T as in the Equation 7.6. Then, estimated location of the object at time $t+1$, M^{3D}_{t+1} is computed by adding average displacement, $p^{3D}_d is$, to each point of M^{3D}_t:

$$p^{3D}_i = T(\mathscr{P},\ p_i) \quad where\ p_i \in M_t,\ \ p^{3D}_i \in M^{3D}_t \tag{6.9}$$

$$M^{3D}_{t+1} = M^{3D}_t \ + \ p^{3D}_{dis} \tag{6.10}$$

7) Since the camera calibration parameters are known, also estimated object points in the image, M_{t+1} can be achieved:

$$p_i = T^{-}(p_i^{3D}) \tag{6.11}$$

where $p_i \in M_{t+1}$, $p_i^{3D} \in M_{t+1}^{3D}$, and T^{-} is the transformation function from the 3D world space to the image space.

6.2.2.3 Feature Extraction and the Classifier

The human classifier, explained in the previous chapter, $H - Classifier$, uses a point-wise shape descriptor, f_s. This shape descriptor fuses the normal, η_i of a point p_i, its vectorial spatial distance, Δ_v, and geodesic distance, GD_i, relative to the middle point, mid_B, of given bounding box. The experiments shows that this shape descriptor powerful in the point-wise classification problem of the human. It is originally depended on the middle point of the bounding box. However, it is extendable to other type of the objects or conditions only by finding a way to calculate this middle point.

There is no a given bounding box around the objects in point-wise tracking problem of the objects, so the middle point, mid_B, is considered as the center of the mass of the object, M_t^{3D}. It is calculated by the following formula:

$$mid_B = \frac{1}{m}\sum_{i=1}^{m} p_i^{3D} \tag{6.12}$$

where m is the number of the points in M_t^{3D} and $p_i^{3D} \in M_t^{3D}$.

The shape descriptor, f_s, can be incorporated as a cue to the proposed tracker, $MM-Tracker$, by modifying mid_B as described. In addition to the shape information of a point, p_i, also the color of the point is added to the point-wise descriptor, f_p, to make $MM - Tracker$ more robust. f_p becomes:

$$f_p = [\eta_x\ \eta_y\ \eta_z\ \Delta_x\ \Delta_y\ \Delta_z\ GD_i\ r\ g\ b]^T \tag{6.13}$$

where r, g, and b are red, green, and blue channels of the point, p_i, in the color image of a sequence.

The positive, f_p^+, and negative, f_p^-, samples are obtained from inside and outside of the object, M_t, as in the training process of $H - Classifier$. In order to classify the points in next image whose time stamp is $t + 1$, a tracking classifier, $T - Classifier$, is trained with the samples obtained from a image set, S_{img}. This image set, S_{img}, includes the images whose timestamps are t, $t - 1$, ..., $t - n$. This mechanism can be considered as sliding a tracking window, w_t, on the image sequences and the size of the w_t is n. Since it has been showed Random Forest algorithm performs best in the previous section, it is chosen for $T - Classifier$ as the framework.

In the classification process of the points in next image with time stamp t+1, first the object location is estimated by SIFT based method explained in previous section. The center of the mass of the object, mid_B, is computed according to the estimated object location. Then, the points of this frame is classified by $T - Classifier$ which is trained by the samples obtained from previous image set of n frames.

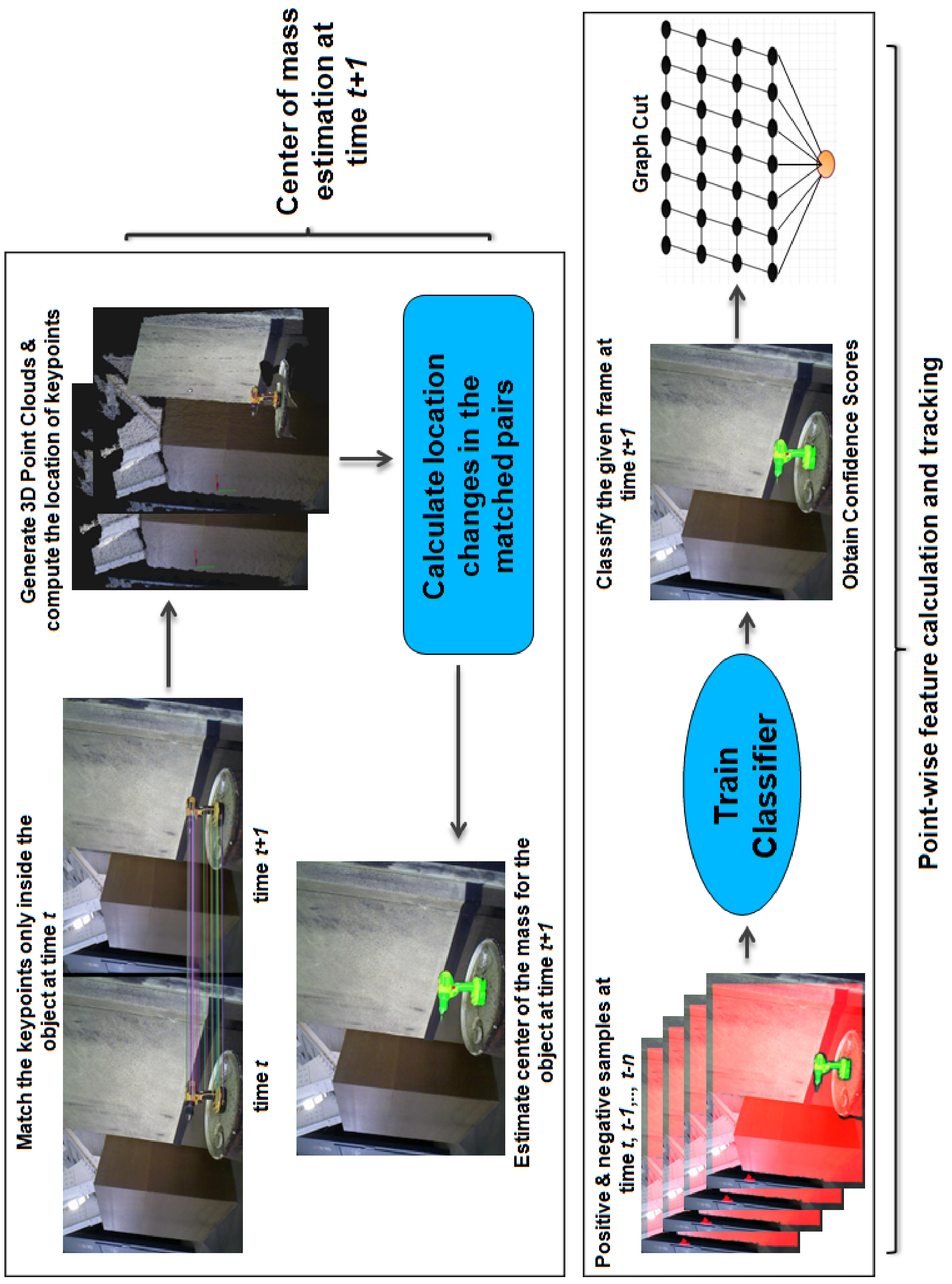

Center of mass estimation at time t+1
Match the keypoints only inside the object at time t
time t
time t+1
Generate 3D Point Clouds & compute the location of keypoints
Calculate location changes in the matched pairs
Estimate center of the mass for the object at time t+1
Positive & negative samples at time t, t-1,.., t-n
Train Classifier
Classify the given frame at time t+1
Obtain Confidence Scores
Graph Cut
Point-wise feature calculation and tracking

Figure 6.6: Overall diagram of multimodal tracker.

6.2.2.4 Graph Cut Smoothing

$T-Classifier$ provides a confidence score, $\mathcal{C}_i$, of being in object region for a point, p_i, in the image. The details of this function is showed in Equation 5.10. In this case, $\mathcal{C}_i$ is the decision distribution at the leaf node of the Random Forest Tree. As discussed earlier, graph cut is great method to achieve more smooth results or to eliminate noises in the final result mask of the object, M_{t+1}. The confidence scores, $\mathcal{C}_i$, can be used to set edge weights in graph cut. Therefore, $MM-Tracker$ applies graph cut technique as the last step in the tracking process by utilizing from $\mathcal{C}_i$. In this case, only single layer graph framework without incorporating any high level observation is constructed. The *Regional* and *Boundary* terms of the graph cut is formulated as:

$$R(l_i) = -\ln(p_c(p_i, l_i)) \tag{6.14}$$

where $R(l_i)$ is the regional term, l_i is the label of the point, p_i, in the image, and $p_c(i, l_i)$ is the likelihood of the point as it can be shown in Equation 5.21.

$$B_{i,j} = |R_i(\text{"}Object\text{"}) - R_j(\text{"}Object\text{"})| \tag{6.15}$$

where $B_{i,j}$ is the boundary term penalty two neighbor points p_i and p_j. $R_i(\text{"}Object\text{"})$ and $R_j(\text{"}Object\text{"})$ are the regional penalties of being in the object region of those points.

6.2.2.5 Discussion on the Motion Model and Temporal Tracking

The purpose of proposed location estimation method is to be able to use it for all type of the objects. It is simply a generic technique which do not include any constraint in its motion model. It does not make any assumptions in the motion of the object, so it moves only the center of the mass of the object according to displacement of it between the frames in 3D world space. Even though, this type of the motion model is not the best option to track a specific object, it is applicable for all kinds of the object, such as rigid, articulated, and deformable.

The proposed motion model can be modified/updated for some certain type of the objects, if their characteristics are known beforehand. For example, if the target object which is tracked is a car on the road, the motion of the car is constrained by some parameters, such as minimum turning radius of the car, maximum possible speed can be made by the car, the slope of the car. In this case, a specific motion model based on these constraints can be developed for the car. If the road is not inclining in any direction, even one of the axes in the motion model can be dropped and it can be considered that the car is moving on a 2D plane. After establishing such a motion model, the movement of the car can be estimated by a RANSAC based method computing the displacement among the keypoints. Also, this type of estimation method can remove the outliers and/or mismatches in the keypoints. It can tolerate the problems caused by the keypoint matching methods.

The motion model to track a human differs than tracking a car. In contrast to rigid shape of a car, a human has articulated parts. These articulated parts can move independently than each other. Hence, it is not a correct assumption that all points in the human body will move under all conditions. For example, a human can only raise up and/or wave his hands. In another case, he/she can walk on a planary surface. Instead of using just one motion model for all points in the body, it seems more reasonable to develop part based motion models for the human body.

Proposed tracking method does not utilize from Kalman or Particle Filter type approaches. It seems as open to noise accumulation in the temporal displacement over time. However, it is assumed that the classification and graph cut steps, explained in the previous sections, will prevent from the error accumulation by working as some kind of internal fixing steps. The training and the graph cut processes at the each step of the tracking help to remove some incorrectly estimated object points by SIFT based location estimation method, explained in the previous section.

6.2.2.6 Results

Several experiments were conducted to analyze and measure the performance of the proposed $MM - Tracker$.

In the first experiment, a scenario in which a ground robot is far away from a hand tool which is a drill is established. In this scenario, the robot would like to walk toward the drill to grab it by avoiding from the obstacles on its path. The robot might make different movements because of the obstacles, such as left/right turns, while walking. The target point of the robot finally it will arrive is near the drill. Properly tracking the drill is important for the robot for several reasons. First of all, it shows the path/way for it. Also, in order to be able grab the drill for a certain task, it must recognize and know the orientation/position of the object relative to its arms to execute correct motion plan. For this problem, tracking its low dimensional representation is not enough, but more detailed point-wise representation is necessary. Alternatively, this task can be done by detecting the drill at each frame and then refining detailed borders of the drill whenever it arrives near of it. However, the detection methods require to search entire scene which is computationally expensive and at the end the refinement process might not be accurate.

Tracking and finally grabbing an object whenever the robots comes near it is a common problem for the ground robots. The object might be any kind of hand tool, house or kitchen equipments. Darpa Robotics Challenge [61], *DRC*, became the inspiration to create this kind of scenario to test the described method. The tasks in this competition that a robot is expected to execute ranging from driving a small car, closing valves, climbing the ladder, to use some hand tools in dangerous, degraded and human-engineered environments. In one of the tasks, the robot must accomplish using a drill to open a hole on the wall.

A similar testing environment as in *DRC* tried to be created to record a dataset, called as *DRC-Track*. The setup moved around the object while recording the data. Different types of movements that the robot can do while walking is captured, such as left/right turns, approaching to the target or walking away from it. Both of the time aligned and same space registered depth and color images are saved. Also, their color registered 3D point clouds were generated. The frame rate was set to about 20 per second during the recording. *DRC-Track* contains the depth and color images

Method Name	*Median Overlap Score*
$MM - Tracker$ (Only Color)	0.16
$MM - Tracker$ (Only Shape)	0.74
$MM - Tracker$ (No Location Estimation)	0.41
$MM - Tracker$	0.96
$GC - Tracker - 3D$ (No Location Estimation)	0.23
$GC - Tracker - 3D$	0.48
$Farneback$ [44]	0.28
$Godec$ [54]	0.46

Table 6.2: Median overlap scores of the methods for *DRC-Track* dataset. Each row specifies an experiment in which different combinations of the cues are used in the method. For example, $MM - Tracker$ (Only Shape) means that color cue is removed from the point-wise descriptor, f_p, of $MM - Tracker$. $MM - Tracker$ (No Location Estimation) means that the displacement of the object between the frames is not computed for that experiment. In the same way, $GC - Tracker - 3D$ (No Location Estimation) does not include the SIFT based location estimation between the frames for $GC - Tracker - 3D$ algorithm.

of 1900 frames. Total 76 ground-truths (one for every 25 frames) of the object is manually generated by a human utilizing from a special pixel labeling software. These ground-truths were used to compare the performance the tested methods.

The results of the $MM - Tracker$ was compared to several other methods in the same category which aim to track objects as point-wise. A dense optical flow method, $Farneback$, is explained in [44]. In contrast to detecting a tracking a set of keypoints, this method estimates the displacement of all points inside a mask in next frames. Its good performance comparing to other types of the optical flow methods showed in [44] and its dense point-wise tracking characteristics became the reasons to compare its results to $MM - Tracker$.

Another method explained, in [54], is used in the experiments to analyze the results of $MM - Tracker$. This method includes a self online learning of the object appearance to track it. Its online learning step is based on generalized Hough-Transform which provides a rough estimation of the object. The rough segmentation of the object is given to the *GrabCut* [157] to obtain final point-wise mask of the tracked object. This method outperforms some of the state-of-the-art methods to track non-rigid objects for several challenging videos. This method will be called as *Godec* hereafter.

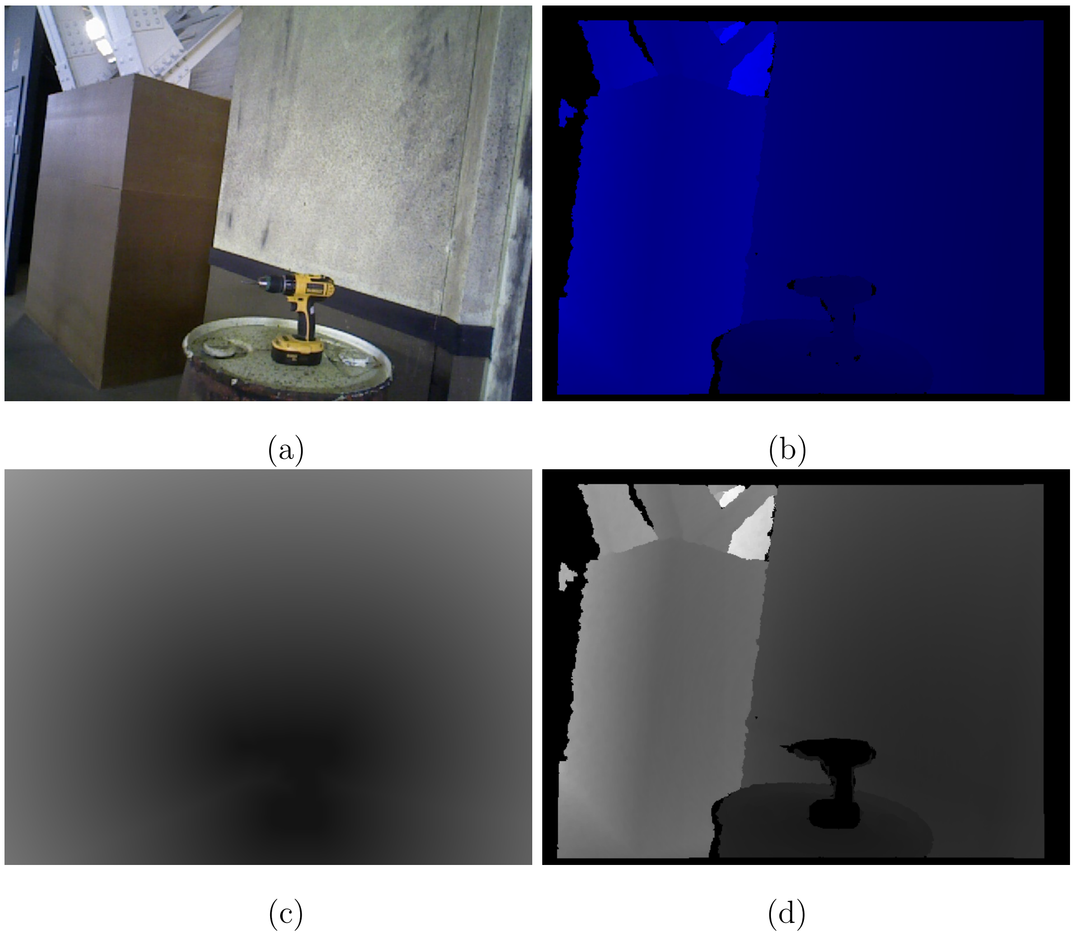


Figure 6.7: Examples of 2D and 3D distance penalty maps. (a) displays given color image, and (b) its depth image. (c) shows its 2D distance penalty map computed in 2D image space, and (d) its distance penalty map computed in 3D. Please note that no penalty is assigned for missing depth data and inside the object. Darker points in the maps have less penalty.

Also, *ColorD GC-Tracker* whose details are explained in Chapter 6.2.1 was modified to be used in the comparisons. *ColorD GC-Tracker* was a single frame graph cut based tracker which does not estimate the location of the object in next frames. In other words, it works as refinement method for given next frames. In addition, it computes the distance penalty term in 2D image space to incorporate into its *Regional* term in the graph cut formula. The same SIFT based location estimation approach, described in Chapter 6.2.2.2, has been embedded to give the ability of predicting the object in next frames. In this way, the produced object mask by the method at time *t-1*, is shifted by the estimation method at time *t*.

Moreover, the availability of depth data provides the opportunity of computing distance penalty map in 3D Euclidean space. In order to prepare a 3D distance penalty map, first the 3D point cloud of the scene is generated. Then, Euclidean distance of each point to the object is calculated. The sample distance penalty maps of two ways, one is computed in 2D image and the other one is computed in 3D space, are showed for a given image in Figure 6.7. As it can be realized in (d) image of this figure, even though some points are close to the border of the object in the image space, they might have large penalties due to their distances to the object in 3D space. *ColorD GC-Tracker* with these two modifications will be called as $GC - Tracker - 3D$ hereafter.

The performance of $MM - Tracker$ was compared to $GC - Tracker - 3D$, *Farneback* [44], and *Godec* [54] by fusing different cues in its point-wise descriptor, f_p. Two tests were conducted by removing the shape or color information from $MM - Tracker$. Also, in another test, all cues were kept in f_p, but the displacement of the object between the frames were not computed for $MM - Tracker$. It is simply assumed that the object does not move in the next frame. This test was conducted for $GC - Tracker - 3D$ in order to see how $GC - Tracker - 3D$ performs in a refinement way. In all of these tests, the trackers were re-started with the ground-truth masks whenever they reached those ground-truth frames. In other words, the trackers were re-initialized with the ground-truths per 25 frames. Their produced object masks at the last frame were recorded to analyze their performances and to compute the overlap scores with their ground truths. Some results of the trackers when they hit a ground-truth can be seen in Figure 6.10. Total 76 overlapping scores were calculated in this way. Median overlapping scores of the methods can be seen in Table 6.2.

The distance penalty constant term, β, of $GC - Tracker - 3D$ in Equation 4.6 is set to 1.49 and the constant term, λ, for weighting the smoothness term is set 0.1 in the experiments. It was observed that finding a good combination of these constant terms, β and λ, sometimes becomes cumbersome. More importantly, using constant weights cause problems in the tracking process, if locally different distributions of the color exist around the object borders. For example, if the color discontinuity with the background is not big near the bottom of the object, but it is large in the top side of the object, playing with different variations of the constant terms do not improve the results. This is mainly because of weighting the cues globally, every point in the scene achieves same equal weight between the cues in $GC - Tracker - 3D$. The weighting mechanism is explicit which is handled by the user for this method. Therefore, as it can be seen in Table 6.2, $GC - Tracker - 3D$ performed worse than $MM - Tracker$. As it is expected, estimating the displacement of the object improved the results of $GC - Tracker - 3D$ by taking the median overlap score from 0.23 to 0.41.

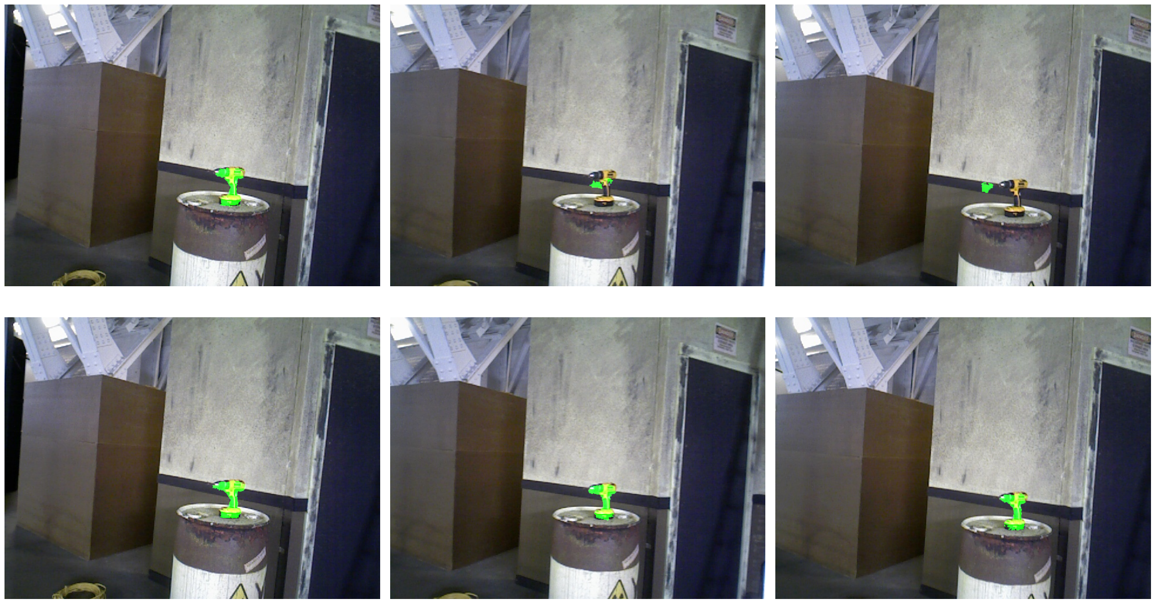


Figure 6.8: Sample results of $MM - Tracker$ with and without location estimation of the object over time. First row results show the outcome of $MM - Tracker$ without estimating the displacement at different time. Second row displays the results of $MM - Tracker$ if it includes SIFT based location estimation method described in Chapter 6.2.2.2. In this example, the camera makes first a right and then a left movement.

In contrast to $GC - Tracker - 3D$, $MM - Tracker$ has an implicit learning mechanism. It can figure out that which cue is more important and distinctive in which part of the scene around the object. This ability is given to it by the vectorial spatial distance, Δ_v, part of its point-wise descriptor, f_p. In the experiments, sliding window size of $MM - Tracker$, w_t, is set to 4. Only one tree was trained to reduce the computational load. The max deep of the tree is set as 40. It was observed that training one tree produced sufficiently good results. $MM - Tracker$ which includes all cues and SIFT based location estimation outperformed all other methods by having highest median overlap score 0.96. Also, the experiment showed that the shape related cues in the descriptor, f_p, is so powerful. Even it may produce remarkable results in the absence of the color information. However, the color itself is enough to track the object in front of background which share similar color information with the object. Terrible performance, only 0.16 overlapping score, is showed by $MM - Tracker$, if it does not integrate shape cues. It was concluded that estimating the displacement of the object is crucial for $MM - Tracker$. In the absence of the location estimation, the performance dropped from 0.95 to 0.41. One of the example case which demonstrates the mistracking of the object when the displacement is not integrated to $MM - Tracker$ can be see in Figure 6.8. In this example, the camera first moved to the right. It caused the tracker to loose the object location and stick on the wall. Then, the camera made a left movement. The method kept track a part of the wall which has similar color cue. However, SIFT based location estimation employed in $MM - Tracker$ helped to continue to track the object over time as can be seen in the second row of Figure 6.8. *Godec* [54] performed slightly worse than $GC - Tracker - 3D$ in this test. Since the background has same similar color information as the object has, and *Godec* does not use any 3D related features, this performance of *Godec* can be expected.

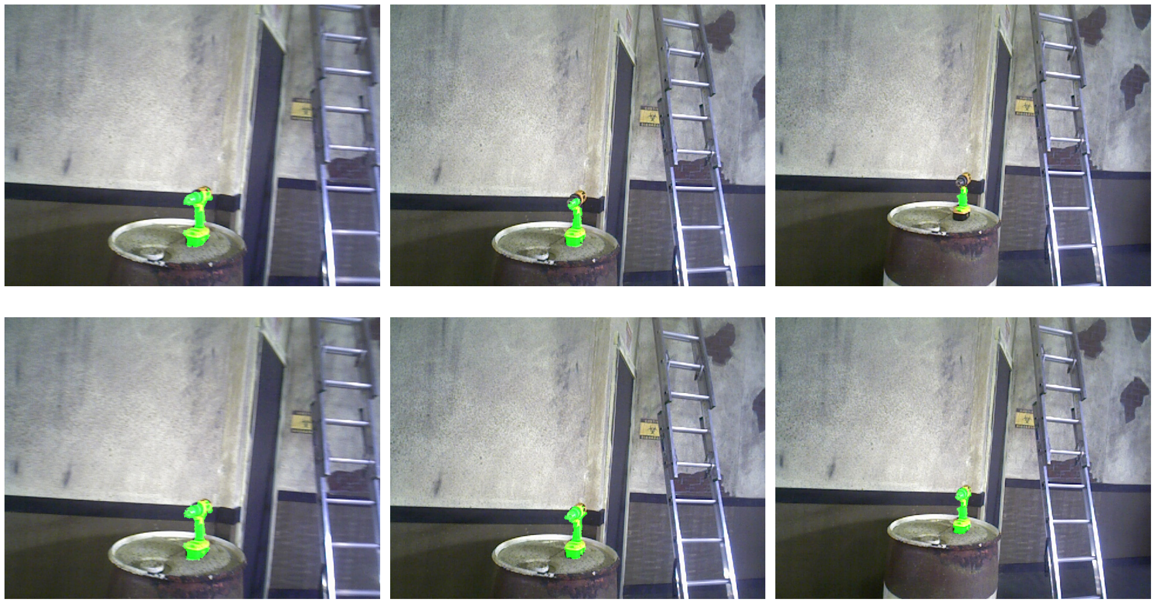


Figure 6.9: Sample results of $MM - Tracker$ with and without incorporating the color information at time t=0, 30, and 60. First row shows the results of $MM - Tracker$ without fusing the color in its point-wise descriptor, f_p. Second row displays the results of $MM - Tracker$ when all the cues are incorporated.

Figure 6.9 demonstrates a case when the color information helps $MM - Tracker$ to track the object properly. In this case, the changes in the shape of the object because of the camera rotation, lead to loosing some part of the object. Only shape related cues started becoming not enough when the camera was rotated over time. First row results display that $MM - Tracker$ sticked only middle part of the object not by grabbing the top and bottom part of the object. However, the color information which is associated with the point-wise shape related cues helped $MM - Tracker$ to work more robust in the case of noticable shape changes.

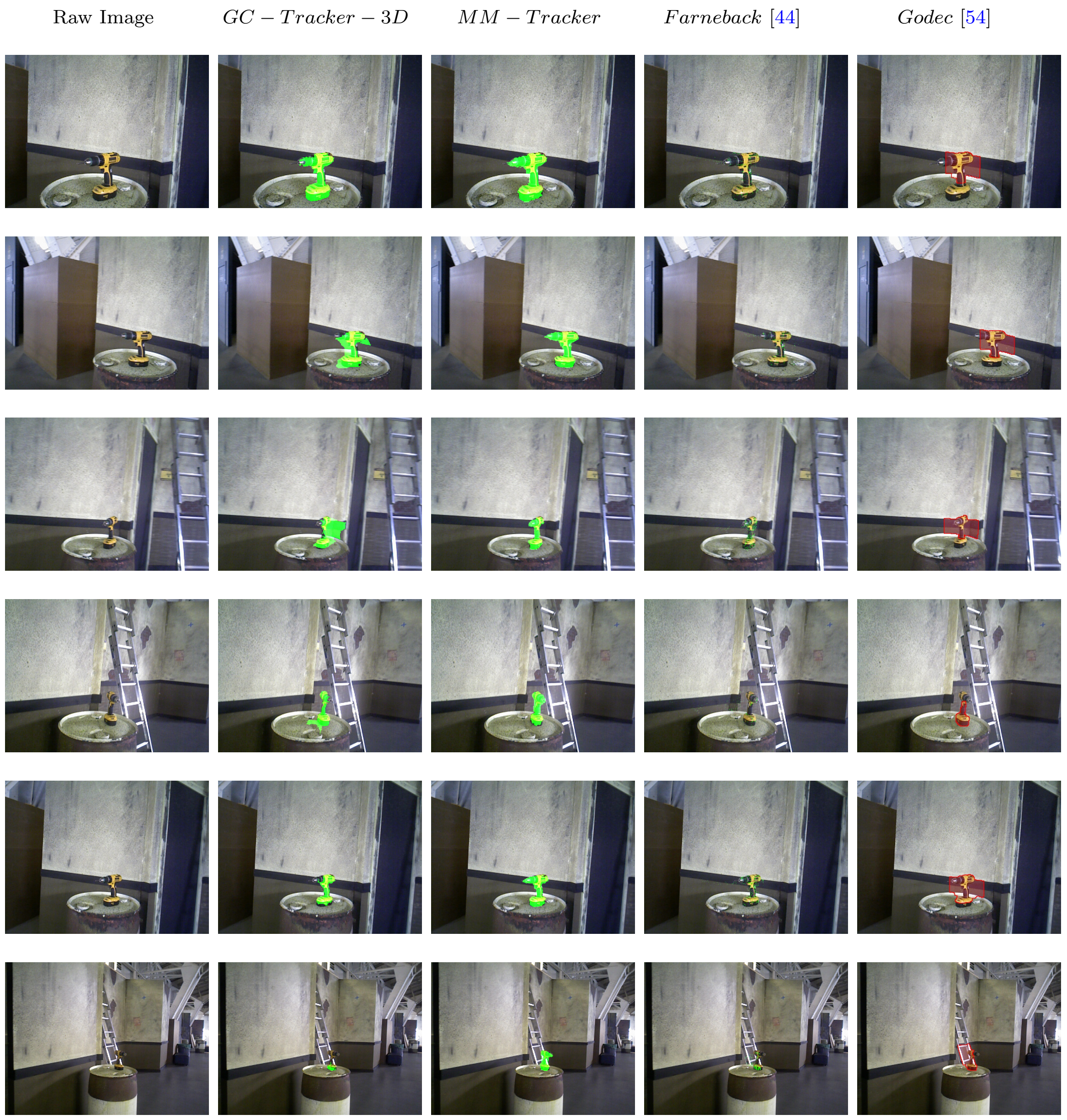


Figure 6.10: Sample results of different tracking methods whenever they hit a ground-truth in *DRC-Track* dataset. Column headings show the name of the methods. If the object was lost by the tracker, there is no green colored mask.

Dense optical flow method of $Farneback$ [44] utilizes only from the visual cues.

Method Name	*Median Overlap Score*
$MM-Tracker$ (Only Color)	0.18
$MM-Tracker$ (Only Shape)	0.82
$MM-Tracker$ (No Location Estimation)	0.73
$MM-Tracker$	0.93
$GC-Tracker-3D$ (No Location Estimation)	0.37
$GC-Tracker-3D$	0.43
$Farneback$ [44]	0.45
$Godec$ [54]	0.61

Table 6.3: Median overlap scores of the methods for *HandShaking* data set. Each row specifies an experiment in which different combinations of the cues used in the method. For example, $MM-Tracker$ (Only Shape) means that color cue is removed from the point-wise descriptor, f_p, of $MM-Tracker$. $MM-Tracker$ (No Location Estimation) means that the displacement of the object between the frames is not computed for that experiment. In the same way, $GC-Tracker-3D$ (No Location Estimation) does not include the SIFT based location estimation between the frames for $GC-Tracker-3D$ algorithm.

Therefore, $MM-Tracker$ outperformed its results by a factor of 2.6 at the level of the median overlap scores. It can be realized in Figure 6.10 that the results of $Farneback$'s method do not seem as dense, but some sparse points. The reason of this situation is that some computed displacements of the points inside the object hit same points in the next frame, so it causes the aggregation of the points to only one point.

Another experiment was conducted to analyze and compare the performance of $MM-Tracker$ for the articulated and deformable objects. Since the proposed SIFT based location estimation is a generic approach for all types of the objects, it does not consider the deformations and part based movements to determine local displacements of a group of the points inside the object. For this purpose, a short video of two persons approaching to each other, shaking their hands, and then walking away in front of a complex background was recorded. This dataset, called as *HandShaking*, includes 220 frames of the color and depth images. Its video frame rate is about 20 per second. In the same way as it was generated for *DRC-Track* dataset, a ground truth per 25 images was labeled manually.

The tested trackers were restarted at the ground truth images as it was performed in the previous experiment. The result masks of the methods at every ground truth were saved for the comparisons and analysis. Median overlap scores of the methods for this dataset can be seen in Table 6.3. $MM-Tracker$ produced again the best results in this experiment. It was observed that in the case of no location estimation, the results of $MM-Tracker$ for this dataset is considerably higher than *DRC-Track* dataset. It is mainly because of more situations where the object does not move so much or make movements towards the camera, such as during the hand shaking of the persons, at the start and end of the video. Also, the same reasons explain the better performance of *Farneback*'s method [44]. In this experiment, *Godec* [54] performed better than *Farneback* and $GC-Tracker-3D$. It can be said that *Godec*'s online learning step to build the object model works better than the refinement procedure inside of $GC-Tracker-3D$. Some sample results of the trackers recorded at the frames which have ground truths can be seen in Figure 6.13.

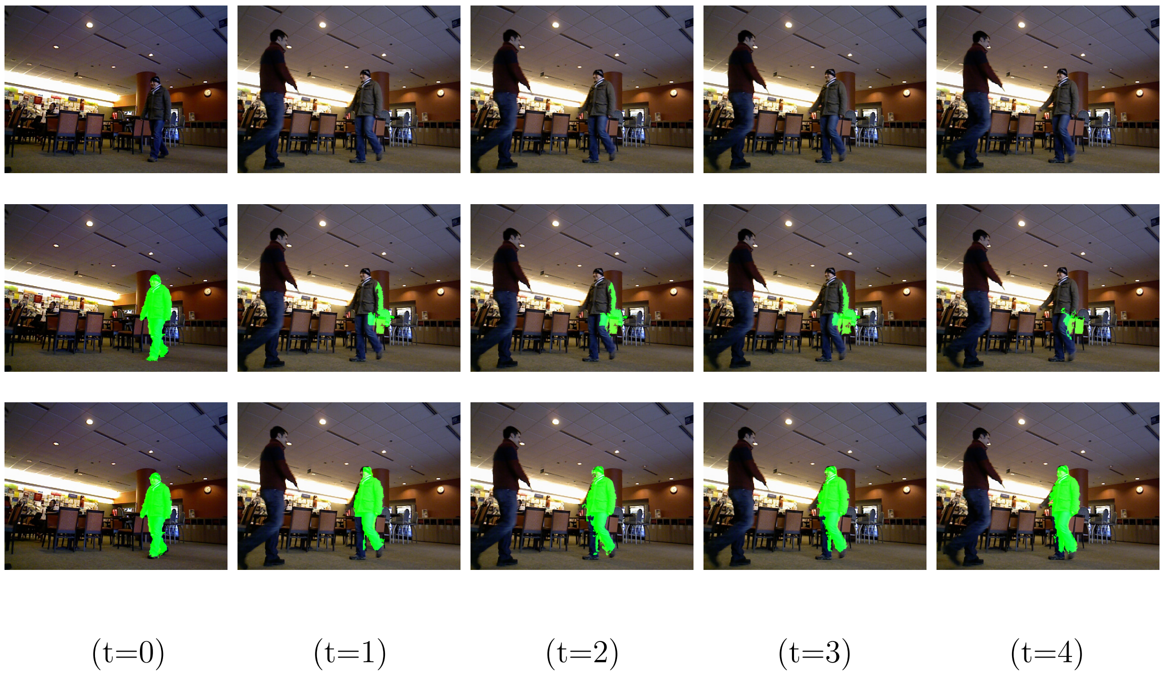


Figure 6.11: The effect of the proposed location estimation method during tracking of the person for a special case in which the hardware system stalled between t=0 and t=1. It caused a large skip at the object location. Second row displays the result of $MM-Tracker$ without utilizing from the SIFT based location estimation method. Last row shows the results of $MM-Tracker$ which includes the location estimation of the object.

A special and rare case was encountered during recording of *HandShaking* dataset. At one point of recording, the hardware stalled due to so rare buffer problem in the sensor setup. This kind of conditions can always be expected in real time robotics systems which require to process high bandwidth data. It is so useful, if the employed method can handle these kind of issues in software level. In this case, there had been large change at the location and pose of the human between two frames. Therefore, it was decided to analyze this case separately. Figure 6.11 demonstrates the results of $MM-Tracker$ with and without having proposed SIFT based location estimation method. The stall occurs between time=0 and time=1. As it can be seen in the second row of the Figure 6.11, not employing the estimation mechanism to $MM - Tracker$ caused loosing the person in the next frame. However, the location estimation helped $MM - Tracker$ to capture most part of the person at time=1, even though its pose at the foot and arm level changed. Also, $MM - Tracker$ was be able to grasp the other parts of the body at subsequent frames, after time=1.

It was investigated during this experiment that $MM - Tracker$ is sensitive to the local motions inside of the object. As it can be noticed in the fourth row of Figure 6.13, $MM - Tracker$ was unable to track some parts of the hand and the arm of the person. If the whole body of the person does not move but only his arm makes local movement, a generic SIFT based location estimation method is not enough to handle these types of the local motions. Also, same problem occurs when the motion is not homogeneous for the all points of the body. For example, when a person is walking the direction and magnitude of the motion in the head or torso of the person is different than in the level of the feet. In order to capture and integrate these type of motions into $MM - Tracker$, object specific estimation methods can be developed.

Figure 6.12 displays the point-wise confidence scores of given images produced by Random Forest Classifier and their final results obtained from graph cut step employed in $MM - Tracker$. It can be seen in the final results that some points which have weak scores inside the objects were included into the result masks. Moreover, some non-connected weak points were eliminated by graph cut. These improvements can be explained by the ability of graph cut which can combine smoothness and data terms in one framework.

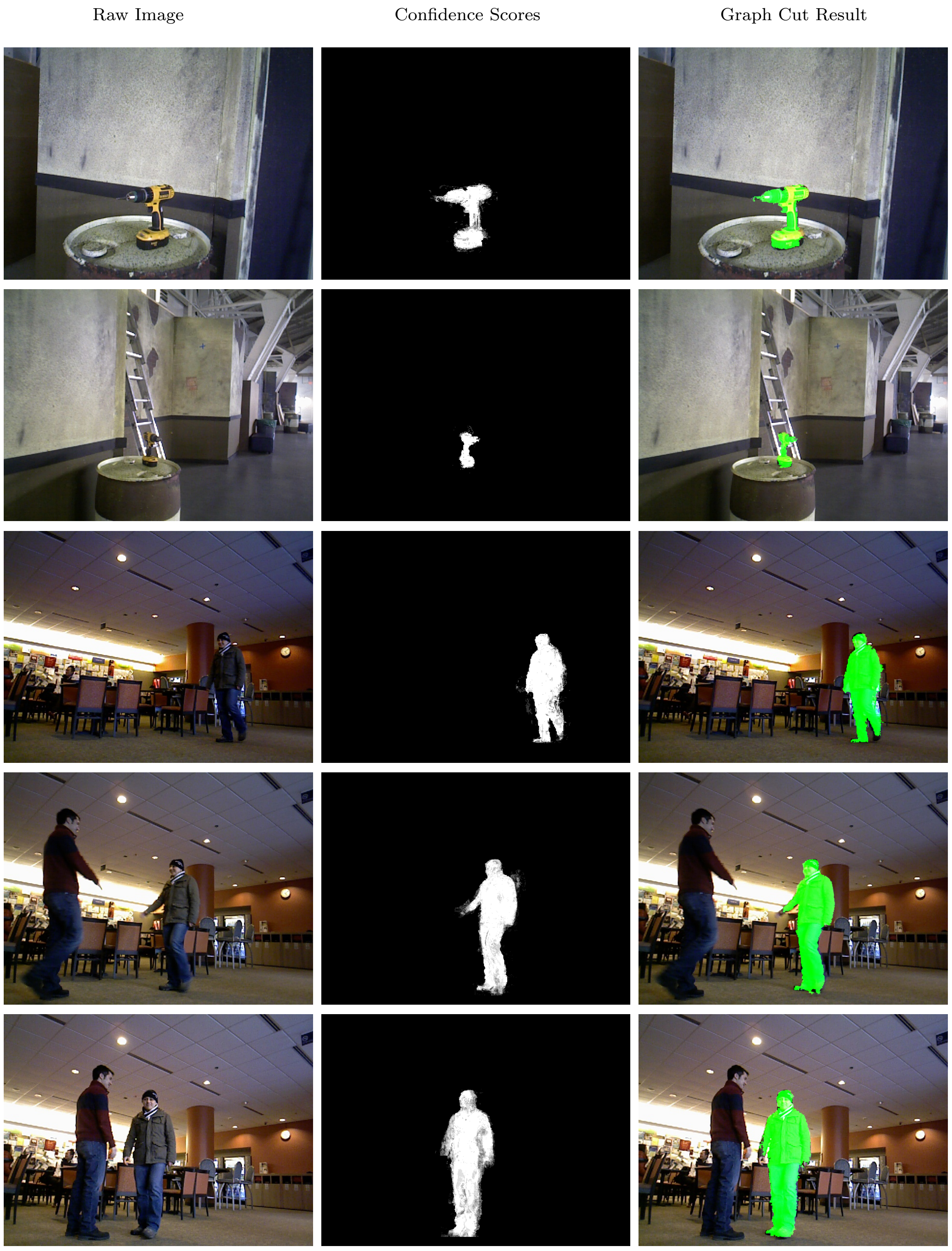


Figure 6.12: Sample confidence scores and their result masks produced by graph cut in $MM - Tracker$.

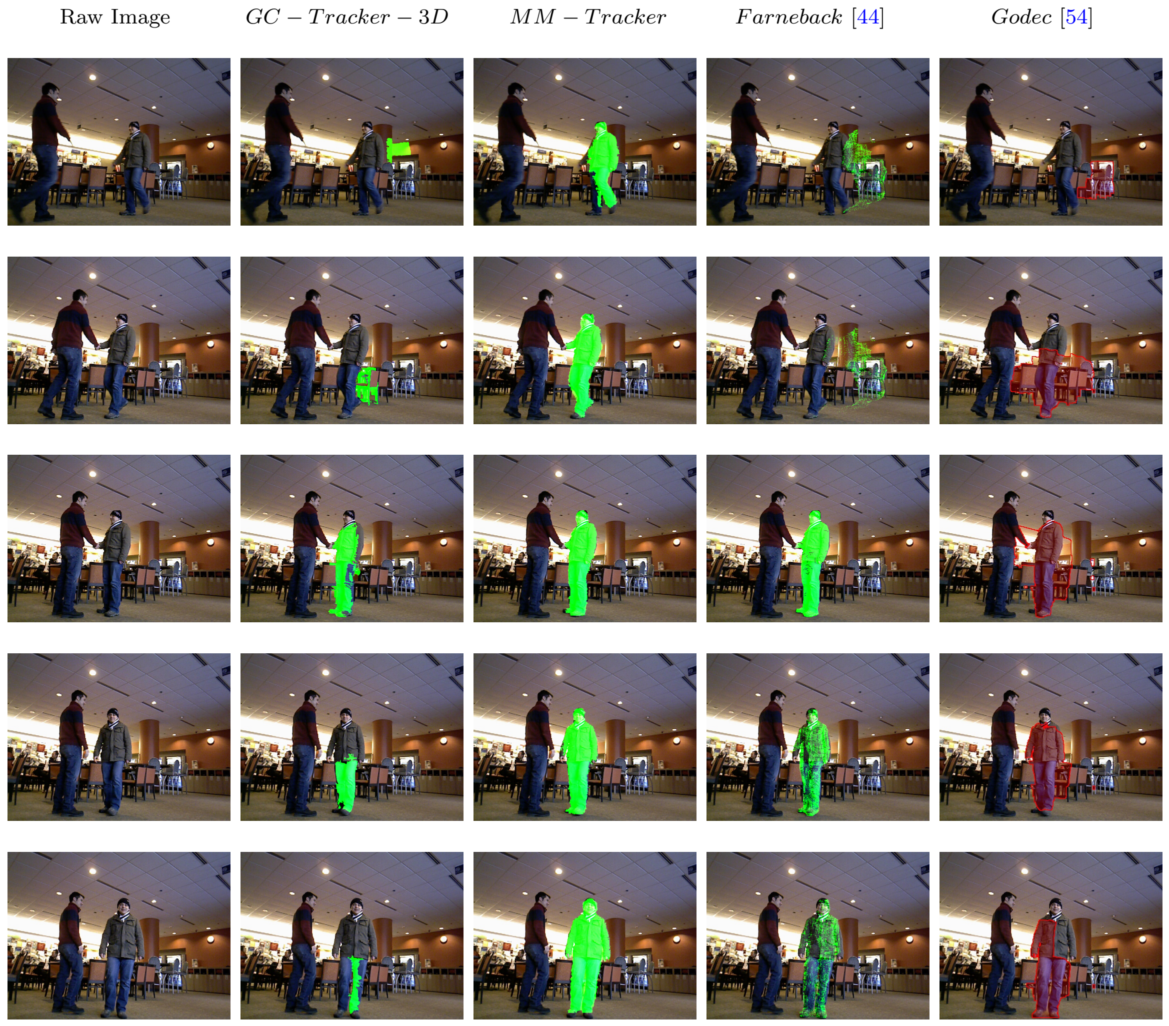


Figure 6.13: Sample results of different tracking methods whenever they hit a ground-truth in *HandShaking* dataset. Column headings show the name of the methods.

6.3 Conclusion

In this chapter, two different types of point-wise tracking methods are proposed. Their results are compared to the existing methods and analyzed. In the first proposed approach, it is assumed that the object is not moving, so the location of the object is not estimated over frames, but the result mask produced in last frame is carried without changing its location to the next given image. A graph cut based refinement method is developed to refine the carried mask iteratively. This method, called as *ColorD GC-Tracker*, combines the color and the distance information to the object fuses in its *Regional* term. It adaptively switches between the color space under different illumination conditions over time in order to work in most distinctive color space. Moreover, it employs dynamic distance penalty weighting mechanism which provides the ability of capturing distant object points around the border, if the selected color space really distinctive.

The performance of *ColorD GC-Tracker* is analyzed by using three different datasets which include deformable and complex shaped objects, such as trail, the head and hand of the people. The effects of incorporated color cue and distance penalty information are quantified in the experiments. Its results are compared to a same category SVM based tracker and a low dimensional trail tracker. The results of *ColorD GC-Tracker* outperformed the outcome of the other methods for all three datasets. It was observed that if the location of an object does not change so much, several iterations of *ColorD GC-Tracker* can produce globally optimal solutions.

On the other hand, a novel point-wise multimodal object tracker, $MM{-}Tracker$, is described. This tracker estimates the displacement of the object in a generic way from one frame to another by utilizing from the matchings of SIFT keypoints. It computes the center of the mass of the object by taking the mean of all displacements between keypoints. In this way, it does not estimate the local motions inside the object, but it assumes that all points move in the same direction and magnitude.

$MM - Tracker$ utilizes from a powerful point-wise shape descriptor in addition to the color cue of the object. This shape descriptor consists of vectorial spatial distance and geodesic distance information relative to the center of the mass of the object. Also, it includes the normal of the point. The availability of depth data provides computing the shape descriptor in 3D Euclidean space. Positive and negative point-wise descriptors are trained by Random Forest algorithm in $MM - Tracker$. In order to achieve robust performance, not only the samples obtained from the last frame, but also the descriptors coming from last n frames can be integrated. A graph smoothing process is applied to produce better results.

The performance of $MM-Tracker$ is compared two other point-wise trackers. In one of them, *ColorD GC-Tracker* is upgraded by a 3D distance penalty map which computes the Euclidean distance to the object in 3D space. Also, the same SIFT based location estimation technique is added to *ColorD GC-Tracker*. Other compared method is a powerful dense optical flow algorithm.

Two different experiments are performed to quantify and analyze the results of $MM-Tracker$. One of the dataset includes a complex shaped hand tool, which is a drill, and the other dataset consists of two persons approaching each other to shake their hands and then walking apart. In both of the experiments, $MM-Tracker$ showed the best performance. It was investigated that generic SIFT based motion estimation method is powerful, if the object is not highly deformable. Also, even if the object demonstrates local motion changes, $MM-Tracker$ performs better than compared dense optical flow method.

Chapter 7
CONCLUSION

7.1 Future of Computer Vision

In last decade, we have started seeing some computer vision applications as the home consumer products, such as Kinect, like.com, and incogna.com. There were some other companies which produce vision applications for specific consumer groups for years. However, there had been no product which addressed the need of every person until the launch of like.com or incogna.com which are the websites allow image based search. Partially, Kinect can get into this category by interesting the people from the age of 7 to 70 years old.

Why these kind of products were waited to be launched until the last decade? Examining the reasons of this question can enlighten the possible market grow of computer vision applications for the home consumers in the future. It can be clearly seen that the advances in some other fields of the technology effects the opening of new markets or developing new products. For example, the spread of Internet and web on the earth triggered the need of text based and image based search engines, so the accumulation of the research done in image retrieval field turned to a image based search website, like.com which was acquired by Google in its first years for 100 million dollars. Another impact factor can be the advances in the CPU technology which allows more computational power to create the solutions for a such expensive task.

Some other examples can be given before discussing about Kinect. A similar trend can be observed in the launch of youtube.com. The main reason in the opening of such an industrial market was the advance of video streaming technology and Internet infrastructure. After Adobe launched the best video streaming server, called Adobe Flash Media Server, and its player became available in most of the computers on the earth, youtube had opportunity to fill in the gap of video sharing problem on the web. Indeed, dynamic data update feature coming with Ajax for the web sites, called as the creation of Internet 2, was another reason why youtube was not investigated before 2005, even though the research community offers really good video codec algorithms.

Kinect was launched in the last quarter of 2010. It is the first device which allows the home users to play computer games by interacting with their gestures and movements. This was a time-breaking product for the computer vision field. It created a new market and also affected the research community by investigating to develop new vision algorithms which use depth data obtained from its IR camera. Therefore, its impact happened in two ways, to the industrial market and to the research community of the vision. The main underlying reason, which also answers the question of *"why did such a huge market arise in 2010?"* , is the advancements in the sensor technology. Kinect employs an IR camera to measure the depth of the points into scene. It opens a door to produce new, robust, complementary vision based software. Therefore, Microsoft research team was be able to develop human body gesture recognition algorithms which were not done robust enough by using only color camera to deliver in a home consumer device until that point. And again, the advancement in computer hardware industry allowed them to train their method using 1000 core CPUs.

Improvements in the sensor technology affected the autonomous car field. It was a dream thought to develop a fully autonomous car 20 years ago. However, we can see an autonomous car, Google car, nowadays which is be able to drive 300,000 miles by itself without having any accidents in real traffic. Google continues working on this project to change the roads of the future. This car employs a Velodyne LIDAR whose price is about 50,000 dollars. This sensor provides a 3D point cloud of the scene to help the car for perceiving its surrounding. A concrete effect of using this sensor can be seen in 2004 and 2005 Darpa Grand Challenge. No team was used this kind of LIDAR in 2004. In my opinion, it was a big factor why no team was be able to finish the competition. In 2005, Stanford team, Stanley, and some other teams updated their sensor setup by adding a Velodyne LIDAR. Stanley won the Grand Challenge. Also, in 2007 Darpa Urban Challenge, some teams such as CMU's Tartan Racing removed their LIDAR setup, and replaced a Velodyne LIDAR which shows that they realized the need of a reliable LIDAR which provides dense 3D data of the environment. To sum up, Velodyne LIDAR changed the way of the competition and the research which was conducted for that competition by the teams.

In addition, one more point can be mentioned about this topic. One of the factor which might improve the accuracy of a vision method is the availability of computational power. Parallel computation allows to reduce running time of some algorithms. For example, in order to detect a human, multiple threads can decrease the detection time by dividing the image space into several pieces to assign a thread for each of them. Or the number used particles in a *Particle Filter* tracking method can be increased in the availability of more computational power. In this way, more particles might improve the accuracy of the method. NVIDIA which is a company produces GPUs makes remarkable amount of investment to develop some computer vision products, especially for autonomous cars, by focusing on providing more computational power for the algorithms.

On the other hand, even just adding a sensor to an existing device opens new huge markets. Fore example, adding a camera to the cell phone created new software companies, such as Instragram which was acquired by Facebook for 1 billion dollars. Or it causes exiting market to grow, such as the video conference features of Skype and whatsApp. WhatsApp has been acquired recently by Facebook for 16 billions dollars. Also, it affected the research in the field of the imaging and computational photography. The research budget of these fields have been increased in last years due to availability of the cameras in cell phones.

It can be investigated from these examples that the advancements in the sensor technology effects the industrial market and at the same time the research community. I believe in that we are going to see new computer vision applications which uses multiple different types of the sensors in the future. This applications will generate new companies, correspondingly new billion dollar markets. Dell offers an option for its Alienware laptops to add an IR emitter to allow playing 3D games by paying extra 100 dollars. The annual market of Alienware laptops is millions of dollars. Can someone tell that a cell phone company or a PC company will not make the same offer in the future for adding the Infrared cameras as Kinect has? Imagine that a cell phone or your PC employs a sensor which provides the depth of the points. In that condition, new vision applications will be delivered to the market causing to establish maybe billion dollar companies.

Combining the advantages of different sensors in the same application generates new research problems. How much is going to be trusted to *sensor-x* or *sensor-y*? Which cues will be obtained? How are those cues which have different scales will be weighted by the algorithm? These are the fundamental questions need to be solved. Additional questions which are specific to the application type might raise. They need to be handled in the application context.

The idea of preparing the vision algorithms of the future which fuse multiple sensor data to increase the robustness as they can be employed to the consumer devices was one of my motivation for writing this thesis. I believe in that multimodal data fusion will be a big problem later as the technology produces new sensors. Current research community of the computer vision should get prepared for those incoming problems.

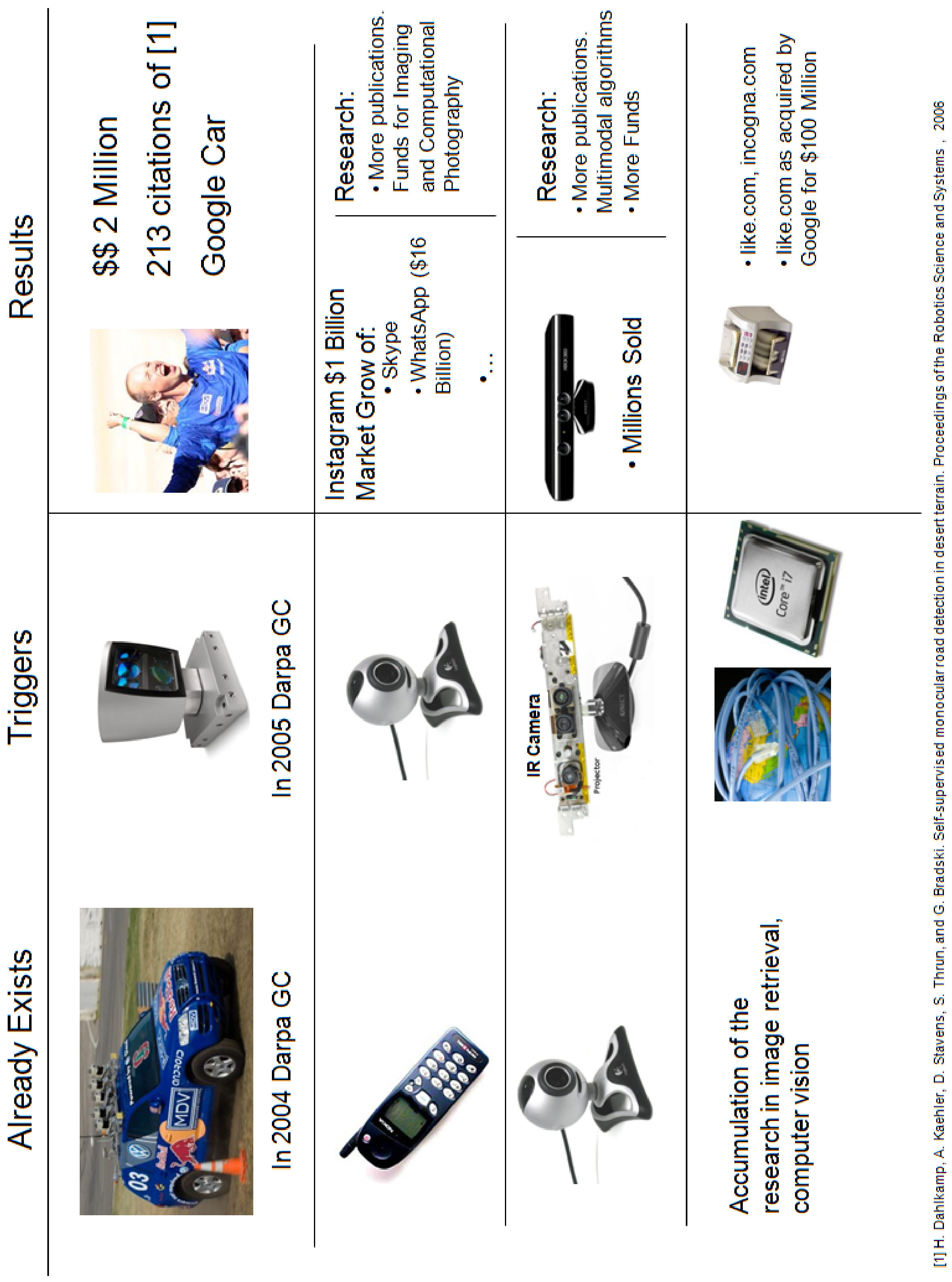

Already Exists
Triggers
Results
In 2004 Darpa GC
In 2005 Darpa GC
$$ 2 Million
213 citations of [1]
Google Car
Instagram $1 Billion
Market Grow of:
• Skype
• WhatsApp ($16 Billion)
• ...
Research:
• More publications.
Funds for Imaging and Computational Photography
IR Camera
Projector
• Millions Sold
Research:
• More publications.
Multimodal algorithms
• More Funds
Accumulation of the research in image retrieval, computer vision
• like.com, incogna.com
• like.com as acquired by Google for $100 Million
[1] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. Proceedings of the Robotics Science and Systems , 2006

Figure 7.1: The effects of the some advancements in technology in the last decade.

7.2 Key Observations

The following key observations can be highlighted after extensive experiments of the proposed algorithms in this thesis:

- Graph cut is powerful framework to incorporate multiple cues which can be different types, such low and high level observations, jointly. *Boundary* and *Regional* terms of graph cut can be set by using different types of the cues. For example, learning the color appearance of the human is not possible, because of the infinite color variations of the clothes. Therefore, the color is not incorporated into the point-wise human descriptor to refine the low dimensional shape in Chapter 5. However, *Boundary* term of graph cut provides the ability to fuse the color discontinuity to the proposed algorithm. Thus, the performance of the method increased. It enabled to classify some points which do not have valid depth data inside the body.

- On the other hand, the performance of the graph cut highly depends on the *quality* of the fore/background models to set the *Boundary* and *Regional* terms. In this context, *quality* implies how good the models represent the object and the background. Only possible way to boost the accuracy of the results to top of the quality of the models is selecting a good value to set the relative influence between the *Boundary* and *Regional* terms in graph cut. In some cases which graph cut formula includes multiple constant terms to set the weight of multiple cues or the terms, it becomes cumbersome to find a good combination values to set these constants. Similar situation was encountered in the experiments of the graph cut tracker described in Chapter 6.2.1. In that case, it was tried so hard to find a good combination for two constants of the graph cut to set the shape influence and pairwise interactions.

- It is investigated that fusing multiple cues obtained from different sensors increases the accuracy of the methods. Especially, multiple different types of sensors form a robust and complementary architecture. This type of construction allows utilizing different advantages of the sensors in the software level. For example, bringing geometry from the LIDAR and visual silhouette information from the color camera created a robust human classifier as described in Chapter 2. Also, fusing shape cue obtained from IR camera and color cue from the color camera in proper way increased the accuracy of the refinement and tracking methods in Chapter 5 and 6. Multimodal feature vectors do not affect only the accuracy of the method, but in some cases, such as in proposed human detection algorithm, it reduce the computational time. This opens a door creating real-time applications for the consumer products.

- Combining multiple different kinds of cues in one task require choosing a careful way, such as determining which cue is discriminative and representative in which part of the scene or how much it is important and incorporated for those regions into the system is a crucial observation to develop robust algorithms. In other words, setting the weights of the cues locally according to their dominance improves the accuracy and robustness of the methods. These methods can be either a refinement or tracking algorithm. In the proposed human shape refinement method explained in Chapter 5 and tracking method described in Chapter 6, basically the power of Random Forest machine learning technique if a smart local shape information is given with associated color and structure cues is investigated. Unlike SVMs or a *k-means* based fore/background model construction methods are not be able to discriminate or weight the importance of the cues dynamically and locally. An example of this case can be highlighted for the refinement of human shape. At the bottom level of the provided bounding box the normal of the points are much important than the depth cue. Or at the top level of the box, the depth of a point is more discriminative. SVMs or any kind of SVMs inherited machine learning method do not supply a way to learn or evaluate the characteristic of weighting cues dynamically within a feature vector. However, a technique which offers a tree structure based on evaluating the cues within a vector in a cascaded manner provides this ability.

7.3 Future Work and Limitations

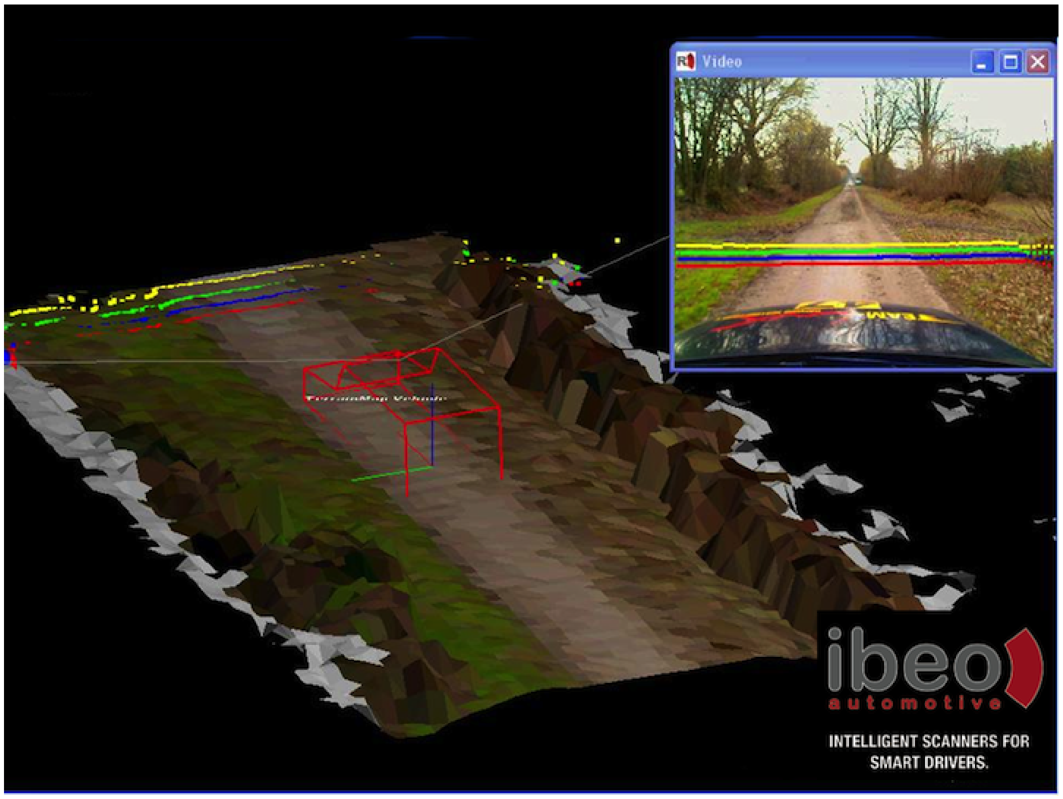
7.3.1 Multimodal Human Classifier

The proposed multimodal human detection algorithm can be extended for multilayer LIDARs. It is demonstrated that even one single layer structure information from the scene drastically improves the accuracy of the detection comparing to the single model algorithms. Multilayer LIDARs provide several layers of structure information. One of the is Ibeo Alasca XT which produces four vertical layers laser beams with an aperture angle of 3.3°. Another one is Velodyne HDL-64E which splits 64 layers of laser beams. The sample data of these two sensors can be seen in Figure 7.2. Some pedestrian detection algorithms which use these type of the sensors are proposed in [143] [53].

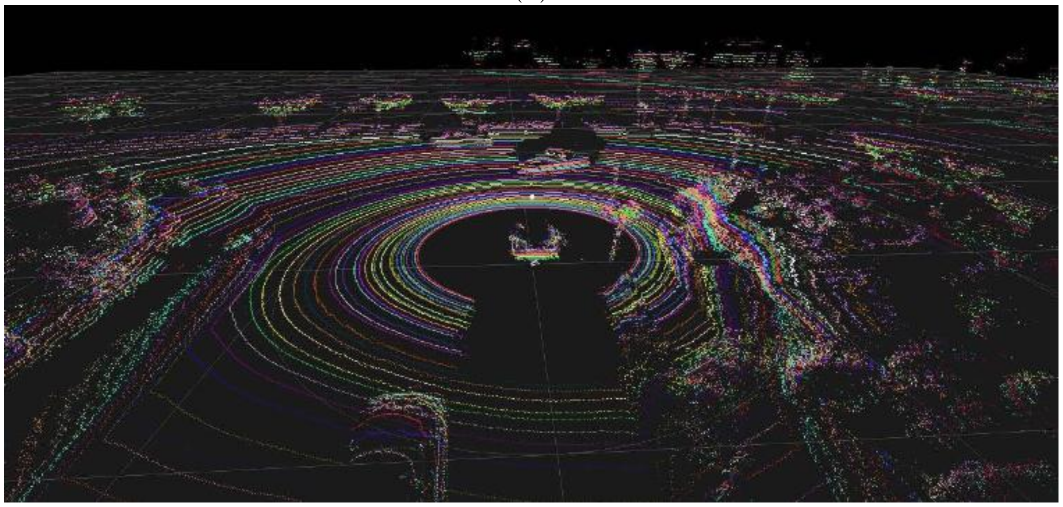
Utilizing from multiple layer geometric information can improve the detection rate of proposed multimodal tracker in Chapter 2. In the same way of extracting single layer geometric information from human body, several layers of these information can be combined in one proposed human descriptor. The association of geometric cues from multiple parts of the human body can form a better geometric descriptors. Or multiple human descriptors which includes only one layer of geometric cue can be formed. Each of these descriptors are asked to the human classifier at run time. The evaluations of each of these tests might be in a cascaded way, or by aggregating the scores of each test.

Proposed multimodal human detection algorithm is not developed to handle the partially occluded cases. Multilayer LIDAR sensor integration to the system can help to detect partially occluded cases. In occlusion scenarios, even if one layer of the LIDAR hits occluded part of the body, another layer might hit body points, so that this layer can detect the human correctly.

On the other hand, similar approach which relies on fusing one slice of geometric information from human body and visual cue in one feature vector can be extended to detect the human in indoor scenes by using depth and color images provided by Kinect. In this case, just as the artificial LIDAR data is obtained in the training process of the proposed classifier, multiple slices can be taken from the depth image to form the geometric descriptor. Since the computation of the proposed geometric descriptor is so fast, a fast and accurate human classifier can be developed for indoor scenes. In addition, the search space in the images can be constrained by testing random piece of depth slices taken at different orientations.



(a)



(b)

Figure 7.2: Appereance of multilayer LIDARs data. (a) shows the Ibeo Alasca XT data projected on to the color image and its visualized terrain map by a software taken from [69]. (b) displays the data of Velodyne HDL-64E [191].

7.3.2 Refinement and Recoginition with Graph Cuts

Graph cut method has been extensively used in the object segmentation, stereo matching, refinement and some tracking problems in the computer vision field. It demonstrates remarkable results in all of these vision tasks. However, it has not show up in the keypoint matching side of the object recognition. There are various ways to find the matches between the keypoints. Most of the methods collects some local information around the point and try to match the keypoints by considering the similarity between these local information. This process can be called as *one-to-one* keypoint matching. However, the problem can be converted to *n-to-n* keypoints matching task. In other words, matching n keypoints from one set to n keypoints from another set by incorporating pairwise relations or interactions.

This problem looks like labeling each keypoints from set-one to another keypoint from set-2 by considering pairwise relations in the system. For example, spatial proximity, especially in 3D point clouds, is an important pairwise cue which could improve the accuracy of the matches. Such a work which considers the spatial proximity between the keypoints is described in [155]. This work is not based on the graph cuts. It is possible to convert this problem to a graph cut structure and formulate it by two terms, *regional* and *boundary* in the graph cut formula. Spatial proximity can play a role to form the pairwise *boundary* penalty. The regional term can be weighted by the matching scores of the pairs in two different sets. The powerful and fast approximation method embedded in the graph cut might minimize this equation which represents the keypoint matches.

Furthermore, as it is outlined in Chapter 3, there is no existing work to set relative influence between *regional* and *boundary* terms of the graph cut by utilizing from machine learning techniques. Current approaches set this influence parameter without learning it by a training process. A generic way of setting this parameter can be developed by training a neural network type classifier for certain type of the image sets or scenes.

7.3.3 Low Dimensional Human Shape Refinement

In this thesis, the approaches to the *refinement* task do not consider obtaining some cues from the internals of method which provides the low dimensional shape. The purpose was to develop generic refinement methods independent of the algorithms which stay on the low dimensional space. This approach can be explained with an idiom, *"Take it, use it"*. It means that low dimensional shape is obtained from the provider, and used in the refinement method without touching back to the provider method. However, for specific types of the objects, such as a human, also some cues can be incorporated by going deep into the details of the low dimensional shape provider. For example, *Humanising GrabCut* [56] retrieves some features from HOG classifier to incorporate into its segmentation process. In a similar way, the proposed method can be extended for specific types of the underlying low dimensional shape providers. In addition to a bounding box output of the method, its descriptors for some points can be transferred to the refinement process in order to improve the accuracy. This type of work is left for the future.

The proposed low dimensional human refinement method does not have an ability to rectify the problematic outputs of the providers. If provided bounding box does not include a human, the human is outside of the box or partially inside the box, it can not refine all points of the human body. It only considers the human body points inside given bounding box. Same limitation also exists for *GrabCut* [157]. It segments the objects points just inside the bounding box. However, underlying method in our proposed classifier does not limit testing the points outside of given shape. This special issue can be addressed by removing virtual classification border which is the sides of the bounding box. All points in the scene can be classified by the method.

In addition to the estimated ground plane, other high level observations such as estimated walls, or detected objects can be incorporated into the multi-layer graph framework of the human refinement method. More interestingly, multiple bounding boxes which includes detected people are tried to be refined by the same method. This might be helpful to reduce false positive positives which originally belong to a second human.

7.3.4 Multimodal Complex Object Tracking

The proposed multimodal tracking method, $MM-Tracker$, in Chapter 7, does not include motion models specific to the objects. Therefore, it is sensitive to the local motions inside the objects. It has been observed that this limitation does not cause a big issue for tracking the complex rigid objects, such as a drill. However, local motions in some parts of deformable or articulated objects caused mistracking of some points where local motion exists at high range. $MM-Tracker$ is be able to capture those points back if the movement of the object is not so big. In these kinds of cases, $MM-Tracker$ works as an iterative refinement process at subsequent frames.

In order to make $MM-Tracker$ more robust to the local motions, one of the solution would be to employ motion models which are specific to the tracked objects. Another way can be to improve the SIFT based location estimation method. Instead of estimating the movement of the object averaging the displacement between the keypoints, a point-wise weighting method which considers the spatial distance to a keypoint can be developed. Basically, the displacement of the points between two frames are computed according to their distances to the keypoints. Or another approach could be to use an off the shelf dense optical flow method such as [44], but it would require more computational power and again to develop a way to handle the estimation of a group of the points to a single point in the next frame by [44]. This problem seem as an interesting one to work on it in the future.

On the other hand, fusing offline and online shape learning to $MM-Tracker$ can be interesting idea for specific types of the objects, such as a human. In Chapter 6, the shape of the human is learned by a classifier offline from a set of the images. This information can be incorporated while tracking a human online. Since the shape of the tracked human can be different than the learned generic human shape, developing a way to fuse two shape information is necessary. This process might require some changes at the deep level of learning algorithm.

BIBLIOGRAPHY

[1] Radhakrishna Achanta, Mohan Kankanhalli, and Philippe Mulhem. Compressed Domain Object Tracking for Automatic Indexing of Objects in MPEG Home Videos. In *IEEE International Conference on Multimedia and Expo (ICME) 2002*, pages 983–986, 2002.

[2] Lus A. Alexandre. 3d descriptors for object and category recognition: a comparative evaluation. In *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, 2012.

[3] Yali Amit and Donald Geman Y. Shape quantization and recognition with randomized trees. *Neural Computation*, 9:1545–1588, 1997.

[4] Kai O. Arras, Oscar Martinez Mozos, and Wolfram Burgard. Using boosted features for the detection of people in 2D range data. In *In IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2007.

[5] M.A. Balafar, A.R. Ramli, M.I. Saripan, and S. Mashohor. Review of brain mri image segmentation methods. *Artificial Intelligence Review*, 33:261–274, 2010.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *International Journal of Computer Vision and Image Understanding*, 110(3):346–359, jun 2008.

[7] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc J. Van Gool. Pedestrian detection at 100 frames per second. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'12)*, pages 2903–2910. IEEE, 2012.

[8] M. C. Best, T. J. Gordon, and P. J. Dixon. An Extended Adaptive Kalman Filter for Real-time State Estimation of Vehicle Handling Dynamics. *Vehicle System Dynamics*, 34(1):57–75, July 2000.

[9] A. Blake and M. Isard. The condensation algorithm - conditional density propagation and applications to visual tracking. In *Advances in Neural Information Processing Systems*, pages 36–1. The MIT Press, 1996.

[10] A. Blake, C. Rother, M. Brown, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. *European Int. Conf. on Computer Vision*, 2004.

[11] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm, 2000.

[12] E. Boyer, A. M. Bronstein, M. M. Bronstein, B. Bustos, T. Darom, R. Horaud, I. Hotz, Y. Keller, J. Keustermans, A. Kovnatsky, R. Litman, J. Reininghaus, I. Sipiran, D. Smeets, P. Suetens, D. Vandermeulen, A. Zaharescu, and V. Zobel. Shrec 2011: Robust feature detection and description benchmark. In *Proceedings of the 4th Eurographics Conference on 3D Object Retrieval*, pages 71–78, 2011.

[13] Y. Boykov and G. Funk-Lea. Graph cuts and efficient n-d image segmentation. *Int Journal of Computer Vision*, 70:109–131, 2006.

[14] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proceedings of the International Conference on Computer Vision*, 2001.

[15] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, September 2004.

[16] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *Proceedings of the International Conference on Computer Vision*, volume 1, 1999.

[17] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 2001.

[18] Yuri Boykov and Marie-Pierre Jolly. Interactive organ segmentation using graph cuts. In *Proceedings of the Third International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 276–286, 2000.

[19] Yuri Boykov and Marie-Pierre Jolly. Demonstration of segmentation with interactive graph cuts. In *ICCV'01*, 2001.

[20] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *Proceedings of the Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, EMMCVPR '01, pages 359–374, 2001.

[21] Yuri Boykov, Olga Veksler, and Ramin Zabih. A new algorithm for energy minimization with discontinuities. In *In International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 26–29, 1999.

[22] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct 2001.

[23] Bjrn Browatzki, Vadim Tikhanoff, Giorgio Metta, Heinrich H. Blthoff, and Christian Wallraven. Active object recognition on a humanoid robot. In *ICRA*, pages 2021–2028. IEEE, 2012.

[24] Aurlie Bugeau and Patrick Prez. Track and cut: Simultaneous tracking and segmentation of multiple objects with graph cuts. In *VISAPP (2)'08*, pages 447–454, 2008.

[25] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, pages 121–167, 1998.

[26] Sema Candemir and Yusuf Sinan Akgul. Adaptive regularization parameter for graph cut segmentation. In *ICIAR (1)'10*, pages 117–126, 2010.

[27] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[28] Ju Yong Chang, Kyoung Mu Lee, and Sang Uk Lee. Stereo matching using iterated graph cuts and mean shift filtering. In *ACCV (1)'06*, pages 31–40, 2006.

[29] Alan Y. Chen, Masayoshi Matsuoka, and Surya P. N. Singh. Autonomous helicopter tracking and localization using a self-surveying camera array. *The International Journal of Robotics Research*, 26, 2007.

[30] Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, (10):1252–1262.

[31] Dorin Comaniciu, Visvanathan Ramesh, Peter Meer, Senior Member, and Senior Member. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:564–577, 2003.

[32] Cash J. Costello, Christopher P. Diehl, Amit Banerjee, and Hesky Fisher. Scheduling an active camera to observe people. In *Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks*, VSSN '04, pages 39–45, 2004.

[33] Ana-Maria Cretu, Emil M. Petriu, Pierre Payeur, and Fouad F. Khalil. Deformable object segmentation and contour tracking in image sequences using unsupervised networks. In *Proceedings of the 2010 Canadian Conference on Computer and Robot Vision*, CRV '10, pages 277–284, 2010.

[34] Jacek Czyz, Branko Ristic, and Benoit Macq. A particle filter for joint detection and tracking of color objects. *Image Vision Computing*, 25:1271–1281, August 2007.

[35] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. Self-supervised monocular road detection in desert terrain. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[36] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 01*, CVPR '05, pages 886–893, 2005.

[37] Piali Das, Olga Veksler, Vyacheslav Zavadsky, and Yuri Boykov. Semiautomatic segmentation with compact shape prior. *Image Vision Computing*, pages 206–219, 2009.

[38] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2173–2180, 2010.

[39] Andrew Delong and Yuri Boykov. Globally optimal segmentation of multi-region objects. In *ICCV'09*, pages 285–292, 2009.

[40] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, (1):2173–2180, 2010.

[41] T. Dinh and G. Medioni. Two-frames accurate motion segmentation using tensor voting and graph-cuts. In *IEEE Workshop on Motion and Video Computing*, 2008.

[42] Thang Dinh, Qian Yu, and Gerard Medioni. Real time tracking using an active pan-tilt-zoom network camera. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'09, pages 3786–3793, 2009.

[43] W Eric, L Grimson, and Theo Pavlidis. Discontinuity detection for visual surface reconstruction. *Computer Vision Graphics and Image Processing*, 30(3):316–330, 1985.

[44] Gunnar Farneback. Two-frame motion estimation based on polynomial expansion. In *Image Analysis, 13th Scandinavian Conference*, volume 2749 of *Lecture Notes in Computer Science*, pages 363–370, 2003.

[45] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *Int Journal of Computer Vision*, 59(2), 2004.

[46] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Pictorial structures for object recognition. *Int. J. Comput. Vision*, 61(1):55–79, jan 2005.

[47] Silvio Filipe and Luís A. Alexandre. A comparative evaluation of 3d keypoint detectors. In *9th Conference on Telecommunications, Conftele 2013*, pages 145–148, Castelo Branco, Portugal, May 2013.

[48] National Center for Statistics and Analysis. Traffic safety facts annual reports, October 2012.

[49] Daniel Freedman and Petros Drineas. Energy minimization via graph cuts: Settling what is possible. In *CVPR (2)'05*, pages 939–946, 2005.

[50] Daniel Freedman and Tao Zhang. Interactive graph cut based segmentation with shape priors. In *CVPR (1)'05*, pages 755–762, 2005.

[51] Z. Garrett and H. Saito. Live video object tracking and segmentation using graph cuts. In *IEEE International Conference on Image Processing*, pages 1576–1579, 2008.

[52] Dariu Gavrila and Vasanth Philomin. Real-time object detection for smart vehicles. In *ICCV'99*, pages 87–93, 1999.

[53] Samuel Gidel, Paul Checchin, Christophe Blanc, Thierry Chateau, and Laurent Trassoudaine. Pedestrian detection and tracking in an urban environment using a multilayer laser scanner. *Transactions on Inteligent Transportation Systems*, 11(3):579–588, sep 2010.

[54] Martin Godec, Peter M. Roth, and Horst Bischof. Hough-based tracking of non-rigid objects. In *Proc. International Conference on Computer Vision*, 2011.

[55] Stephen Gould, Paul Baumstarck, Morgan Quigley, Andrew Y. Ng, and Daphne Koller. Integrating visual and range data for robotic object detection. In *ECCV workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications (M2SFA2)*, 2008.

[56] Varun Gulshan, Victor Lempitsky, and Andrew Zisserman. Humanising grabcut: Learning to segment humans using the kinect. In *Workshop on Consumer Depth Cameras in Computer Vision, ICCV 2011*, 2011.

[57] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[58] Antonio Hernndez-Vela, Miguel Reyes, Sergio Escalera, and Radeva Petia. Spatio-temporal grabcut human segmentation for face and pose recovery. In *Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition- Workshops*, pages 33 – 40, 2010.

[59] Antonio Hernndez-Vela, Miguel Reyes, Vctor Ponce, and Sergio Escalera. Grabcut-based human segmentation in video sequences. *Sensors*, 12(11):15376–15393, 2012.

[60] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[61] http:// darparoboticschallenge. org. Darpa robotics challenge website, February 2014.

[62] http://trace.eas.asu.edu/yuv/index.html. Video benchmarks, February 2014.

[63] http://www.thefreedictionary.com/complex. Definition of complex, November 2012.

[64] Chunhua Hu, Xudong Ma, Xianzhong Dai, and Kun Qian. Reliable people tracking approach for mobile robot in indoor environments. *Journal of Robotics and Computer-Integrated Manufacturing*, 26(2):174–179, April 2010.

[65] A. Huang, D. Moore, M. Antone, E. Olson, and S. Teller. Multi-sensor lane finding in urban road networks. In *Robotics: Science and Systems*, 2008.

[66] Chen Huang, Xiaoqing Ding, and Chi Fang. Pose robust face tracking by combining view-based aams and temporal filters. *Comput. Vis. Image Underst.*, 116(7):777–792, July 2012.

[67] Xiaolei Huang, Zhen Qian, Rui Huang, and Dimitris Metaxas. Deformable-model based textured object segmentation. In *Proceedings of the 5th international conference on Energy Minimization Methods in Computer Vision and Pattern Recognition*, EMMCVPR'05, pages 119–135, 2005.

[68] Nikolaos Kyriazis Iason Oikonomidis and Antonis Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *Proc. BMVC*, 2011.

[69] Ibeo. Appereance of ibeo alasca xt data. http://www.ibeo-as.com/ibeo_lux_81.html, Accessed January 2014.

[70] Sho Ikemura and Hironobu Fujiyoshi. Real-time human detection using relational depth similarity features. In Ron Kimmel, Reinhard Klette, and Akihiro Sugimoto, editors, *Computer Vision ACCV 2010*, volume 6495 of *Lecture Notes in Computer Science*, pages 25–38, 2011.

[71] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *Int. J. Comput. Vision*, 43(1):45–68, jun 2001.

[72] Michael Isard and Andrew Blake. Condensationconditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[73] V. Jagadeesh and B. S. Manjunath. Interactive graph cut segmentation of touching neuronal structures from electron micrographs. In *2010 Proceedings of 17th IEEE International Conference on Image Processing (icip 2010)*. IEEE, Jan 2010.

[74] O. Javed, Z. Rasheed, O. Alatas, and M. Shah. Knight/spl trade/: a real time surveillance system for multiple and non-overlapping cameras. In *Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 2*, ICME '03, pages 649–652, 2003.

[75] Omar Javed and Mubarak Shah. Tracking and object classification for automated surveillance. In *Proceedings of the 7th European Conference on Computer Vision-Part IV*, ECCV '02, pages 343–357, 2002.

[76] D. Jayadevappa, D. Murty, and S. Kumar. Medical image segmentation algorithms using deformable models: A review. *IETE Technical Review*, 28(3):248–255, 2011.

[77] Simon J. Julier, Jeffrey, and K. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE Review*, pages 401–422, 2004.

[78] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls*, pages 182–193, 1997.

[79] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[80] Bharath Kalyan, K. W. Lee, W. Sardha Wijesoma, D. Moratuwage, and Nicholas M. Patrikalakis. A random finite set based detection and tracking using 3d lidar in dynamic environments. In *SMC*, pages 2288–2292, 2010.

[81] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[82] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 506–513, 2004.

[83] Junhwan Kim, Vladimir Kolmogorov, and Ramin Zabih. Visual correspondence using energy minimization and mutual information. In *International Conference on Computer Vision*, pages 1033–1040, 2003.

[84] Taewan Kim, Sangho Cho, Jongmin Yoon, and Daijin Kim. Pose robust human detection in depth image using four directional 2d elliptical filters. In *Proceedings of the 2009 11th IEEE International Symposium on Multimedia*, ISM '09, pages 130–135. IEEE Computer Society, 2009.

[85] Jan Knopp, Mukta Prasad, Geert Willems, Radu Timofte, and Luc J. Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *ECCV*, volume 6316, pages 589–602, 2010.

[86] M. Kocamaz and C. Rasmussen. Automatic refinement of foreground regions for robot trail following. In *Proc. International Conference on Pattern Recognition*, 2010.

[87] M. K. Kocamaz, Y. Lu, and C. Rasmussen. Deformable object shape refinement and tracking using graph cuts and support vector machines. In *ISVC (2)'11*, pages 506–515, 2011.

[88] Mehmet Kemal Kocamaz and Fatih Porikli. Unconstrained 1d range and 2d image based human detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[89] Pushmeet Kohli, Jonathan Rihan, Matthieu Bray, and Philip H S Torr. Simultaneous segmentation and pose estimation of humans using dynamic graph cuts. *International Journal of Computer Vision*, 79:285–298, 2008.

[90] Pushmeet Kohli and Philip Torr. Efficiently solving dynamic markov random fields using graph cuts. *Tenth IEEE International Conference on Computer Vision ICCV05 Volume 1*, 2(1):922–929, 2005.

[91] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *Proc. Int. Conf. Computer Vision*, 2002.

[92] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *International Conference on Computer Vision*, pages 508–515, 2001.

[93] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *European Conference on Computer Vision*, pages 82–96, 2002.

[94] Vladimir Kolmogorov and Ramin Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:65–81, 2004.

[95] Vladimir Kolmogorov, Ramin Zabih, and Steven Gortler. Generalized multi-camera scene reconstruction using graph cuts. In *Proceedings of the International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 501–516, 2003.

[96] Vivek Kwatra, Arno Schdl, Irfan A. Essa, Greg Turk, and Aaron F. Bobick. Graphcut textures: image and video synthesis using graph cuts. *ACM Trans. Graph.*

[97] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Detection-based object labeling in 3d scenes. In *IEEE International Conference on on Robotics and Automation*, 2012.

[98] Jose-Luis Landabaso, Li-Qun Xu, and Montse Pardas. Robust tracking and object classification towards automated video surveillance. In *Image Analysis and Recognition*, volume 3212 of *Lecture Notes in Computer Science*, pages 463–470, 2004.

[99] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1265–1278, 2005.

[100] D Lee and T Pavlidis. One-dimensional regularization with discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):822–829, 1988.

[101] Vincent Lepetit, Pascal Lagger, and Pascal Fua. Randomized trees for real-time keypoint recognition. In *In CVPR*, pages 775–781, 2005.

[102] Lu Liu, David Raber, David Nopachai, Paul Commean, David Sinacore, Fred Prior, Robert Pless, and Tao Ju. Interactive separation of segmented bones in ct volumes using graph cut. In *Proceedings of the 11th international conference on Medical Image Computing and Computer-Assisted Intervention - Part I*, MICCAI '08, pages 296–304, 2008.

[103] Herve Lombaert, Yiyong Sun, Leo Grady, and Chenyang Xu. A multilevel banded graph cuts method for fast image segmentation. In *IEEE International Conference on Computer Vision*, pages 259–265, 2005.

[104] David G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[105] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[106] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, pages 674–679, 1981.

[107] M. Lucena, J.M. Fuertes, and N.Perez Blanca. Using optical flow for tracking. In *Progress in Pattern Recognition, Speech and Image Analysis*, volume 2905 of *Lecture Notes in Computer Science*, pages 87–94, 2003.

[108] Simon Lucey, Yang Wang, Jason Saragih, and Jeffery F. Cohn. Non-rigid face tracking with enforced convexity and local appearance consistency constraint. *Image Vision Comput.*, 28(5):781–789, May 2010.

[109] J. Malcolm, Y. Rathi, and A. Tannenbaum. Tracking through clutter using graph cuts. In *Proc. British Machine Vision Conference*, 2007.

[110] James Malcolm, Yogesh Rathi, and Allen Tannenbaum. Multi-object tracking through clutter using graph cuts. In *Non-Rigid Registration and Tracking Through Learning (in ICCV)*, 2007.

[111] James G. Malcolm, Yogesh Rathi, and Allen Tannenbaum. Graph cut segmentation with nonlinear shape priors. In *ICIP (4)'07*, pages 365–368, 2007.

[112] P. Mehrani and O. Veksler. Saliency segmentation based on learning and graph cut refinement. In *Proc. British Machine Vision Conference*, 2010.

[113] N. Metropolis and S.Ulaml. The monte carlo method. *Journal of the American Statistical Association*, pages 335–341, 1949.

[114] A. Mian, M. Bennamoun, and R. Owens. On the repeatability and quality of keypoints for local feature-based 3d object retrieval from cluttered scenes. *International Journal of Computer Vision*, 89(2-3):348–361, 2010.

[115] Cyrille Migniot, Pascal Bertolino, and Jean-Marc Chassery. Iterative human segmentation from detection windows using contour segment analysis. In *VISAPP*, pages 405 – 412, 2013.

[116] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Multiple object class detection with a generative model. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, pages 26–36, 2006.

[117] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision*, pages 525–531, 2001.

[118] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005.

[119] Krystian Mikolajczyk, Cordelia Schmid, and Andrew Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV'04*, pages 69–82, 2004.

[120] R. Minetto, T. V. Spina, A. X. Falcão, N. J. Leite, J. P. Papa, and J. Stolfi. Iftrace: Video segmentation of deformable objects using the image foresting transform. *Comput. Vis. Image Underst.*, 116(2):274–291, feb 2012.

[121] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example based object detection in images by components. *IEEE Trans. Pattern Anal. and Machine Intell*, 23:349–361, 2001.

[122] J. Mooser, S. You, and U. Neumann. Real-time object tracking for augmented reality combining graph cuts and optical flow. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 145–152, 2007.

[123] Frank Moosmann, Bill Triggs, and Frederic Jurie. Fast discriminative visual codebooks using randomized clustering forests. In *In NIPS*, 2007.

[124] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application*, pages 331–340. INSTICC Press, 2009.

[125] M. Jgersand A. Murtha T. Kesztyues N. Birkbeck, D. Cobzas. An interactive graph cut method for brain tumor segmentation. In *IEEE Workshop on Applications of Computer Vision (WACV 2009)*, 2009.

[126] Tomoyuki Nagahashi, Hironobu Fujiyoshi, and Takeo Kanade. Image segmentation using iterated graph cuts based on multi-scale smoothing. In *ACCV (2)'07*, pages 806–816, 2007.

[127] Tomoyuki Nagahashi, Hironobu Fujiyoshi, and Takeo Kanade. Video segmentation using iterated graph cuts based on spatio-temporal volumes. In *ACCV (2)'09*, pages 655–666, 2009.

[128] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Computer Vision Conference (ECCV)*, 2012.

[129] Luis Ernesto Navarro-Serment, Christoph Mertz, Nicolas Vandapel, and Martial Hebert. Ladar-based pedestrian detection and tracking. In *Proc. 1st. Workshop on Human Detection from Mobile Robot Platforms, IEEE ICRA 2008*. IEEE, May 2008.

[130] A. Nelson and J. Neubert. Object tracking via graph cuts. In *SPIE Applications of Digital Image Processing*, 2009.

[131] Viet-Anh Nguyen and Yap-Peng Tan. Efficient block-matching motion estimation based on integral frame attributes. *IEEE Transactions on Circuit and Systems for Video Technology*, 16(3):375–385, March 2006.

[132] Tatyana Nuzhnaya, Erkang Cheng, Haibin Ling, Despina Kontos, Predrag R. Bakic, and Vasileios Megalooikonomou. Segmentation of anatomical branching structures based on texture features and graph cut. In *ISBI'11*, pages 673–676, 2011.

[133] Andreas Opelt, Axel Pinz, and Andrew Zisserman. Incremental learning of object detectors using a visual shape alphabet. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 3–10, 2006.

[134] Caroline Pantofaru, Leila Takayama, Tully Foote, and Bianca Soto. Exploring the role of robots in home organization. In *Proc. of Human-Robot Interaction*, pages 327–334, 2012.

[135] N. Papadakis and A. Bugeau. Tracking with occlusions via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 33(1):144–157, January 2011.

[136] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *Int. J. Comput. Vision*, 38(1):15–33, June 2000.

[137] Bo Peng and Olga Veksler. Parameter selection for graph cut based image segmentation. In *BMVC'08*, 2008.

[138] Bo Peng, Lei Zhang, and Jian Yang. Iterated graph cuts for image segmentation. In *ACCV (2)'09*, pages 677–686, 2009.

[139] Stephen M. Pizer, Sarang C. Joshi, P. Thomas Fletcher, Martin Styner, Gregg Tracton, and James Z. Chen. Segmentation of single-figure objects by deformable m-reps. In *Proceedings of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, MICCAI '01, pages 862–871, 2001.

[140] Christian Plagemann, Varun Ganapathi, Daphne Koller, and Sebastian Thrun. Real-time identification and localization of body parts from depth images. In *In IEEE Int. Conf. on Rob. & Autom. (ICRA)*, pages 3108–3113, may 2010.

[141] D. Pomerleau. RALPH: Rapidly adapting lateral position handler. *Proc. IEEE Intelligent Vehicles Symposium*, pages 506–511, 1995.

[142] Christiano Premebida, Oswaldo Ludwig, and Urbano Nunes. Exploting lidar-based features on pedestrian detection in urban scenarios. In *Conference on Intelligent Transportation Systems*, 2009.

[143] Cristiano Premebida, Oswaldo Ludwig, and Urbano Nunes. Lidar and vision-based pedestrian detection system. *J. Field Robot.*, 26(9):696–711, September 2009.

[144] Brian L Price, Bryan Morse, and Scott Cohen. Geodesic graph cut for interactive image segmentation. *Childhood A Global Journal Of Child Research*, pages 3161–3168, 2010.

[145] J. R. Quinlan. Induction of decision trees. *Mach. Learn*, pages 81–106, 1986.

[146] J. M. Keller R. H. Luke, D. Anderson and M. Skubic. Human segmentation from video in indoor environments using fused color and texture features. *Technical Report, Electrical and Computer Engineering Department, University of Missouri*, 2008.

[147] Petia Radeva. Graph cuts optimization for multi-limb human segmentation in depth maps. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 726–732, 2012.

[148] Deva Ramanan, David A. Forsyth, and Andrew Zisserman. Tracking people by learning their appearance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(1):65–81, jan 2007.

[149] C. Rasmussen, Y. Lu, and M. Kocamaz. Appearance contrast for fast, robust trail-following. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2009.

[150] C. Rasmussen, Y. Lu, and M. Kocamaz. Trail following with omnidirectional vision. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2010.

[151] C. Rasmussen, Y. Lu, and M. Kocamaz. Integrating stereo structure for omnidirectional trail following. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 2011.

[152] C. Rasmussen and D. Scott. Shape-guided superpixel grouping for trail detection and tracking. *Proc. Int. Conf. Intelligent Robots and Systems*, 2008.

[153] Christopher Rasmussen, Yan Lu, and Mehmet Kocamaz. A trail-following robot which uses appearance and structural cues. In *International Conference on Field and Service Robotics*, pages 265–279, 2012.

[154] Christopher Rasmussen, Yan Lu, and Mehmet Kocamaz. A trail-following robot which uses appearance and structural cues. In *Field and Service Robotics*, volume 92 of *Springer Tracts in Advanced Robotics*. Springer Berlin Heidelberg, 2014.

[155] J. Revaud, G. Lavoue, Y. Ariki, and A. Baskurt. Learning an efficient and robust graph matching procedure for specific object recognition. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 754–757, Aug 2010.

[156] Remi Ronfard, Cordelia Schmid, and Bill Triggs. Learning to parse pictures of people. In *In European Conference on Computer Vision*, pages 700–714, 2002.

[157] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23:309–314, 2004.

[158] Samuele Salti, Federico Tombari, and Luigi Di Stefano. A performance evaluation of 3d keypoint detectors. In *Proceedings of the 2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 236–243, 2011.

[159] S. Sclaroff and L. Liu. Deformable shape detection and description via model-based region grouping. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(5), 2001.

[160] David Serby and Luc Van Gool. Probabilistic object tracking using multiple features. In *In IEEE International Conference of Pattern Recognition (ICPR)*, pages 184–187, 2004.

[161] Vinay Sharma and James W. Davis. Integrating appearance and motion cues for simultaneous detection and segmentation of pedestrians. In *ICCV'07*, pages 1–8, 2007.

[162] K. Shen and E. J. Delp. A fast algorithm for video parsing using mpeg compressed sequences. In *Proceedings of the 1995 International Conference on Image Processing - Volume 2*, ICIP '95, pages 2252–, 1995.

[163] B. A. Shepherd. An appraisal of a decision tree approach to image classification. In *in Proc. International Joint Conference on Artificial Intelligence*, pages 473–475, 1983.

[164] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593 – 600, 1994.

[165] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*, 2011.

[166] Jamie Shotton, Ross Girshick, Andrew Fitzgibbon, Toby Sharp, Mat Cook, Mark Finocchio, Richard Moore, Pushmeet Kohli, Antonio Criminisi, Alex Kipman, and Andrew Blake. Efficient human pose estimation from single depth images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[167] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[168] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, 2011.

[169] Amira Soudani and Ezzeddine Zagrouba. People tracking based on predictions and graph-cuts segmentation. In *Advances in Visual Computing*, volume 8034 of *Lecture Notes in Computer Science*, pages 158–167, 2013.

[170] B. Southall and C. Taylor. Stochastic road shape estimation. In *Proceedings of the International Conference on Computer Vision*, pages 205–212, 2001.

[171] Luciano Spinello, Kai Oliver Arras, Rudolph Triebel, and Roland Siegwart. A layered approach to people detection in 3d range data. In *AAAI*, 2010.

[172] Luciano Spinello and Roland Siegwart. Human detection using multimodal and multidimensional features. In *In IEEE Int. Conf. on Rob. & Autom. (ICRA)*, 2008.

[173] Johannes Strom, Andrew Richardson, and Edwin Olson. Graph-based segmentation for colored 3D laser point clouds. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[174] Yanchao Su, Haizhou Ai, Takayoshi Yamashita, and Shihong Lao. Human pose estimation using exemplars and part based refinement. In *Proceedings of the 10th Asian conference on Computer vision - Volume Part II*, ACCV 2010, pages 174–185, 2010.

[175] Akira Suga, Keita Fukuda, Tetsuya Takiguchi, and Yasuo Ariki. Object recognition and segmentation using sift and graph cuts. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[176] Kah Kay Sung and Tomaso Poggio. Example based learning for view-based human face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:39–51, 1995.

[177] V. Torre T. Poggio and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985.

[178] Zhang Tao, Yang Diange, Li Ting, and Lian Xiaomin. Vehicle state estimation system aided by inertial sensors in gps navigation. In *Proceedings of the 2010 International Conference on Electrical and Control Engineering*, ICECE '10, pages 5793–5796, 2010.

[179] C. Taylor, J. Malik, and J. Weber. A real-time approach to stereopsis and lane-finding. In *Proc. IEEE Intelligent Vehicles Symposium*, 1996.

[180] Demetri Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Trans. Pattern Anal. Mach. Intell.*, pages 413–424, 1986.

[181] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[182] Sebastian Thrun, Maren Bennewitz, Wolfram Burgard, Armin B. Cremers, Frank Dellaert, Dieter Fox, Dirk Hhnel, Charles Rosenberg, Nicholas Roy, Jamieson Schulte, and Dirk Schulz. Minerva: A second-generation museum tour-guide robot. In *In Proceedings of IEEE International Conference on Robotics and Automation*, 1999.

[183] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *Proceedings of the 11th European Conference on Computer Vision Conference on Computer Vision: Part III*, pages 356–369, 2010.

[184] B. Ugur Toreyin, Yigithan Dedeoglu, Ugur Gudukbay, and A. Enis Cetin. Computer vision based method for real-time fire and flame detection. *Pattern Recognition Letters*, 27(1):49 – 58, 2006.

[185] O. Tuzel, F. Porikli, and P. Meer. Human Detection via Classification on Riemannian Manifolds. In *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, pages 1–8, 2007.

[186] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region covariance: a fast descriptor for detection and classification. In *Proceedings of the 9th European conference on Computer Vision - Volume Part II*, ECCV'06, pages 589–600, 2006.

[187] C. Urmson and R. Whittaker. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8), 2008.

[188] Olga Veksler. Graph cut based optimization for mrfs with truncated convex priors. In *CVPR'07*, 2007.

[189] Olga Veksler. Star shape prior for graph-cut image segmentation. In *European Conference on Computer Vision*, volume 5304, pages 454–467. Springer, 2008.

[190] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Perspectives in neural computing'10*, pages 211–224, 2010.

[191] Velodyne. Appereance of velodyne hdl-64e data. `http://velodynelidar.com/lidar/products/white_paper/HDL%20white%20paper_OCT2007_web.pdf`, Accessed January 2014.

[192] Lubor Ladicky Vibhav Vineet, Jonathan Warrell and Philip Torr. Human instance segmentation from video using detector-based conditional random fields. In *Proceedings of the British Machine Vision Conference*, pages 80.1–80.11, 2011.

[193] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Graph cut based image segmentation with connectivity priors. In *CVPR'08*, 2008.

[194] Vibhav Vineet and P. J. Narayanan. Cuda cuts: Fast graph cuts on the gpu. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.

[195] Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[196] G. Vogiatzis, P. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 391–398, 2005.

[197] Nhat Vu and B. S. Manjunath. Shape prior segmentation of multiple objects with graph cuts. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008.

[198] Jue Wang. Discriminative gaussian mixtures for interactive image segmentation. In *ICASSP (1)'07*, pages 601–604, 2007.

[199] Robert Y. Wang and Jovan Popović. Real-time hand-tracking with a color glove. In *ACM SIGGRAPH 2009*, SIGGRAPH '09, pages 63:1–63:8, 2009.

[200] Yichen Wei, Jian Sun, Xiaoou Tang, and Heung yeung Shum. Interactive offline tracking for color objects. In *IEEE International Conference on Computer Vision*, pages 1–9, 2007.

[201] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J.B. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *IEEE Intl. Conf. on Robotics and Automation, ICRA*, May 2013.

[202] Bo Wu and Ram Nevatia. Simultaneous object detection and segmentation by boosting local shape feature based classifier. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, 2007.

[203] Ning Xu, Ravi Bansal, and Narendra Ahuja. Object segmentation using graph cuts based active contours. In *Computer Vision and Pattern Recognition*, volume 2, page 46, 2003.

[204] Jing Yang and James S. Duncan. 3d image segmentation of deformable objects with joint shape-intensity prior models using level sets. *Medical Image Analysis*, 8(3):285 – 294, 2004.

[205] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38, December 2006.

[206] Alper Yilmaz, Xin Li, and Mubarak Shah. Contour based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1531–1536, 2004.

[207] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *International Conference on Computer Vision and Pattern Recognition*, pages 373–380. IEEE, 2009.

[208] Jian Zhao and S Cheung Sen-ching. Human segmentation by geometrically fusing visible-light and thermal imageries. In *In IEEE International Conference of Computer Vision- Workshops*, pages 1185 – 1192, 2009.

[209] Jian Zhao and S Cheung Sen-ching. Human segmentation by geometrically fusing visible-light and thermal imageries. *Multimedia Tools and Applications*, pages 1–29, 2012.

[210] T. Zhao and R. Nevatia. Bayesian human segmentation in crowded situations. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 406–413, 2003.

[211] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *International Conference on Computer Vision, Workshop on 3DRR*, pages 689–696, Sep 2009.

[212] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 1491–1498, 2006.

[213] Z. Zivkovic and B. Krose. Part based people detection using 2D range data and images. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2007.

Appendix A
IMAGE MODIFICATIONS

In order to comply with the PhD Thesis requirements of University of Delaware, the faces of the people are blurred in some images. This has been done using the smudge tool of open source GIMP (version 2.82) image editor. Please note that these modifications were made on the result images of the proposed algorithms in this thesis. The proposed algorithms take the original images as the input without modifications.

Appendix B

PERMISSIONS

The following permissions are for using some images of the people in this thesis. They appear in Figure 1.1, 4.5, 6.1, 6.3, 6.11, 6.12, 6.13.

Google apps @UDel.edu

Mehmet Kocamaz <kocamaz@udel.edu>

About My image

Abdullah Kaplan <kaplana@udel.edu> Fri, Apr 4, 2014 at 3:51 PM
To: kocamaz@udel.edu

Hello Mehmet,
I hereby grant you permission to use my images and videos in your dissertation and research publications.
Abdullah Kaplan

Google apps @UDel.edu

Mehmet Kocamaz <kocamaz@udel.edu>

Video/Image permission

2 messages

Mehmet Kocamaz <kocamaz@udel.edu> Fri, Apr 11, 2014 at 4:22 PM
To: reisslein@asu.edu

Hello Professor Martin Reisslein,
I am in the process of submitting my dissertation to the grad office and they require a written approval for using videos/images from one of your YUV Video Sequences in my dissertation for the research purpose.
I would be glad if you could send this as an e-mail, which I will include in my dissertation.
Thank you.

Best Wishes,

-- Mehmet Kocamaz
University of Delaware
Dynamic Vision Lab

Martin Reisslein <reisslein@asu.edu> Sat, Apr 12, 2014 at 10:03 AM
To: Mehmet Kocamaz <kocamaz@udel.edu>

Hi Mehmet,

sure, you have my permission. These sequences are openly available for research.

You may also find interesting, that we have extensive tutorials on video traces that employ these and longer sequences, see

http://mre.faculty.asu.edu/ModEnc.pdf

http://mre.faculty.asu.edu/H264VidTraceTut.pdf

Martin

--
Martin Reisslein

Professor

School of Electrical, Computer,

and Energy Engineering
Arizona State University
Goldwater Center, MC 5706

Google apps @UDel.edu

Mehmet Kocamaz <kocamaz@udel.edu>

Permission

Alparslan SARI <asari@udel.edu> Sun, Apr 6, 2014 at 10:58 PM
Reply-To: asari@udel.edu
To: Mehmet Kocamaz <kocamaz@udel.edu>

Dear Mehmet,

I hereby grant you permission to use my images and videos in your dissertation and research publications.

Sincerely yours,
Alparslan Sari

--
Alparslan SARI
M.Sc, Computer and Information Science'08
M.Sc, Bioinfomatics and Computational Science, Concentration in Computational Sciences'12
M.Eng, Software Engineering'13
University of Delaware
Newark,DE

http://www.cis.udel.edu/~asari/

Google apps @UDel.edu

Mehmet Kocamaz <kocamaz@udel.edu>

permission

Tuna Demir <tunad@udel.edu> Mon, Apr 7, 2014 at 2:27 PM
To: kocamaz@udel.edu

Hello Mehmet,

I hereby grant you permission to use my images and videos in your dissertation and research publication.

Tuna Demir

Google apps @UDel.edu

Mehmet Kocamaz <kocamaz@udel.edu>

Permission for my picture

Cuneyt Kilicaslan <ck1704@nyu.edu> Tue, Apr 8, 2014 at 2:58 PM
To: kocamaz@udel.edu

Hello Mehmet, I hereby grant you permission to use my images and videos in your dissertation and research publications. -- Cuneyt Kilicaslan