Visual Tracking with an Application to Augmented Reality

DISSERTATION

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Changlin Xiao

Graduate Program in Civil Engineering

The Ohio State University

2017

Dissertation Committee:

Dr. Alper Yilmaz, Advisor

Dr. Charles Toth

Dr. Rongjun Qin
Abstract

In simple terms, visual tracking is the process of visually following a given object. This very initial task is one of the fundamental problems in many computer vision applications, like in movement pattern analysis, animal surveillance, robot navigation, and so on. Currently, with the increasing popularity of cameras, large video data are generated every day. However, the algorithms that used to handle this visual information are far from being enough. Besides, the technical progress and increasing demands for unmanned aerial vehicle (UAV) with the automatic pilot capability promote the requirement of visual tracking in many practical applications. Therefore, visual tracking is still one of the most interesting topics in computer vision tasks.

However, even various methods have been proposed for visual tracking, it is still an open problem due to its complexity. Tracking means following the target’s motion, and the primary challenge in visual tracking is the inconsistency of target’s appearance. The status of the target may be changed with the illumination change, the non-rigid motion, and occlusions. Additionally, the similar background may cause the drift problem like switching the targeted player to the untargeted ones in the games. In real life, there are more problems such as the scale change, fast motion, lower resolution and out of plane rotation which cause the tracking tasks even more challenging. Therefore, visual tracking, after several decades’ research, is still an active research topic with many unsolved problems.
In this dissertation, three tracking methods are proposed trying to deal with the tracking problems for different targets and various scenarios. Also, besides the tracking in 2D images, this work further introduces a 3D space tracking model in augmented reality application.

For a simple tracking scenario, an efficient tracking method with distinctive color and silhouette is proposed. The proposed method uses colors that most exist on the target to represent and track it. It is a dynamic color representation for the target which is updating with the background changes. This appearance model can substantially reduce the distractors in the background, and the color is constant to the shape change which significantly alleviates the nonrigid deformation problem.

Based on the above tracking idea, a unique feature vote tracking algorithm is further developed. This work divides the feature space into many small spaces as storage cells for feature descriptions. And if most of the descriptions in the cell are from the target, the features in the cell are treated as unique features. Besides counting how likely the feature from the target, each feature’s location respect to the target center is recorded to reproject the center in the new coming frames. This voting machine makes the tracker focus on the target against the occlusion and cluster background.

Recently, deep learning and neural network show powerful ability in computer vision applications. The neural network, especially the convolutional neural network has been successfully used in object recognition, detection, and tracking. This work uses a pre-trained network that has learned high-level semantic features to represent the target as a
concept model. This concept is a combination of these high-level features that learned from myriads of objects. With the concept, the network can generate a hot map of the new coming frame that shows the possible distribution of the target. Finally, a Siamese network is used to locate the target location. The high-level semantic features are robust to general appearance changes and can retrieve the target in many complicated scenarios.

Besides 2D tracking, 3D space tracking is more useful in many applications. To demonstrate that, this work uses a stereo camera to form a space tracking system for the surgical scalpel. Three LED lights are attached to the top of the scalpel to help the tracking of its tip. After the registration between cameras, operation table, and the augmented reality device, the scalpel’s motion can be displayed in the space by the augmented reality device. This holographic display is advantageous in the surgical operation for education and navigation. The localization experiment also shows the accuracy of the 3D space tracking.

In summary, these research efforts have offered different methods for visual tracking at various scenarios. It also demonstrates the visual tracking’s usefulness in practical application that both in 2D and 3D space.
Acknowledgments

It’s an honor to study at the Ohio State University and these years are the most precious time of my life. I would like to thank my Ph.D. advisor, Dr. Alper Yilmaz. He is the most important and kindly person in my academic life. He is not just an advisor but also a mentor, not only shows me how to do research but also shows me how to think about the world. Also, my thanks go to Dr. Rongxing Li who showed me the attitude to learning and work. I am also fortunate to learn from many excellent professors like Dr. Alan Saalfeld, Dr. Jekeli, Dr. Toth and Dr. Qin. Besides, thanks to Dr. Young Jin Lee, Dr. Siavash HosseinyAlamdary, Shirui Li, Nima Ajam Gard, Yujia Zhang, M. Taha Koroglu, and Michael Karnes, the many peers who we had many discussions and happy times. Finally, I would like to thank my family, the source of power and motivation for my Ph.D. study.
Vita

2009.......................................................... B.S. of Measuring, Testing Technologies and Instruments, University of Electronic Science & Technology of China

2012.......................................................... M.E. of Measurement Technologies and Instruments, University of Electronic Science & Technology of China

2012 to present ........................................... Graduate Student, Department of Civil, Environmental, and Geodetic Engineering, The Ohio State University

Publications


* equal contribution


Fields of Study

Major Field: Civil Engineering

Geodetic and Geoinformation Engineering
Table of Contents

Abstract ................................................................................................................................. ii

Acknowledgments ................................................................................................................ v

Vita ........................................................................................................................................ vi

Table of Contents ................................................................................................................. viii

List of Tables ......................................................................................................................... xiii

List of Figures ....................................................................................................................... xiv

Chapter 1: Visual Tracking ................................................................................................. 1

1.1 Motivation ...................................................................................................................... 2

1.2 Research Scope ............................................................................................................ 3

1.3 Contribution .................................................................................................................. 4

1.4 Dissertation Structure ................................................................................................. 5

Chapter 2: Related Work .................................................................................................... 7

2.1 Overview of Visual Tracking ....................................................................................... 7

2.2 Visual Tracking in Images .......................................................................................... 9

2.2.1 Generative Tracking with Appearance Representation ......................................... 9
2.2.2 Tracking with Discriminative Information ........................................... 12
2.2.3 Tracking Based on Neural Network ...................................................... 14
2.3 Visual Tracking in Space ........................................................................ 15

Chapter 3: Background Knowledge .............................................................. 18
3.1 Image Feature ......................................................................................... 18
3.1.1 Feature Point Detection ..................................................................... 18
3.1.2 Local Feature Descriptors ................................................................. 25
3.1.3 The Application in Image 3d Reconstruction ..................................... 29
3.2 Classifiers ............................................................................................. 30
3.3 Artificial Neural Networks ..................................................................... 38
3.3.1 The Structure of Neural Network ...................................................... 40
3.3.2 Convolutional Neural Network (CNN) ............................................... 41
3.4 Camera Geometry .................................................................................. 45
3.4.1 Camera Calibration: ......................................................................... 45
3.4.2 Bundle Adjustment ............................................................................ 49
3.4.3 Epipolar Geometry ............................................................................ 52

Chapter 4: Efficient Tracking with Distinctive Target Colors and Silhouette .... 55
4.1 Introduction ............................................................................................ 56
4.2 Methodology .......................................................................................... 58
List of Tables

Table 3.1. The performance of different point detectors. The data is from [79]. ............ 24
Table 3.2. The performance of SIFT, PCA-SIFT, SURF. The data is from [82]. ............ 27
Table 4.1. The center location error. (red as best, blue as second). ......................... 73
Table 4.2. The successful tracking ratio. (red as best, blue as second). .................... 74
Table 4.3. The proposed algorithm VS pixel track for tracking ratio. ....................... 75
Table 5.1 the track-ability score of each image sequence in OTB50 [123] ............... 90
Table 7.1. The average absolute errors of lights to the fitting lines. A is the angle of the scalpel, P.A is the different positions’ average errors, L.A. is the different lights’ average errors. The values are in mm. ........................................... 148
Table 7.2. The standard deviation of errors that lights locations to the fitting lines. A is the angle of the scalpel, P.A is the different positions’ average errors, L.A. is the different lights’ average errors. All the values are in mm. ......................... 149
List of Figures

Figure 3.1. The illumination of the FAST detector. The image is from [75]. ................. 21

Figure 3.2. The Laplacian-of-Gaussian (LoG) scale selection filter. The filter response is strongest for circular image structures whose radius corresponds to the filter scale. The image is from [72]. ......................................................... 22

Figure 3.3. The SIFT descriptor. For each scale invariant region, first do the orientation normalization, then project to a 16x16 pixel regular grid. Divide the grid to a 4x4 sub-grid and compute the gradient orientation histograms for each sub-grid. Finally, add all sub-grid gradient orientation histograms together to generate a 128-dimensional feature vector. The image is from [80]. .......... 26

Figure 3.4. Neural network and convolution neural network. The image is from O'Reilly Strata blog [143]. .......................................................... 34

Figure. 3.5. A hyperplane used to separate the two data sets. The image is from [135]. . 37

Figure. 3.6. The illustration of the mathematical model of the working of brain cell. The image is from [142]. Image (a) is the biological neuron model of brain cell. Image (b) is an artificial neuron mathematical model of the brain cell. ....... 39

Figure. 3.7. The layer structure of artificial neural network................................. 40

Figure. 3.8. The structure of CNN and its features. The image is from [136]............. 42
Figure 3.9. Convolutional operation of CNN. Images are from [138] and [142]. Image (a) is convolution steps for a filter to go through the whole image. Image (b) is the detailed convolutional operation of a filter at one position.

Figure 3.10. The typical ConvNet architecture.

Figure 3.11. The pinhole camera projection model.

Figure 3.12. The illustration of lens distortion. The image is from [135].

Figure 3.13. The illustration of epipolar geometry for stereo camera.

Figure 4.1. Illustration of the tracking flow. The dashed box represents the initialization of the color model from the target rectangle (red) and its surrounding area (the whole image). The histogram based mode represents the color target. The lower row illustrates the tracking in new frames and updating the color model.

Figure 4.2. Illustration of the distinctive target color. (a) shows the high trackability with many distinctive target colors and (b) shows the low track-ability with little distinctive target colors in HSV color space.

Figure 4.3. The tracking and segmentation. (a) is the search image in a new-comings frame with the tracked target rectangle. (b) is the short-time confidence map used to find target position. (c) is the long-time confidence map used to find the target regions. (d) is the segmentation results of the target. (e) shows the segmentation used to mask the target in the updating color models.

Figure 4.4. Illustration of evaluation of tracking result and detection the target again. (a) Tracking results treat as part detected. (b) The lose of tracking. (c) The
detection of the target after it been lost several times. Upper left blue box
inside (c) is a zoom image of the smaller rectangle in the middle. (d) Detected
the target again with long-time and short-time model. ........................................ 65

Figure 4.5. Tracking in cluttered background............................................................ 67

Figure 4.6. Tracking a non-rigidly deforming object. .............................................. 68

Figure 4.7. Tracking with occlusion................................................................. 70

Figure 4.8. Tracking with illumination change............................................... 71

Figure 4.9. Tracking comparison................................................................. 72

Figure 5.1. The flow diagram of the proposed tracking framework. For each new frame,
we extract descriptors for features within the search window (white rectangle)
generated based on past target location. The mapped cells and their likelihood
measures are used to generate a target center voting map and an appearance
confidence map (red circle means the feature from the target, the blue dot
means the feature from the background). Finally, the descriptors within the
tracked bounding box are used to update the cells. ........................................ 78

Figure 5.2. The cells in the descriptor space are generated based on the labeled pixels
within the target bounding box (smaller white box), and its surrounding
(larger white box). The descriptors generated for each labeled pixel (colored
circles inside the boxes) are used to divide the descriptor space into cells to
either represent target or background, or neither of them................................. 80

Figure 5.3. The spatial codebook kept for each cell in the unique target representation.
The left figure illustrates a 2D descriptor space. The middle figure represents
a number of color coded descriptors computed for target pixels. Their mapping to the feature space introduces an entry for the spatial codebook. For a new frame shown on the right, we estimate the corresponding descriptors and back project the code book to generate votes for the target center.

Figure 5.4. Illustration of the target search based on estimated target center on confidence map. The yellow circle denotes the estimated target center.

Figure 5.5. Illumination change is causing tracking failure. The trackability score at the beginning of the sequence is 0.263 and is close to 0.2. This causes confusion in the following frames and results in target loss.

Figure 5.6. The tracking results from (OTB).

Figure 5.7. Scale changes can cause inaccuracy in scale estimation. The first row shows a successful scenario where the scale of the tracked target remains constant. The second row shows limited performance due to background clutter masquerading scale change. In both cases, the yellow dot represents estimated target center, the 2nd image shows center voting, the 3rd image shows estimated target scale and position and the 4th image shows target estimate superimposed on the image.

Figure 5.8. Tracking under occlusion that the target is covered by another player. Despite, the voted center is correct, and the resulting confidence map provides adequate information to estimate the target state.

Figure 5.9. Illustration of the removal of unique cells in the representation which are contaminated by background clutter. Each triplet of images shows a
consecutive frame from a sequence. In the first image, the background does not cause confusion. The second triplet shows confusing descriptors due to common green color in the background and on target. The last triplet shows that the confusing information is removed from the unique target representation and the target can again be tracked well.

Figure 5.10. Illustration of tracking with target deformations and out plane rotations. When these cases occur the center voting scheme may not work (2nd left image). But the appearance descriptors encoded in the cells may work. In this example, we show that the confidence map provides a good approximation of the target location (red box in the last image) despite center voting errors.

Figure 6.1. The tracking flow of proposition. In the proposed tracking framework, there are two different networks: one is for the target identity concept recognition, and the other is for the target location detection.

Figure 6.2. The classification results of the TB50 target using the GoogLeNet. The vertical axes is the percentage of the target identified as different categories.

Figure 6.3. The new target concept estimation. Given a target, we can generate a number of heat maps for top n classes that the target may belong to like the red classes. Then based on the ground truth on the heat maps, the coefficients for the combination of these classes’ weights can be calculated to generate the new concept for the target.
Figure 6.4. The classification results of the targets in TB50. The vertical axis is the percentage that the target classified as same class in the same image sequence.

Figure 6.5. The heat map for prediction. In each sub four images, the left top is the target image with ground truth to generate the heat map and calculate the target concept also the initial score of the concept (top left). The lower left image is the search image, and the right bottom image is the heat map generated by the concept of the target image. The ground true in search image is used to calculate the predicted score.

Figure 6.6. The predict and initial score in different situations.

Figure 6.7. The architecture of localization network.

Figure 6.8. Average success plot and precision plot for the OPE on TB50.

Figure 6.9. Illustration of good tracking with (a) significant illumination change, (b) texture and edge change, (c) deformation and out plane rotation, (d) fast motion and motion blur.

Figure 6.10. Illustration of the occlusion scenarios.

Figure 6.11. Illustration of suppressing distractions.

Figure 6.12. Illustration of concept and appearance information both lost the target.

Figure 7.1 The medical navigation and education augmented reality system.

Figure 7.2. Sensors in HoloLens. Image from HoloLens document [144].

Figure 7.3. The working flow of the two branches of the AR system.

Figure 7.4. The coordinate system of the operation table.
Figure 7.5. The scalpel with LED lights on the top. .................................................. 137

Figure 7.6. The localization of LED lights. .............................................................. 139

Figure 7.7. The smooth of tip’s trace and the delay of the position. At the upper row, the solid circle is the measured location; the dashed circle is the smoothed location for the solid circle. At the lower row, the new location is calculated from the previous smoothed locations and the new measured location. ....... 141

Figure 7.8. The virtual arm model. ........................................................................ 141

Figure 7.9. The demo of display virtual operation scenario with HoloLens. .......... 144

Figure 7.10. The demo of operation navigation with HoloLens. Image (a) and (b) display the navigation of the virtual scalpel. Image (c) shows the scalpel trace (black) drawing by a human which following the virtual scalpel. The red line is the ground truth that used to record the virtual scalpel’s motion. .................. 145

Figure 7.11. The setup for the experiment of location measurement. .................... 146

Figure 7.12. The track’s locations for the experiment of localization measurement. The blue circle is the positions that measured, the red circle is the center of the points in the same track, and the lines are the fitting results from these positions. ......................................................................................... 147

Figure 8.1. The detection framework of single shot multibox detector [147]. ......... 153

Figure 8.2. The example annotations of the DAVIS 2017 dataset [148]. ............... 154
Chapter 1: Visual Tracking

The tracking problem can be defined as finding a target given its position in the first frame. Tracking is one of the most important and initial steps for many visual tasks. In humans, when an interesting object is detected, the very first step is to focus on the object and follow its motion. The focusing and following processes actually are also steps of a typical visual tracking process. The focusing-step corresponds to target recognition or analysis, and the following step is searching its location again in the future frames. From an analysis perspective, the visual tracking problem can be treated as estimating the new target location in the future frames based on the knowledge gained from the previous location and appearance of the target. Hence, most researchers in visual tracking area are dedicated to extracting and representing previous information which also are the topics of this dissertation. Since there are many practical problems, the performances of visual tracking methods are affected by different situations such as illumination changes, object deformation, clutter background, and occlusion.

The work is devoted to overcoming some of these problems with different target appearance model and tracking algorithms. This chapter will introduce the importance of visual tracking, the scope of research and the contribution of the dissertation to the visual tracking field.
1.1 Motivation

With the increase in demands of automation and the wide availability of cheaper cameras, there has been a sharp growth in various optical applications in many fields. Visual tracking is one of the fundamental components of many computer vision based applications, like safety supervising, motion recognition, and scene understanding. These applications typically require the spatial location of the target at different degrees of precision, which can be computed using visual tracking. For navigation systems, similar to the ones used in mobile robots, autonomous cars, and unmanned aerial vehicles, visual tracking helps identify important objects and their positions in the scene. For surveillance applications, such as security or monitoring of people or other targets, the location and motion information extracted by visual tracking are directly used for situational awareness. The visual tracking is also used in many other events like tracking the game player, counting the person, and automatically making the camera focus on the target.

While visual tracking has been a topic of research for three decades and many successful methods have been proposed, it is still an open problem due to the complicated tracking scenarios. Tracking is generally based on using past information represented in a certain way. However, in most practical scenarios, the past information may be erroneous and may not help to obtain the new target state. The illumination changes, non-rigid motion deformations, and partial or full occlusions may significantly change the appearance of the target. Also, cluttered background, especially which are similar to the target will cause the tracker to drift to wrong regions. There is still no universal method that can deal
with all tracking situations. In addition, research on how to use past information is still an open area.

1.2 Research Scope

As mentioned above, visual tracking is about using target’s previous information to estimate its new state. How to represent and use the past information is a core research topic within visual tracking and it depends on how appearance and motion are modeled. Considering that the motion model is simpler, most research has focused on the description of the target’s appearance. This dissertation also mainly focuses on the target appearance models by proposing different feature extraction methods. In order to show the effectiveness of the introduced models, the results are given on single object tracking datasets.

In visual tracking, the target state is mainly referred to as the target’s location and size. For most research, the location and size are represented by a kind of primary shapes like rectangle, ellipse, and so on. So, in most visual tracking tasks, the goal is to find the target’s trajectory by marking these primary shapes’ locations in the images which are recorded continually. Considering in many cases, the primary shape cannot fit the target appropriately, some researchers included image segmentation methods to mark the targets with their own shapes. The work in this dissertation also considers using the silhouette to mark the target and improve the tracking performance.

Besides how to mark the area of the target, how to represent the appearance is more important. In most researchers, the target is represented by a combination of features. These features may refer to points, edges, colors, shapes and so on. In some scenarios, the
feature may only mean the interesting part or structure in the image, like corner points, lines which are obviously different respect to its surrounding. However, in this dissertation, the feature is mainly referred to the description method of some area in the image. The size of the feature area is decided by the description method and that can be one point, a rectangle or even the whole image. For example, the shape feature may point to a person as a target, or just the head, and the whole human body by its pre-definition. The work in the dissertation discusses three different ways to extract features and represent the target for different situations.

In some practical applications, target’s three dimensions (3D) location is more useful than its location in the two-dimension (2D) images. With the geometric relationship, object tracking with multiple cameras can retrieve the target’s 3D location from the tracking results from 2D images. In the dissertation, a 3D tracking with the stereo cameras is used to track the scalpel and display it with an augmented reality device to show the application of visual tracking in 3D space.

1.3 Contribution

Although a lot of visual tracking methods have been proposed in these years, there are still problems that make the visual tracking very complicated. So, from different view angles and for different situations, this work in the dissertation offers several different ways to perform visual tracking. The main contributions of the work are described as follow:

1) Without complicated and time-consuming features, an efficient tracking method with distinctive target colors and silhouette is proposed to track an object,
especially non-rigid object. The implementation of this method is simple and easy, and the performance is excellent at high execution rate for various practical scenarios.

2) From a different view angle, another tracking method is proposed to represent target by a combination of unique features and offers an efficient way to estimate the uniqueness of them. This method also includes a novel tracking method using feature’s confidence and spatial information to estimate target area.

3) Besides the low-level features, deep learning based neural networks can offer high-level semantic features. Another work in this dissertation proposed a novel method using the learned high-level semantic features to generate tracking object’s concept and track it with a Siamese network.

4) Finally, this work demonstrates a 3D space localization system with high location precision in the practical application. With an augmented reality device, the system can be used as a medical training and surgical navigate tool.

1.4 Dissertation Structure

The rest of this dissertation is organized as follows. Chapter 2 reviews the related work in visual tracking including the tracking in 2D and 3D space as well as some appearance models. Chapter 3 provides the background knowledge about image feature, classifier, and neural network, also, with 3D space tracking, the camera geometric is discussed in this chapter. In Chapter 4, an object tracking method using color and silhouette is introduced to efficient tracking non-rigid deformation objects. In Chapter 5, a novel method using unique feature vote tracking is proposed to robustly represent and track the
target in real time. In chapter 6, a deep learning based neural network tracking system is proposed to track an object with the high-level semantic features which achieved the state-of-art performance. In the next chapter, a 3D point tracking system with an augmented reality device is presented to navigate the surgical operation and education. Finally, Chapter 8 summarizes the works in the dissertation and suggests the future work.
Chapter 2: Related Work

This chapter discusses visual tracking algorithms in image sequences and videos. Object tracking is one of the most basic problems in computer vision. With the proliferation of powerful computers and cheap but high-quality cameras, the need for automated video analysis has been increased greatly. Even visual tracking problem has been studied for decades, it is still an open problem due to its complications, like illumination change, part occlusion, clutter background, and deformation. In this chapter, an overview of general visual tracking is given in Section 2.1. Then, related works in visual tracking are described in Section 2.2. Following that, tracking with multiple cameras that can retrieve 3D space location is discussed in Section 2.3. Finally, a conclusion about visual tracking work is given in Section 2.4.

2.1 Overview of Visual Tracking

Generally, the problem of visual tracking can be defined as estimating the trajectory of an object in the image as it moves along a scene [1]. Simply, given a target in the first frame of a video, the goal of the tracking is to localize the identified target again in the following frames in the video. The applications of object visual tracking are pertinent in motion-based recognition, automated surveillance, gesture, and eye gaze tracking for human-computer interaction, traffic monitoring and autonomous cars [1]. Besides these, multiple view visual tracking can offer information in 3D space which is useful in the car
and robotic navigation, location analysis, and object 3D status tracking. About the multiple view visual tracking, more details will be discussed in Chapter 3 and Chapter 7. Numerous object tracking algorithms have been proposed for decades, but none of them can claim that they have solved the problem. The core of visual tracking is how to describe the target. The complexity of visual tracking mainly due to the information loss caused by representing a 3D world object on a 2D image. This 3D to 2D projection may make the display of object naturally inconsistent. From different view angles, the object shows infinite appearances which the 2D visualization on the images can’t describe all of these appearances. So, without the 3D information, the visual tracking has a natural defect. Besides this, the environments and the quality of image both produce effects on the tracking performances like illumination changes, clutter background, occlusions and noise in images.

Considering the problems visual tracking is facing, countless tracking methods have been proposed for different visual situations. However, the core is still about how to represent target appearance which is invariant to environment changes. From points, edges, to contours and template description, many different feature descriptors have been proposed trying to find a suitable and robust appearance description. Furthermore, considering appearance changes, multiple appearance models have also been proposed for different tracking scenarios. On the other hand, distinctive information between target and background is more important when there are distractors like clutter background and inter-category objects. Based on the previous visual tracking works, this work in the dissertation has proposed three different algorithms at different aspects for visual
tracking. In order to keep the integrity of components here, in the following sections, the reviewing of related works in visual tracking are discussed first.

2.2 Visual Tracking in Images

The core principles of visual tracking are how to represent target and how to find the target again. From the two aspects, many tracking methods have been proposed to conquer different visual problems in tracking scenarios. Generally, the tracking methods can be categorized into two classes: generative and discriminative. Generative tracking methods, especially the ones that are based on the tracking-by-detection framework, focus on how to represent the target by feature descriptions and measure the similarities between the representation model and candidates [2-6]. These trackers, however, are vulnerable to background clutter, and partial or full occlusions. In order to offset this problem, the discriminative trackers consider the distinctiveness between the target and background. They use classifiers to find the target from the background based on the pre-trained feature distributions [7-11]. During the tracking, the classifiers are also updated online by training with the new background and target samples. Also, recently, more and more methods are combining these models together to get robust tracking methods. So, in this review, the two categories are first discussed separately, then some new methods which have both the characters of general and discriminative trackers are illustrated.

2.2.1 Generative Tracking with Appearance Representation

The object can be defined or represented by any method or rule that will be further used. These description methods or rules are called descriptors which designed to describe appearance. Usually, descriptors are linked with the word “feature” which used to
describe interesting parts that have significant changes around it, like corner points, edges, blocks, primitive geometric shapes. Actually, from a pixel to the whole image, any place at any scale can both be described with a descriptor even they are just pure planes. So, in this paper, the features not only refer to interesting parts but any area on the target. To represent a target, points, edges, contours and geometric shapes are commonly employed.

To represent an object by a set of points [12] is the simplest and most efficient way to track the object. These points should contain distinctive information to distinguish themselves from others. Generally, interest point like the corner is chosen as tracking feature. There are basically two ways to track points: optical flow tracking and feature point detection and matching. Optical flow is used to estimate a dense field of motion vectors which defines the movement of each pixel in a limited region [13]. There are two basic constraints for optical flow. The first one is brightness constraint which assumes the brightness of the same pixel in consecutive frames is not changed. The other constraint is that the movement of the pixel is limited to a small value. There are many works have been proposed to alleviate the constraints including methods by Horn and Schunck [13], Lucas and Kanade [14], Black and Anandan [15]. The other way to track interest points is by using descriptors to describe them then re-detect and match them by the same descriptor. The most popular point descriptors include SIFT, Binary, and et al with more detail can be found in Chapter 3. They use different ways to collect local information around the point to generate a description that represents the points. Then, in the new frames, another set of interesting points are detected and described by the descriptors.
Through cross comparison, the most similar descriptions between the two point sets are considered as the matching points which means they are the same point. The point is the cheapest feature that can be detected and tracked, however, points are sensitive to noise and appearance changes.

With more surrounding information is considered, the feature descriptor is more robust to unexpected appearance changes. So, there are many trackers that based on the edge and contours. Tracking edge or contour is not just considering the pixels on them, but their distribution and shape. There are many ways to detect edges, like Canny [16], and segmentation method may also generate edges. Like points, the edges are also first been detected and then described to find the best match. There are several methods to describe edges like texture-edge descriptor [17], Hierarchical Chamfer matching [18]. Contour tracking is similar to the edge tracking. Shape matching is the most popular way to track contours. From the previous works like Yilmaz 2004 [19] to the recent Iterative closest point (ICP), the methods can be very complicated or extremely simple. Currently, segmentation becomes more and more accurate and fast [20], then the contour can be directly tracked by detection.

The target in visual tracking is usually marked by a regular primitive shape like circle and rectangle. The appearance in the marked area is the most important information for visual tracking. Besides interesting points and edges which focus on the local information, more and more descriptors are using global information to describe the whole target. In all of the representation methods, texture descriptions which measure the intensity variation of the target have been proposed at long time ago. Gray-Level Co-occurrence Matrices
(GLCM’s) [21] used a 2D histogram which shows the co-occurrences of intensities in a specified direction and distance and Law [22] used twenty-five 2D filters generated from five 1D filters corresponding to level, edge, spot, wave, and ripple. The texture features are more robust to illumination changes, however, are still sensitive to translation, rotation, and deformation which are very common problems in visual tracking. So, more researchers focus on how to design a feature descriptor that invariant to these appearance changes. Black and Allan extracted Eigen features to represent target which undergoes continuous appearance changes [5]. TLD [3] uses random forest with a 2 bit BP feature to store the appearance of the target. More recently, Hog [23] counts occurrences of gradient orientation in the target area to form a descriptor and has been wildly used in object detection, recognition, and tracking. Despite reported success in some benchmark sequences, these representations have been shown to fail in the cases of noise and similarity between the target and background regions [24] [25]. In order to overcome this shortcoming, researchers have tried fusing multiple models and features to improve the tracking performance [26] [27] [28]. Besides the multiple models, MUltiSTore Tracker (MUSTer) [29] uses short-term memory and long-term memory for the appearances at different situations. Also, the TGPR tracker [30] models the probability of target appearance using Gaussian Process Regression. The appearance model is learned in a semi-supervised fashion.

2.2.2 Tracking with Discriminative Information

The goal of tracking is finding the target from the image, in another word, tracking can be described as classify the target area from the background area in the image. So, based on
this idea, many trackers are trying to find discriminative information generated from the target and non-target regions to perform tracking. Most of these methods train a classifier and choose discriminative descriptors. In a tracking framework, the classifier is learned from off-line data, and adjusted online to separate the target from the background [11] [27] [31]. These methods choose from a number of classifiers, among which the support vector machine (SVM) is the most commonly adopted one [10] [32] [33]. SVM uses kernel functions to project extracted descriptors into a higher dimensional space which contains separable data under required conditions. Boosting-based tracking is also commonly used [34] [35]. These methods cascade a set of weak classifiers to create a better one. The samples used to train the classifier can affect the tracking performance. Multiple instance learning (MIL) used a set of ambiguous positive and negative samples which in part belong to the target or the background [36]. Similarly, in TLD [3] [37], the authors use positive and negative experts to improve the learning process. An important phase of these discriminative trackers is the updating of the classifier. For this purpose, the detected target, and its neighborhood are used to generate new target and non-target samples. Online or incremental training, however, are time-consuming and slows down the tracker throughput. MEEM [38] uses SVM to detect target based on the entropy of the score function which uses an ensemble of experts about historical snapshots during tracking.

Aside from the aforementioned trackers, researchers have tried to directly find the discriminant features instead of using classifiers [8] [9] [39]. In [8], Collins et al. use histograms to estimate the feature distribution for background and target separately.
Similarly, in [40], Horst et al. use the color histogram to represent distinctive colors and use them to prevent the drift problem.

Recently, a correlation filter based tracker (KCF) has achieved robust tracking performance with very high-speed [41]. The KCF tracker only considers the previous appearance of the target, and it fails in case of large scale and appearance changes. There are several improved algorithms based on the correlation filter [42] [43] [44], that include more appearance models or scale search algorithms.

2.2.3 Tracking Based on Neural Network

Recently, Deep Neural Networks (DNNs) have demonstrated promising performance in tasks like image classification [45], object detection [46] [47] and segmentation [48]. Different from the low-level features, the DNNs, especially the convolutional neural networks (CNNs) have been shown to learn high-level semantic information of the object. After training on large-scale datasets like ImageNet [49], it has been shown that the networks can learn distinctive information for different object categories.

Motivated by this fact, many CNN based trackers have been proposed. For tracking, the network is typically used as a feature extractor [50-53]. These features have higher semantic information than low-level features used in the past. In [50] [51], the authors use the characteristics of lower and higher feature maps in the network to represent the target at the category and individual levels. In [52], previous target patches are stored in a filter bank to do a convolution operation at new frames to highlight the object location. In [53], sampled feature maps are classified by SVM to generate a saliency map. More recently, a number of studies use Recurrent Neural Networks (RNNs) for visual tracking
In [55], Held et al introduced a tracker which has a network with Siamese architecture. The inputs to their system are the target and search image and the output is the bounding box within the search image. In [54], Bertinetto et al use another Siamese architecture with a different output which is a confidence map that shows the similarities of the corresponding sub-windows. Unlike the sliding window methods, this Siamese network can directly generate all possible locations’ similar scores by only one scan; and it handles the scale variations by repeated estimations.

Many of the aforementioned trackers predict a heat map (or a confidence map) that is generated by a correlation filter [52], a sparse combination of feature maps and direct use of a convolutional network [54]. This activation map [50] can highlight the structure or region of the image that belongs to the given class. Their method is used to detect objects which have been pre-trained and the accuracy is critically dependent on the classification quality. However, in visual tracking, a novel target may not always belong to one of the pre-trained object categories and there also may not be enough data for a new training. So, it is not preferred to directly use this heat map for a tracking prediction.

2.3 Visual Tracking in Space

3D space tracking is a challenging task, especially for the real-time, multi-view object tracking. A 3D space tracking system usually performs object detection, tracking, and 3D positioning estimation simultaneously in multiple camera setups with the real-time requirement. Despite its complexity, space tracking has tremendous potential for applications in fields such as visual surveillance, monitoring, robot vision, SLAM (Simultaneous Localization and Mapping), and indoor localization [58].
The most useful function of the multi-camera system is the localization ability. In most cases, the 3D position of the target is calculated by using triangulation of its image in multiple views [59] [60] [61]. It is very popular to use the stereo camera in automatic vehicles for the navigation and target recognition [62] [63]. In [64], Morat et al. used Lucas-Kanade and epipolar constraint to track feature points with a stereo-camera to localize the car’s position in 3D. In [60], Badino et al. used optical flows and a Kalman filter to iteratively refine the velocity and position of 3D points for the autonomous navigation. More details about the formulas that are used to estimate 3D position by multi-camera will be introduced in Chapter 3.

On the other hand, the multi-camera will offer multi-view which can greatly relieve the occlusion problem in visual tracking [65] [66]. For instance, Krumm et al. [67] introduced a multi-person tracking system using two sets of stereo cameras with three computers to process images from the stereo camera separately or together. The 3D position of the target may be calculated from points like [68] introduced. In [68], a multi-person tracking system using multiple stereo cameras which used a Harris detector to find feature points is proposed. Also, region-based stereo algorithms that do not need point matching to find the 3D position of a point are proposed by [69,70].

In practice, the tracking object usually is small like points and the tracking problem is relieved by well-controlled illumination and background. For example, Grover and Tavare[71] introduced a real-time, insect 3D tracking system with controlled blue background and the visual hull that been captured by three cameras. In this dissertation, the target of the space tracking system is a scalpel. The purpose of the system is to
acquire precision location information of the tip of the scalpel. Unlike the works introduced above, the 3D tracking needs extremely high quality at real-time. In order to implement it, the work in the dissertation simplified the scalpel tracking into LED light tracking with a stereo-camera. The details of the tracking method will be presented in Chapter 7.
Chapter 3: Background Knowledge

This chapter discusses background information to aid in understanding the concepts, algorithms, and methods discussed in this dissertation. Firstly, the background knowledge discussed is the features and classifiers, which are the fundamental part of many visual tasks that this dissertation will discuss. Then, convolutional neural network in deep learning that have been used in many the-state-of-art methods in the computer applications right now will be introduced. Finally, the photogrammetry including camera calibration, bundle adjustment; and the epipolar geometry which this dissertation will use in 3D tracking will be discussed.

3.1 Image Feature

In general, image feature means a distinctive place in the image, and it usually includes two parts: 1. Feature detection, the way to identify the distinctive area in the images. 2. Feature description, how to represent features with unique and robust description. There are various methods to detect features and to describe them. This section will review these methods of the two parts separately.

3.1.1 Feature Point Detection

Feature detection is a process that automatically examines an image to identify features. The features include single points, edges, blobs, contours and so on. Since local feature is
the most basic and popular one, the dissertation here will mainly focus on them, like key points.

The purpose of feature point detection is to find a set of distinctive points that can be reliably localized under varying imaging conditions, viewpoint changes, and in the presence of noise [72]. Motivated by this, the feature points should be those exhibiting signal changes compared to nearby areas, like corners. There are a lot of corner point detectors, the following will review some most popular detectors like Harris and Hessian based corner detectors.

**Harris Detector [73]**

The basic idea of the Harris detector is to find a region or window that can yield a large change in appearance by shifting it in any direction. The mathematic equation about the change of intensity for the shift \([u, v]\) is:

\[
E(u, v) = \sum_{x,y} w(x, y) [I(x + u, y + v) - I(x, y)]^2, \tag{3.1}
\]

Where the \(w(x, y)\) is the window function, \(I(x, y)\) is the image intensity at location \((x, y)\). Since \([u, v]\) is a small value vector, the equation can be approached with Taylor series’ first partial derivation as:

\[
E(u, v) \approx \sum_{x,y} w(x, y) [I(x, y) + u I_x + v I_y - I(x, y)]^2
\]

\[
= \sum_{x,y} w(x, y) (u^2 I_x^2 + 2uv I_x I_y + v^2 I_y^2)
\]

\[
= [u \ v] \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} [u \ v] = [u \ v] M [u \ v], \tag{3.2}
\]
where $M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$.

Since, the distribution of $x$ and $y$ derivation can be characterized by the shape and size of the principal component ellipse, the Harris detector searches points where its second-moment matrix (here the $M$ matrix) around it has two large eigenvalues ($\lambda_1, \lambda_2$).

Generally, the measure of corner response $R$ is:

$$R = \det M - k (\text{trace} M)^2,$$

where $\det M = \lambda_1 \lambda_2$, $\text{trace} M = \lambda_1 + \lambda_2$,

$k$ is an empirically constant at the range 0.04 to 0.06.

The response $R$ depends only on eigenvalues of $M$. $R$ is large for a corner, negative with large magnitude for an edge and close to 0 for a flat region.

**Hessian detector [74]**

The Hessian detector searches the points which have strong derivatives in two orthogonal directions. It uses the matrix of second derivatives, the hessian matrix:

$$H(x, \sigma) = \begin{bmatrix} I_{xx}(x, \sigma) & I_{xy}(x, \sigma) \\ I_{xy}(x, \sigma) & I_{yy}(x, \sigma) \end{bmatrix},$$

where the $\sigma$ stand for the smoothing parameter of a Gaussian filter. The detector computes the second derivatives $I_{xx}, I_{xy},$ and $I_{yy}$ for each image point and then searches for points where the determinant of the Hessian becomes maximal:

$$\det(H) = I_{xx} I_{yy} - I_{xy}^2,$$

The detector finds points which have the maximal determinant of the Hessian matrix in its sub-area. That means only keeping pixels whose determinant value is larger than the
values of all the other pixels in a specified neighbor area. Of course, the value should be larger than the pre-defined threshold.

Basically, the Hessian detector will give more points than Harris. Hessian can detect point, besides corner, but the one with strong texture variation. On the other hand, Harris’s points are typically more precisely located at the corners. So, Harris points are more suitable for the purpose of detecting corners and acquiring a more precise location. Hessian points provide more interest points and denser coverage which will be better for object 3D reconstruction.

**FAST: Features from accelerated segment test [75].**

FAST compares pixels only on a circle of fixed radius around the point. By considering the circle of 16 pixels around the candidate, if there exists a set of 12 contiguous pixels in the circle which is all brighter than the center or all darker, the candidate will be extracted as feature point.

![Figure 3.1. The illumination of the FAST detector. The image is from [75].](image)
FAST indeed detects points in a fast and dense way. And with Laplacian function which includes scale selection, the FAST is more practical.

This corner detector can find the exact same corners with image rotation and shift. However, the main problem we are dealing with in practice is not only the rotation and shifting changes but also scale and viewpoint changes. So, we want to detect points which can be repeatedly detected in different image conditions. For that purpose, there are some scale unchanged detectors.

**Laplacian-of-Gaussian (LoG) Detector [76]**

![Figure 3.2. The Laplacian-of-Gaussian (LoG) scale selection filter. The filter response is strongest for circular image structures whose radius corresponds to the filter scale. The image is from [72].](image-url)
LoG filter is a circular center-surround structure. In the center region, the weights are positive, but at outside, the weights become negative. Thus, by applying different size of the circular center-surround structure (here defined by the parameter $\sigma$), it will yield a maximal result at the scale which fits the features. Thereby, we can automatically choose the detector scale for the features and the center of the feature should be the key points.

$$L(x, \sigma) = \sigma^2 (I_{xx}(x, \sigma) + I_{yy}(x, \sigma)). \quad (3.6)$$

**The Difference-of-Gaussian (DoG) [77]**

DoG detector can be used to approximate the LoG, but more efficiently computed.

$$D(x, \sigma) = (G(x, k\sigma) - G(x, \sigma)) \cdot I(x), \quad (3.7)$$

The dot operation only need to compute the difference of two adjacent scales, $k$ is the interval factor. The DoG and LoG have very similar obtained regions, but DoG is far more efficient.

**Harris-Laplacian Detector [78]**

Because the LoG or DoG don't consider the discrimination between the interest points, combining the Harris operator to filter the points will keep the points which have more discriminative power. It first generates two separate scales for both Harris and Laplacian, then uses Harris to find the potential interest points on each scale and selects the points that also have max values in the Laplacian detector.

The Harris-Laplacian detector gives points which are robust to scale, rotation, illumination changes, and noise as well as highly discriminative. But, the main problem is that it returns a much smaller number of points than the other detectors.

**Maximally Stable Extremal Regions (MSER) [79]**
The maximally stable extremal region is a connected component from a watershed segmentation algorithm to the image and extracts homogeneous intensity areas. Since the regions are generated by a segmentation process, they have no restricted shapes. So, the MSER is invariant to affine transform.

The method works best for structured images that can be segmented very well. Also, it is sensitive to image noise. But this method is relatively fast. It is the most efficient one among the affine invariant feature detectors.

Summary of local point detectors

Harris and Hessian detectors are the basic point detector which can give precisely localized points which have rotation and shift invariant characters. And basically, the Hessian detector will return more points than Harris.

LoG and DoG detectors respond to blob-shaped structures and we can extract the center as interest points. And these points beside rotation and shifting also have scale-invariant characters. Combining Harris and Hessian with Laplacian scale selection will give more discriminative information but fewer points.

Table 3.1. The performance of different point detectors. The data is from [79].

<table>
<thead>
<tr>
<th>Point detector</th>
<th>Rotation invariant</th>
<th>Scale invariant</th>
<th>Affine invariant</th>
<th>Repeatability</th>
<th>Localization accuracy</th>
<th>Robustness</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harris</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Best</td>
<td>Best</td>
<td>Best</td>
<td>Good</td>
</tr>
<tr>
<td>Hessian</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>FAST</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Best</td>
</tr>
<tr>
<td>Harris-Laplace</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Best</td>
<td>Best</td>
<td>Good</td>
<td>Fair</td>
</tr>
<tr>
<td>DoG</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>MSER</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Best</td>
<td>Best</td>
<td>Good</td>
<td>Best</td>
</tr>
</tbody>
</table>
Finally, with affine transform, Harris-affine and Hessian-affine detectors can deal with the viewpoint change problem. Also, MSER detector can get these at a faster speed but with fewer points. So, based on the different situation and different application, different point detectors can be chosen as needed.

3.1.2 Local Feature Descriptors

After a set of interest points have been identified, each point’s content needs to be represented or encoded using suitable descriptor which can help distinguishing and matching.

**SIFT descriptor [80]**

The Scale Invariant Feature Transform (SIFT) was originally proposed by Lowe [80]. It combined DoG detector and a corresponding feature descriptor. Actually, the feature descriptor can be used with any point detector as mentioned above.

The first step for the descriptor is orientation normalization. Because descriptor not only represents a point but a patch around the point, the patch content needs to be normalized for rotation invariance. The normalization is typically done by finding the dominant orientation. The dominant orientation that computed by the gradient orientation histogram of all pixels in the region with 36 bins covering the 360 degrees. Also, there will be a Gaussian window centered on the key point with specified scale to adjust the weight making the closer ones have higher votes. The highest peak of the orientation should be the dominant orientation.

The second step is to regularize the interest region to a 16x16 pixel regular grid. Then the whole image region is divided into 16 sub-grids with 4x4 pixels. For each sub-grid,
gradient orientation histograms with 8 orientation bins are computed. At last, it combines all the 16 sub-grids together to generate a 4x4x8 dimensional feature vector.

There is another SIFT descriptor called PCA-SIFT [81]. PCA stands for principal components analysis. Instead of using SIFT’s smoothed weighted histograms, this PCA-SIFT uses PCA to the normalized gradient patch which will make the descriptor more distinctive and robust to image deformations as well as increase the accuracy and faster the matching.

![SIFT descriptor diagram](image)

**Figure 3.3.** The SIFT descriptor. For each scale invariant region, first do the orientation normalization, then project to a 16x16 pixel regular grid. Divide the grid to a 4x4 sub-grid and compute the gradient orientation histograms for each sub-grid. Finally, add all sub-grid gradient orientation histograms together to generate a 128-dimensional feature vector. The image is from [80].

**SURF descriptor [28]**

Speeded up robust features (SURF) use integral images to compute Haar wavelets or any box-type convolution filter with a very fast speed. It is a scale-invariant feature detector based on the Hessian-matrix.
There is another detector that combines SURF and a Hessian-Laplace detector. Without Gaussian derivatives for the internal computations, the descriptor is based on Haar wavelet responses which can be calculated efficiently with integral images. Motivated by SIFT, the interest area is divided into 4x4 subareas and each has computed its wavelet response in x and y directions.

\[ v = \{ \sum dx, \sum |dx|, \sum dx, \sum |dx| \}, \]  

(3.8)

For each subarea, a vector \( v \) is calculated. Adding all the vectors together, the interesting point is described by 16 such vectors. Also, the SURF algorithm uses sign of Laplacian to speed up the matching computation.

A further optimized version of SURF has been proposed by using the GPU, the feature extraction can be fast as 200 Hz for 640x480 images [82].

As we can see, SIFT and SURF are both based on histograms of oriented gradients, and there are also other descriptors based on them, like GLOH. SIFT and SURF are the most common descriptors, so this work has only discussed them here. The table below is about the performance of different method’s matching rate at different situation [82].

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
<th>Scale</th>
<th>Rotation</th>
<th>Noise</th>
<th>Illumination</th>
<th>Affine</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT</td>
<td>Fair</td>
<td>Best</td>
<td>Best</td>
<td>Fair</td>
<td>Fair</td>
<td>Good</td>
</tr>
<tr>
<td>PCA-SIFT</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Best</td>
<td>Good</td>
<td>Best</td>
</tr>
<tr>
<td>SURF</td>
<td>Best</td>
<td>Fair</td>
<td>Fair</td>
<td>Good</td>
<td>Best</td>
<td>Good</td>
</tr>
</tbody>
</table>

SIFT and SURF are based on histograms of gradients, and it will need some time to do the computation. At some situations, they are not fast enough for a real-time application.
However, this makes binary descriptors come in handy. This method only compares the intensity of point pairs. And the matching is also fast, just a computation of the Hamming distance.

**BRIEF descriptor [83]**

Binary robust independent elementary features (BRIEF) uses binary strings to describe interest points which can be detected by any point detector. This descriptor is highly discriminative and efficient. BRIEF does not have specified sampling pattern, it can be randomly uniformly sampled or randomly sampled using a Gaussian distribution or other. This descriptor runs very fast and has a reliable matching rate. However, it is not orientation and scale invariant which make it have many limits in the practical application.

**The ORB (Oriented FAST and rotated BRIEF) descriptor [84]**

ORB is very similar to BRIEF. It doesn’t have specified sampling patterns but it learns the optimal sampling pairs. Also, ORB uses an orientation compensation mechanism making it rotation invariant. ORB uses the intensity centroid to measure the point orientation.

In non-geometric transformation, BRIEF outperforms ORB. In affine transformation, BRIEF performs poorly under large rotation or scale change. However, in perspective transformations, BRIEF surprisingly slightly outperforms ORB [85]. Also, there are several other binary descriptors, like binary robust invariant scalable key points (BRISK) [86] and the fast retina key point (FREAK) [87].
3.1.3 The Application in Image 3d Reconstruction

Feature extractors can find interest points and match them in multi-images which would be used in the image 3D reconstruction. However, the interest points cannot cover all of the objects in most situations. They are basically distributed in texture rich areas. For homogeneous areas, there are no features to be extracted. But this short-coming can be compensated to some degree in the final 3D model because the homogeneous areas are usually smooth planes which can be estimated with the feature points. Also, in the stereo situation, the image will not change much, and, there is not too much rotation or affine changes. We can use easy point detectors to find as many as possible interest points, and also use easy ways to match them, like use the BRIF, cross-correlation coefficients are enough to do the matching with the help of epipolar line constraint.

For the application of image 3D reconstructing in some situations, the basic idea is to find as many as possible accurate interest points in each image, then matching these interest points in multi-images also as accurately as possible. So, for interest point detector, the Harris has the best accuracy which can reach sub-pixel, but it gives fewer interest points. Hessian and FAST can give much more points but cannot insure the accuracy. To choose the detector must make a trade-off between accuracy and efficiency and numbers. The DoG, SURF, and FAST are a more efficient way which people like to use in a real-time application. In some situation, combining several point detectors is also a choice for 3D reconstructing, like Furukawa, Yasutaka did in their paper [88]. They combined DoG and Harris to control the number and accuracy of the interest points. Also, some methods use both SIFT and SURF to detect and match the interest points [89].
3.2 Classifiers

Based on whether training samples are used or not, the classification can be divided by supervised classification and unsupervised classification. Supervised methods include maximum likelihood, minimum distance, artificial neural network, decision tree classifier, and support vector machine. When sufficient reference data are available and can be used as training samples, the signatures generated from the training samples are then used to train the classifier to classify the new-coming data into predefined classes.

Unsupervised classifications are clustering-based algorithms. They are used to partition the image or data into a number of classes based on the statistical information inherent in the image. No prior definitions of the classes are used. The analyst is responsible for labeling and merging the classes into meaningful classes.

Also, based on whether parameters such as mean vector and covariance matrix are used or not, classifications can be divided by parametric classifiers and non-parametric classifiers. Parametric classifiers include maximum likelihood and linear discriminant analysis. In these classifications, the Gaussian distribution is assumed. The parameters (e.g. mean vector and covariance matrix) are often generated from training samples. When data is complex, parametric classifiers often produce ‘noisy’ results. Another major drawback is that it is difficult to integrate ancillary data, spatial and contextual attributes, and non-statistical information into a classification procedure. Non-parametric classifiers include artificial neural network, decision tree classifier, evidential reasoning, support vector machine, and expert system. In these kind of classifiers, no assumption about the data is required. Non-parametric classifiers do not employ statistical parameters.
to calculate class separation and are especially suitable for incorporation of other assistant data into a classification procedure.

Different classifiers have their own strengths and limitations [90, 91]. Like parametric classifiers (e.g. maximum likelihood) and non-parametric classifiers (e.g. neural network, decision tree), when sufficient training samples are available and the feature of data in a dataset is normally distributed, a maximum likelihood classifier (MLC) may yield an accurate classification result. In contrast, when image data are anomalously distributed, neural network and decision tree classifiers may demonstrate a better classification result [92, 93].

Also, previous research has indicated that the integration of two or more classifiers provides improved classification accuracy compared to the use of a single classifier. This is the trend of further classifiers [94, 95].

**Decision Trees (DT)**

A decision tree is a decision support tool using a tree-like graph to classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. The structure allows users to take a problem with multiple possible solutions and display it in a simple, easy-to-understand format that shows the relationship between different events or decisions.

The disadvantages of DT are focused on continuous attributes and computational efficiency with growing tree size. According to the comparison provided for different classification methods in emotion recognition [96], DT is the best classifier method on
that group with 15 attributes. They are also not as flexible at modeling parameter space distributions having complex distributions as either neural networks or nearest neighbor methods.

One of the most useful characteristics of decision trees is their comprehensibility. People can easily understand the strategies of a decision tree classification. Since a decision tree constitutes a hierarchy of tests, an unknown feature value during classification is usually dealt with by passing the example down all branches of the node until the unknown feature value was detected. The output is a combination of the different class distributions. The assumption made in the decision trees is that instances belonging to different classes have different values in at least one of their features. Decision trees tend to perform better when dealing with discrete/categorical features [97].

**K-Nearest Neighbors (KNN)**

A very simple classifier can be based on a nearest-neighbor approach. This method is based on the principle that the instances within a dataset will generally exist in close proximity to other instances that have similar properties. Since the neighbor is nearby, it is likely to be similar to the object being classified and so is likely to be the same class as that object.

Nearest neighbor methods have the advantage that they can be easily implemented. They can also give quite good results if the features are chosen carefully and if they are weighed carefully in the computation of the distance. The power of KNN has been demonstrated in a number of real domains, but there are some limits to the usefulness. 1) They do not simplify the distribution of objects in parameter space to a comprehensible
set of parameters. Instead, the training set is retained in its entirety as a description of the object distribution. It needs large storage requirements. 2) The method is also rather slow if the training set has many examples and they are sensitive to the choice of the similarity function that is used to compare data. 3) The most serious short-coming of nearest neighbor methods is that they are very sensitive to the presence of irrelevant parameters. Adding a single parameter that has a random value for all objects can cause these methods to fail miserably. 4) They lack a principled way to choose k, except through cross-validation or similar, computationally-expensive technique [98].

**Neural Networks**

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems deals with information, such as the brain. Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output as shown in the graphic below. More details about neural networks will be discussed in section 3.3.
The biggest advantage of neural network methods is that they are general: they can handle problems with many parameters, and they are able to classify objects well even when the distribution of objects in the N-dimensional parameter space is very complex. The main disadvantage of neural networks is that they are notoriously slow, especially in the training phase. Another significant disadvantage of neural networks is that it is very difficult to find out how the net making its decision. Consequently, it is hard to determine which of the image features being used are important and useful for classification and which are worthless.

**Naïve Bayes**

In machine learning, naïve Bayes classifiers are a family of simple probabilistic classifiers. The Naive Bayes Classifier technique is based on the Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. A naïve Bayes classifier assumes that the value of a feature is unrelated to the presence or absence of any other
feature. Given a class variable $y$ and a dependent feature vector $x_1$ through $x_n$, Bayes’ theorem states the following relationship:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}.$$  \hspace{1cm} (3.9)

Using the naive independence assumption that:

$$P(x_i \mid y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i \mid y),$$  \hspace{1cm} (3.10)

for all $i$, this relationship is simplified to:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}.$$  \hspace{1cm} (3.11)

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y),$$

$$\hat{y} = \arg \max_y P(y) \prod_{i=1}^{n} P(x_i \mid y),$$  \hspace{1cm} (3.12)

If we can use maximum a posteriori (MAP) estimation to estimate $P(y)$ and $P(x_i \mid y)$; the former is then the relative frequency of class $y$ in the training set. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $(x_i \mid y)$.

The basic independent Bayes model has been modified in various ways in attempts to improve its performance. Attempts to overcome the independence assumption are mainly based on adding extra edges to include some of the dependencies between the features. In this case, the network has the limitation that each feature can be related to only one other feature. The Semi-naive Bayesian classifier is another important attempt to avoid the independence assumption [99], in which attributes are partitioned into groups and it is
assumed that $X_i$ is conditionally independent of $X_j$ if and only if they are in different groups.

In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters. Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This, in turn, helps to alleviate problems stemming from the curse of dimensionality. If a feature is numerical, the usual procedure is to discretize it during data pre-processing [100], although a researcher can use the normal distribution to calculate probabilities [101]. However, Naive Bayes requires little storage space during both the training and classification stages.

**Support Vector Machines**

Support vector machines (SVMs) are supervised learning models with associated learning algorithms used for classification, regression and outlier detection. Given a set of training data with labels as belonging to one of two categories, an SVM will train its model that assigns new-comings into one category or the other.

The SVM is a discriminative classifier formally defined by a separating hyperplane. SVMs revolve around the notion of a “margin”—either side of a hyper-plane that separates two data classes. Maximizing the margin and thereby creating the largest
possible distance between the separating hyperplane and the instances on either side of it has been proven to reduce an upper bound on the expected generalization error.

![Figure 3.5: A hyperplane used to separate the two data sets. The image is from [135].](image)

It is easy to show that, when it is possible to linearly separate two classes, an optimum separating hyperplane can be found by minimizing the squared norm of the separating hyperplane. The minimization can be set up as a convex quadratic programming (QP) problem [97]:

$$\min_{w, b} \Phi(w) = \frac{1}{2} ||w||^2, \text{subject to } y_i(w^T x_i + b) \geq 1, i = 1, \ldots, l.$$  \hspace{1cm} (3.13)

Because the model complexity of SVM is not affected by the number of features encountered in the training data, the SVM are well suited to deal with learning tasks where the number of features is large with respect to the number of training instance. However, most real-world problems cannot be solved by a hyperplane. One solution to the inseparability problem is to map the data into a higher-dimensional space and define a
separating hyperplane there. A kernel function can be used in these feature space transforming. The selection of an appropriate kernel function is important. It is common practice to estimate a range of potential settings and use cross-validation over the training set to find the best one. For this reason, a limitation of SVMs is the low speed of the training.

The advantages of support vector machines include: 1) Effective in high dimensional spaces. 2) Still effective in cases where a number of dimensions is greater than the number of samples. 3) Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. 4) Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include: 1) if the number of features is much greater than the number of samples, the method is likely to give poor performances. 2) SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation [102].

3.3 Artificial Neural Networks

Deep learning is the most powerful and useful analytical method in many artificial intelligence fields. For the sake of brevity, the mathematical explanations of deep learning will not be discussed in the dissertation. Instead, this section will explain the intuitions and simple equations behind deep learning and artificial neural network for the sake of completeness.
Figure 3.6. The illustration of the mathematical model of the working of brain cell. The image is from [142]. Image (a) is the biological neuron model of brain cell. Image (b) is an artificial neuron mathematical model of the brain cell.
In the late 1950s, David Hubel and Torsten Wiesel, two pioneering neurophysiologists, made experiments on a cat to show how the retina and the cerebral cortex work. And they found that some certain cortical cells response to some certain contours of a specific orientation. Later, more experiments discovered that the cells in the brain have the layer structure that some cells in the lower level receive simple information and send them to their higher or forward cells which combined all the income information together. Then these cells send their status to even higher places, such, layer by layer forming a topographical structure, like the Figure. 3.7 illustrates below.

3.3.1 The Structure of Neural Network.

![Figure. 3.7. The layer structure of artificial neural network.](image)

Based on this discovery, the deep learning neural network was proposed with similar structure. The basic structure of the deep learning network is shown in Figure 3.7, includes the input layer, the hidden layers, and the output layer. The number of hidden
layers could be 1 or tens decided by how deep the network is. The lower neuron (the first hidden nodes in the image) response to lower raw features like the input nodes in the image that directly get raw input. And then, the lower neural send their response to higher layers’ neural. These neural gather all the information from lower layer and generate their results as middle level information which will be sent to higher layers. The neuron in the last hidden layer contains very high sematic information about the whole input. Finally, with these high level sematic information, the output nodes offer the predicted results. In the artificial neural network, the responding $y$ of neuron are decided by:

$$y = f\left(\sum_i w_i \cdot x_i + b\right),$$  \hspace{1cm} (3.14)

where $f$ is an activation function, $w_i$ and $x_i$ are the connection weight and the corresponding input data, $b$ is the bias for each neural node. Activation function should be a non-linear function such as a sigmoid function:

$$S(x) = \frac{1}{1+e^{-x}},$$ \hspace{1cm} (3.15)

to project the input real value to range $(0,1)$. In neural network, using sigmoid function will cause two problems: one is sigmoids saturate and kill gradients the other one is sigmoid outputs are not zero-centered. The non-linear function in nueral network is the rectified linear unit (ReLU) that simply converts negative value to 0 and keep the positive value: $f(x) = \max(0, x)$. This function is simple and greatly accelerate the convergence of optimization. But ReLU units can be critical in training when the learning rate is high.

### 3.3.2 Convolutional Neural Network (CNN)

In image processing, local information in the 2D direction is more important. So, a convolutional neural network was proposed to implement deep learning network more
efficiently. The input is 2D information and the neural nodes only correspond to nearby areas and repeatedly go through all over the image which greatly reduce the number of parameters of the network. The structure of CNN is showing below.

![The structure of CNN and its features. The image is from [136].](image)

Figure. 3.8. The structure of CNN and its features. The image is from [136].

There are basically two kind of layers, convolutional and full connection layer. Convolutional layers just use the filters with different weights to do the convolutional operation around the image to generate the features like the upper images shows. At the end of the network, general, there are several layers that are full connected to one dimensional layers like the regular neural network with each node connecting to all previous ones and sending information to all forward ones.
Figure 3.9. Convolutional operation of CNN. Images are from [138] and [142]. Image (a) is convolution steps for a filter to go through the whole image. Image (b) is the detailed convolutional operation of a filter at one position.
The main difference between the CNN and a regular neural network is the CNN only consider a limited local area in 2 dimensions instead of considering all input in the full connected neural network. In the image filter operation, the filter’s parameters are the connection weights that decide the output of their filtered area. By moving the same filter all over the image with a fixed stride, the output can generate one feature map. Each filter on the same image will generate a different feature map, so, the number of filters decides the number of output channels of the feature map. Also, the filter is not considered on only one image channel, it includes all the channels. So, for a 3x3 kernel size filter, the parameters inside the neural network are 3*3*C, C is the number of channel.

As the previous parts have mentioned, the network is a sequence of layers which transforms information from low level to higher ones. Beside the input and output, there are several different layers that are used to form the network which include: convolutional layer, RELU layer, pooling layer and fully-connected layer. Figure 3.10 illustrates a typical ConvNet architecture.

![Diagram of ConvNet Architecture](image)

Figure. 3.10. The typical ConvNet architecture.
3.4 Camera Geometry

For space tracking, multiple cameras are utilized to localize the target. And the principle behind this is the photogrammetry. The essential part of photogrammetry is to measure the geometric relationship between camera images and objects in the 3D space. If we have the multiple cameras’ geometric information, an object’s space location can be recovered precisely by their geometric constrain. So, in this section, photogrammetry topics including camera calibration, epipolar geometry and bundle adjustment will be introduced for completeness.

3.4.1 Camera Calibration:

In order to get geometric relationship, the camera must be calibrated. First of all, the geometric relationship between the true object and its image generated by the camera is decided by the hardware inside the camera. Basically, we can use several parameters to describe these inherent characteristics which we call intrinsic parameters.
Firstly, start from a simple case, if we assume the world coordinate system’s origin is at the camera lens center and Z axis is along the camera perpendicular line of the lens like Fig. 3.11 shows. Then, based on the geometric constrains, the object A with space location $[X_A, Y_A, Z_A]'$, is projected to the image at location $a = [X_a, Y_a]'$. The relationship between them will be:

$$X_a = f \frac{X_A}{Z_A},$$

$$Y_a = f \frac{Y_A}{Z_A}. (3.16)$$

This is pinhole camera perspective projection model and the $f$ is the distance from image plane to the pinhole called focal length. If we use the image coordinate system on the image plane with the original center at $(x_o, y_o)$, then the equations will be:
\[ x_a = x_o - f \frac{x_A}{z_A}, \]
\[ y_a = y_o - f \frac{y_A}{z_A}, \tag{3.17} \]

As above, all points’ location is defined in the same coordinate system which has been defined by the camera or photo space. In practice, the object usually is located by its own object space or the world coordinate system. In photogrammetry, the photo space \( x^p \) is usually converted to the object space \( x^o \) with a transformation as:

\[ x^o = R x^p + T, \tag{3.18} \]

where \( R \) is the rotation matrix and \( T \) is translation matrix. And \( R \) stand for:

\[
R = R_x R_y R_z = \\
\begin{pmatrix}
1 & 0 & 0 \\
0 & \cos \omega & \sin \omega \\
0 & -\sin \omega & \cos \omega
\end{pmatrix} \\
\begin{pmatrix}
\cos \phi & 0 & -\sin \phi \\
0 & 1 & 0 \\
\sin \phi & 0 & \cos \phi
\end{pmatrix} \\
\begin{pmatrix}
\cos \kappa & \sin \kappa & 0 \\
-\sin \kappa & \cos \kappa & 0 \\
0 & 0 & 1
\end{pmatrix} \\
= \\
\begin{pmatrix}
\cos \phi \cos \kappa & \sin \omega \sin \phi \cos \kappa + \cos \omega \sin \kappa & -\cos \omega \sin \phi \cos \kappa + \sin \omega \sin \kappa \\
-sin \phi \sin \kappa & \cos \omega \cos \kappa & \sin \omega \cos \phi \\
sin \phi & -\cos \omega \cos \kappa + \sin \omega \sin \kappa & \cos \omega \sin \phi \sin \kappa + \sin \omega \cos \kappa
\end{pmatrix}, \tag{3.19}
\]

where the \( \omega, \phi, \kappa \) stand for the rotation angle omega, phi, and kappa from camera coordinate to world coordinate system along x,y,z axels. \( T \) is the location of the original point of the camera coordinate system in world coordinate system. Then, the equation 3.17 can be represent as:

\[
x_a = x_o - f \left[ \frac{m_{11} (x_A-x_L)+m_{12} (y_A-y_L)+m_{13} (z_A-z_L)}{m_{31} (x_A-x_L)+m_{32} (y_A-y_L)+m_{33} (z_A-z_L)} \right] = x_o - f \frac{r}{q}, \\
y_a = y_o - f \left[ \frac{m_{21} (x_A-x_L)+m_{22} (y_A-y_L)+m_{23} (z_A-z_L)}{m_{31} (x_A-x_L)+m_{32} (y_A-y_L)+m_{33} (z_A-z_L)} \right] = y_o - f \frac{s}{q}. \tag{3.20} \]

Where the \( m_{i,j} \) is the element in rotation matrix \( R \), \( X_L, Y_L, Z_L \) is the translation elements.
This is the equation called collinearity condition equation which is one of the most fundamental equations in photogrammetry.

However, the hardware installation always has errors. The image original center cannot be guaranteed to be at the center. So, the \( x_o, y_o \) need to be calibrated as well as the focal length \( f \). Furthermore, since the lens is not perfect, it may cause distortions like radial/barrel and tangential/pincushion distortions. To correct the radial errors, the following formula are used:

\[
\begin{align*}
x_a &= x \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right), \\
y_a &= y \left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right),
\end{align*}
\]

(3.21)

Here \( k_i \) are radial distortion parameters, \( r \) is the distance of the observed point \((x, y)\) from the original center.

For tangential distortion parameters \( p_i \), the correction formulas are:

\[
\begin{align*}
x_a &= x + [2p_1 x y + p_2 (r^2 + 2x^2)], \\
y_a &= y + [2p_2 x y + p_1 (r^2 + 2y^2)],
\end{align*}
\]

(3.22)

Figure 3.12. The illustration of lens distortion. The image is from [135].
So, at this point, all these camera parameters are only considered the hardware and they are called intrinsic parameters which are not changing with the camera. To get camera’s geometric relationship, the intrinsic parameters calibration is always the first step.

Another type of camera parameter we need to calibrate is the extrinsic parameter which describes the location and orientation of the camera in the world coordinate system. As above paragraphs, the world coordinate system is assumed to be at the camera center. If it is not, then all these relationships need to add a transformation from world center to camera center. There are six parameters to describe the transformation including translation \((X_L, Y_L, Z_L)\) and rotation \((\omega, \phi, \kappa)\). All these parameters are about the location of the camera and they are called extrinsic parameters which are changing while camera is moving. From these equations, if we can find enough corresponding points in the space and image, we can use bundle adjustment to get the parameters.

3.4.2 Bundle Adjustment

Bundle adjustment (BA) is the photogrammetric process that adjusts all photogrammetric observations to control points in a single solution [102]. In simple words, if we don’t know the camera parameters and the 3D location of points but only have some corresponding image points in multiple cameras, the BA can help to calculate all these unknowns.

BA actually is an optimization problem. It boils down to minimizing the re-projection error between the image locations of observed and predicted image points, which is expressed as the sum of squares of a large number of nonlinear, real-valued functions.
In the dissertation, we use bundle adjustment to resolve the camera calibration problem about the parameters \((\omega, \phi, \kappa, X_L, Y_L, Z_L)\) and the object position \((X_A, Y_A, Z_A)\)

\[
F(\omega, \phi, \kappa, X_L, Y_L, Z_L, X_A, Y_A, Z_A) = x_o - f^r \frac{r}{q},
\]

\[
G(\omega, \phi, \kappa, X_L, Y_L, Z_L, X_A, Y_A, Z_A) = y_o - f^s \frac{s}{q},
\] (3.23)

Since the collinearity equations are nonlinear, in most situations, it needs to be linearized to getting the solutions. In this work, we followed the solution from [103] to linearize the equations with Taylor’s theorem by taking partial derivatives of the equations with respect to the unknown parameters. If we have a value \(F_0\) based on the initial parameters \((\omega_0, \phi_0, \kappa_0, X_{L_0}, Y_{L_0}, Z_{L_0}, X_{A_0}, Y_{A_0}, Z_{A_0})\), these equations below show the linearized forms of equation 3.23:

\[
F_0 + \left( \frac{\partial F}{\partial \omega} \right)_0 d\omega + \left( \frac{\partial F}{\partial \phi} \right)_0 d\phi + \left( \frac{\partial F}{\partial \kappa} \right)_0 d\kappa + \left( \frac{\partial F}{\partial X_L} \right)_0 dX_L + \left( \frac{\partial F}{\partial Y_L} \right)_0 dY_L + \left( \frac{\partial F}{\partial Z_L} \right)_0 dZ_L + \\
\left( \frac{\partial F}{\partial X_A} \right)_0 dX_A + \left( \frac{\partial F}{\partial Y_A} \right)_0 dY_A + \left( \frac{\partial F}{\partial Z_A} \right)_0 dZ_A \approx x_a,
\]

\[
G_0 + \left( \frac{\partial G}{\partial \omega} \right)_0 d\omega + \left( \frac{\partial G}{\partial \phi} \right)_0 d\phi + \left( \frac{\partial G}{\partial \kappa} \right)_0 d\kappa + \left( \frac{\partial G}{\partial X_L} \right)_0 dX_L + \left( \frac{\partial G}{\partial Y_L} \right)_0 dY_L + \left( \frac{\partial G}{\partial Z_L} \right)_0 dZ_L + \\
\left( \frac{\partial G}{\partial X_A} \right)_0 dX_A + \left( \frac{\partial G}{\partial Y_A} \right)_0 dY_A + \left( \frac{\partial G}{\partial Z_A} \right)_0 dZ_A \approx y_a,
\] (3.24)

The equation can be simplified and normalized as:

\[
b_{11} d\omega + b_{12} d\phi + b_{13} d\kappa - b_{14} dX_L - b_{15} dY_L - b_{16} dZ_L + b_{14} dX_A + b_{15} dY_A + b_{16} dZ_A \\
= J,
\]

\[
b_{21} d\omega + b_{22} d\phi + b_{23} d\kappa - b_{24} dX_L - b_{25} dY_L - b_{26} dZ_L + b_{24} dX_A + b_{25} dY_A + b_{26} dZ_A \\
= K,
\] (3.25)
where $J = x_a - F_0$, and $K = y_a - G_0$, also, we use $\Delta X, \Delta Y, \Delta Z$ to represent $X_A - X_L, Y_A - Y_L, Z_A - Z_L$,

\[
b_{11} = \frac{f}{q^2} \left[ r (-m_{33} \Delta Y + m_{32} \Delta Z) - q (-m_{13} \Delta Y + m_{12} \Delta Z) \right],
\]

\[
b_{12} = \frac{f}{q^2} \left[ r (\cos \phi \Delta X + \sin \omega \sin \phi \Delta Y - \cos \omega \sin \phi \Delta Z) - q (-\sin \phi \cos k \Delta X + \sin \omega \cos \phi \cos \kappa \Delta Y - \cos \omega \cos \phi \cos \kappa \Delta Z) \right],
\]

\[
b_{13} = -\frac{f}{q} (m_{21} \Delta X + m_{22} \Delta Y + m_{23} \Delta Z),
\]

\[
b_{14} = \frac{f}{q^2} (r m_{31} - q m_{11}),
\]

\[
b_{15} = \frac{f}{q^2} (r m_{32} - q m_{12}),
\]

\[
b_{16} = \frac{f}{q^2} (r m_{33} - q m_{13}),
\]

\[
b_{21} = \frac{f}{q^2} \left[ s (-m_{33} \Delta Y + m_{32} \Delta Z) - q (-m_{23} \Delta Y + m_{22} \Delta Z) \right],
\]

\[
b_{22} = \frac{f}{q^2} \left[ s (\cos \phi \Delta X + \sin \omega \sin \phi \Delta Y - \cos \omega \sin \phi \Delta Z) - q (\sin \phi \cos k \Delta - \sin \omega \cos \phi \cos \kappa \Delta Y - \cos \omega \cos \phi \cos \kappa \Delta Z) \right],
\]

\[
b_{23} = \frac{f}{q} (m_{11} \Delta X + m_{12} \Delta Y + m_{13} \Delta Z),
\]

\[
b_{24} = \frac{f}{q^2} (s m_{31} - q m_{21}),
\]

\[
b_{25} = \frac{f}{q^2} (s m_{32} - q m_{22}),
\]

\[
b_{26} = \frac{f}{q^2} (s m_{33} - q m_{23}),
\]
To resolve the linearized equations, this work uses the least-squares solution to minimize the re-projection error which means $J$ and $K$ are optimized to close 0. To solve it, we change equation 3.23 and 3.24 to matrix format as:

$$A_{m\times n} \times X_{n\times 1} = L_{m\times 1},$$

(3.26)

where $A_{m\times n} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & \ldots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \ldots & b_{2n} \\ b_{31} & b_{32} & b_{33} & \ldots & b_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & b_{m3} & \ldots & b_{mn} \end{bmatrix}$,

$x_{n\times 1} = \begin{bmatrix} d\omega \\ d\phi \\ d\kappa \\ dX_L \\ dY_L \\ dZ_L \\ dX_A \\ dY_A \\ dZ_A \end{bmatrix}$,

$L_{m\times 1} = \begin{bmatrix} J_1 \\ K_1 \\ J_2 \\ K_2 \\ J_3 \\ K_3 \\ \vdots \\ J_m \\ K_m \end{bmatrix}$,

$m$ is the number of corresponding points*2 (for x and y). $n$ is the number of unknowns (here $n$ is 9). Hence, to solve the equations, there must be at least 5 point pairs. Using the least squares adjustment [102], we can get the solutions as:

$$(A^T A) \times X = A^T L,$$

(3.27)

$$X = (A^T A)^{-1} A^T L,$$

(3.28)

Then we can add the $X$ to the initial values of unknowns $(\omega_0, \phi_0, \kappa_0, X_{L0}, Y_{L0}, Z_{L0}, X_{A0}, Y_{A0}, Z_{A0})$ to iteratively optimize until the sum of $L$ close to 0. This is a typical way to get the solution for many non-linear problems. In this dissertation, besides the camera calibration, we also use this method to calculate point 3D locations and some other problems.

### 3.4.3 Epipolar Geometry

Using the epipolar line to find corresponding points in multiple cameras is efficient and powerful. In the stereo camera, the two camera centers and the object would form a plane...
which is called the epipolar plane. The intersection line of the epipolar plane with an image is called the epipolar line which can be seen in Figure 3.13.

![Figure 3.13. The illustration of epipolar geometry for stereo camera.](image)

Epipolar geometry is used to find corresponding image points in stereo cameras which reduces a two-dimensional search to a one-dimensional search along with the epipolar line. (More details in [103]). As illustrated in Figure 3.13, camera center \((L_1, L_2)\), and the object \(A\) define the plane with red lines. Point \(a_2\) is a projection of \(A\) in the image plane of camera 2. Only from \(a_2\), the depth information of object \(A\) is unknown. However, it's definitely that object \(A\) should be along the ray which is through point \(L_2\) and \(a_2\). So, no matter how far the object is from \(L_2\), the projection of object \(A\) on image from camera 1 must be in the plane that defined by \(L_1, L_2,\) and \(A\). So, if the object \(A\) has a projection on image 2, its location must be on the intersection line \((e_1, a_1)\) which caused by the image.
plane and the epipolar plane. In reverse, if the interesting points in image 2 is at $a_1$, then, its corresponding points should be along the epipolar line $(e_2, a_2)$. 
Chapter 4: Efficient Tracking with Distinctive Target Colors and Silhouette

Target tracking using color based appearance models is quite popular in visual tracking. However, trackers based only on color are fragile and often drift to the background when it has a similar appearance. In this chapter, we propose an efficient way to use distinctive target colors to track the target and reduce the drift problem. In this method, colors are sampled from the target and its immediate surrounding region. The colors that are more coming from the target are more distinctive.

The approach proposed in this chapter uses a short and a long-time color histogram to represent the target color. The short-time color histogram is used to calculate the distinctiveness of colors while the long-time color histogram is used to keep target color that is consistent over time. Furthermore, in this approach, the target is not considered as a rectangle or other geometric primitives. Instead, the tracker tracks the target with its silhouette. Using silhouette to mark target can significantly reduce the false positive information during online learning. Also, color models are updated with a dynamic learning factor which is based on the tracking performances. After testing with many tracking sequences and comparison with other state-of-art trackers, the proposed tracking algorithm shows comparable performance with very high tracking rate, especially for non-rigid objects.
4.1 Introduction

Although there are various feature descriptors, color is still one of the most popular ones and it is naturally invariant to translation, rotation and scale changes. Color can be used in two kinds of representation models in visual tracking. One is the generative representation which studies the complete target appearance. These trackers typically use appearance templates to find the target that highly fit the appearance template during tracking [23, 32, 105, 106, 107]. Two of the most popular color based generative trackers are the mean shift tracker with adaptive scale [108], and the adaptive color-based particle filter [109]. However, this generative representation is fragile to cluttered background. When the background regions have similar colors, the trackers are easily distracted by them and failed the tracking. To resolve this problem, many methods involved the idea of using discriminative representation to get better performance. These methods focus on discriminative information which can distinguish the target from background [11, 27, 31]. Discriminative trackers typically use classifiers, like SVM. Unlike these methods, the proposed algorithm in this chapter combines the generative and discriminative information during the tracking without explicitly using a classifier. Hence training and updating steps for the classifiers are also eliminated which can greatly boost the tracking speed. In the literature, some color based trackers also didn’t use classifiers to get discriminative information. Collins et al. [8] uses a modified histogram representation to estimate the color distribution for background and target separately. In [110], Zhang et al. use background and foreground color histograms to classify pixels. Their method has a similar idea like the work here to find the distinctive target color. But the work in this
chapter only uses one histogram to measure or classify the color. Moreover, in their papers, the long-term and short-term color histograms are mixed to represent the appearance model. Unlike above mentioned method, the proposed short-time model is used to record the distinctive color that mostly exist on the target. The long-time model keeping the general color of the target is used to retrieve the target. In contrast to other methods, the proposed method doesn't predict the distractors, but directly counts the discriminative colors in the target's immediate surrounding area and use the whole target color histogram to estimate the target location and regions. Also, the method segments the target from the background and uses its silhouette to mark the target region instead of using a rectangle or an ellipse. The silhouette will significantly remove false positives during the target model update.

There are several advantages of the proposed tracking method. The tracker is at the pixel level which makes the processing very fast. Besides, the color models are updated during the tracking with a dynamic learning factor which depended on the tracking result. In summary, the contributions of this tracking method include (1) a novel and efficient color model to do object visual tracking. (2) Estimating target silhouette to identify the target during the tracking and model updates. (3) Offer a way to check the track-ability of the tracker and dynamically update color models.
Figure 4.1. Illustration of the tracking flow. The dashed box represents the initialization of the color model from the target rectangle (red) and its surrounding area (the whole image). The histogram based mode represents the color target. The lower row illustrates the tracking in new frames and updating the color model.

4.2 Methodology

This section will first present how to model and find distinctive target colors. Then the initialization, segmentation, and model updating, as well as target detection, are discussed.

4.2.1 Distinctive Target Colors and Target Model

To estimate the likelihood of colors belonging to the target or the background, this work introduces a voting method by sampling colors from target $T$ and its surrounding area $B$. 
For a given color $C$, we can count how many colors are from the target and background area. As shown in Figure 4.1, the likelihood of the color $C$ to be target or background can be estimated as:

$$V(C) = \frac{1}{n} \sum_{j=0}^{n} \beta^{+,-}(f_j),$$

(4.1)

where $\beta^+(f_j) = \{1|f_j = \text{target}\}, \beta^-(f_j) = \{-1|f_j = \text{background}\}$, $n$ is the number of pixels $f_j$ which have same color $C$ in the voting, and $-1 \leq V(C) \leq 1$. The closer the $V(C)$ to 1, the more the color represents the target; otherwise, the closer $V(C)$ to $-1$, the more it represents the background, like the upper row in Figure 4.1 shows.

Two models are used to represent the target, the long-time model and the short-time model. The long-time model is used to record the color of the target in order to protect the tracker from drifting and retrieve the target. The short-time model is used to record the distinctive target colors. The likelihood of ambiguous colors has lower value in short-time model, and this can help remove confusing colors from tracking. The histogram is used to represent the two target models. The long-time model only accounts for colors that are treated as target color. Since the initial target is given by a rectangle in most tracking test, some background is included as the target. The same idea of finding distinctive target colors are used to find the true target color by using a small surrounding area $B_s$ like the lavender rectangle in the first image in Figure 4.1. The histogram is combined with $m^k$ bins, where $k$ is the number of dimensions in the color space. Finally, we can get target color histogram as:

$$H_t(b) = \frac{1}{z} \sum V(c), c \in b,$$

(4.2)
where $H_l$ is the long-time color histogram, $Z$ is normalization factor, $b$ is a bin in the histogram. During tracking, the target histogram will be updated dynamically. For short-time color histogram, a larger surrounding area $B$ is used with same equation. Here, the $B_s$ is set to 1.5 times target box and $B$ is set to 2 times.

4.2.2 Model Checking and Initialization

In the beginning, there is no previous knowledge about which pixels belong to target or background. But we can assume pixels inside the initial target box belong to target otherwise belong to background. Then we can use equation 4.1 to calculate the color likelihood. Since the tracking algorithm is only based on color, if the target and background have similar colors, tracking will be infeasible. In order to quantitatively check the initialization, we test how distinctive the colors are by computing track-ability score:

$$F_e(B_x, I) = \frac{s_t - s_b}{s_x},$$  \hspace{1cm} (4.3)

where $B_x$ is the initial target box, $I$ is the image, $S_x$ is the size of initial target box, $S_t$ is the size of regions identified as target in the target box, $S_b$ is the target region identified as target outside the target box (see Figure 4.2 for illustration). The more distinctive the target colors are, the more likely the target can be tracked by its distinctive features. We can divide the score into three scenarios:

$$F_e(B_x, I) = \begin{cases} > threshold_1, & \text{Trackable} \\ > threshold_2, & \text{May be tracked} \\ \text{otherwise, Cannot be tracked} & \end{cases}$$  \hspace{1cm} (4.4)

experientially, we chose $threshold_1$ as 0.5 and $threshold_2$ as 0.2. The track-ability score only considers the initial frame and is not necessary to compute all the following.
frames. However, we still prefer to check and make a decision if it is suitable to track the target for current scenario.

![Image](image.png)

(a) (b)

Figure 4.2. Illustration of the distinctive target color. (a) shows the high trackability with many distinctive target colors and (b) shows the low track-ability with little distinctive target colors in HSV color space.

After checking the track-ability, we need to refine the long-time model and short-time model by the initial target color model from equation 4.2. Since a rectangle gives the initial target in most tracking test sequences, some background is marked as the target, like Figure 4.1 shows, we need to generate the target silhouette. So, in the initial frame, we find each pixel’s target likelihood based on long-time model and generate a confidence map like Figure 4.1 and Figure 4.3. Following this step, the confidence map is used to segment the target silhouette (more details in next section). With the target silhouette estimated, the short and long-time color models can be refined by doing the color likelihood one more time.
4.2.3. Target Tracking and Segmentation

When a new frame arrives, a target search window is subtracted respecting to last target’s position. This algorithm assumes that the target can’t move too fast such that it cannot get out of the surrounding window in the last frame. So, we treat the same window as the search window in the new coming frame. This surrounding window is normalized to a predefined size to eliminate negative effects of scale changes during model computation. Then, each pixel in the normalized surrounding window is assigned two confidence values which respectively depend on the short-time model and the long-time models:

\[ v^{s,l}(c) = H^{s,l}(b_c), \]  

where \( v^{s,l}(c) \) is the confidence value of pixels with color \( c \) for short and long-time model separately. The short-time confidence map is used to find the location of the target. Since the short-time color model treats immediate target color variation, it will prevent the tracker drifting to distractive color in the background. As shown in Figure 4.3, image (b), the blue target color is deemphasized since similar colors are shared with at the upper left. After removing the confusing colors, the positions (target candidates) with higher confidences based on the short-time confidence map suggests potential target regions. We test these multiple positions by using the integral image formulation \([111]\) to reduce computational cost. Under the small motion assumption, we impose a spatial Gaussian model to punish large motion. Finally, the long-time confidence map is used to estimate the target silhouette. Since the long-time model considers all the target colors at once which may include many cluttered colors, we correct the values on long-time confidence map by:
\[
v^l(c) = a * v^l(c) + (1 - a) * v^s(c),
\]

(4.6)

where \(v^l(c)\) is the corrected confidence value and \(a\) is the weight factor. Hence, the long-time confidence map depresses ambiguous colors and highlight distinctive target colors with higher values. The confidence map is used to find the connected regions which are the best candidate.

Figure 4.3. The tracking and segmentation. (a) is the search image in a new-coming frame with the tracked target rectangle. (b) is the short-time confidence map used to find target position. (c) is the long-time confidence map used to find the target regions. (d) is the segmentation results of the target. (e) shows the segmentation used to mask the target in the updating color models.

After post processing with a morphological filter, the resulting regions are marked as target silhouette, (see Figure 4.3, image (d)). The silhouette can represent target during the tracking and updating steps. In order to compare to other trackers’ performances, we use circum-rectangle to represent the target. But we constrain the shape and size changes based on the previous target rectangle.
In some situations, the segmented regions are less than the best target candidate which means the segmentation may be not correct. For this, we use the best target candidate as the target rectangle in the update and tracking.

4.2.4. Tracking Evaluation and Model Updating

Each time, after getting the final target box, we generate a surrounding image and calculate the color likelihood with the new silhouette. Then we compare the new long-time model $\hat{H}$ with old long-time model $H_{\text{target}}$ to see how the target is preserved in the new frame. In the case the tracking is perfect and the target appearance does not change, we expect a performance measure $\Lambda$ to give the highest possible similarity between $H_{\text{object}}$ and $\hat{H}$. In this paper, we compute the performance measure by:

$$\Lambda(H_{\text{object}}, \hat{H}) = \frac{1}{n} \sum_{i=1}^{n} \delta(H_{\text{target}}(b_i), \hat{H}(b_i)),$$  \hspace{1cm} (4.7)

Where $\delta(\cdot)$ is delta Dirac function to estimate the similarity of two histograms, and $\Lambda(\cdot)$ ranges from 0 to 1 respectively denoting bad to good performance. As an example of how this measure works, let’s assume the target is partially occluded. Depending on the amount of occlusion, the performance measure will drop to a very small value signifying that the model should not be updated and even the target is assumed as lost. Otherwise, an update to the target representation may be necessary to provide updates to the changing appearance. We achieve the representation update by updating each color model using:

$$H_{\text{new}}(b_i) = \Lambda^k \hat{H}(b_i) + (1 - \Lambda^k)H_{\text{target}}(b_i),$$  \hspace{1cm} (4.8)

where the performance measure serves as the learning factor for the target in the new frame.
4.2.5. Lost Targets and Re-detection

Figure 4.4. Illustration of evaluation of tracking result and detection the target again. (a) Tracking results treat as part detected. (b) The lose of tracking. (c) The detection of the target after it been lost several times. Upper left blue box inside (c) is a zoom image of the smaller rectangle in the middle. (d) Detected the target again with long-time and short-time model.

If the tracker lost the target after several times, then the same tracking algorithm can be used to detect the target at the whole image. Each pixel’s color likelihood will be calculated like the corrected long-time confidence value. Then the tracking algorithm will
be used to find the target as well as estimation the result. If the detected object is similar to a target, the updating followed, Figure 4.4 gives the detection example. Otherwise, the tracker still losing the target and waiting for the next time. If there are several target candidates, the previous target location will be used to help find the closest candidate as primal one.

4.3 Experiment

4.3.1 Experimental Setup

Our algorithm uses the HSV color space and a histogram with 12*12*5 bins (H, S, V). The surrounding area for distinctive color is set as three times target while for long-time model is 1. In equation 4.5, \(a\) is set as 0.5. And during the tracking, the environment window is normalized into 120*120 pixel. The software runs on a laptop with 2.2 GHz CPU and 12GB memory with single thread without further optimization and we observe tracking at highest as 126 frames per second. We evaluate the proposed algorithm with 18 challenging sequences from benchmark datasets used in other studies. Tracking results are compared with several state-of-art tracking methods, including TLD [3], Struck [32], MIL [36], II [112], Frag [113], IVT [6], and Pixel Track [114]. We use two typical evaluation criteria in the experiments. They are Center Location Error (CLE) and Pascal VOC [115] Overlap Ratio to find the number of successfully tracked frames.

4.3.2 Tracking Performance

1) Background Clutter: When the target and background have similar colors, the tracking will be challenging. In our algorithm, the short-time model will depress these
similar colors and only focus on distinctive target colors. Besides this, the motion model based on Gaussian distribution can also constrain the target to a nearby place and keep them from draft moving. As shown in Figure 4.5, the two bottles of liquor have similar color, the tracked target box is in the middle of them when they are close, after the short-time model learned the background, the importance of the liquid color would be lower and make the tracking back to the relative black part.

Figure 4.5. Tracking in cluttered background.
2) **Non-rigid Deformation**: A large change in the target’s shape will make some trackers lose target if they model spatial information. Unlike many other trackers, the proposed algorithm didn’t include any geometric information of the target. Hence the algorithm has the advantage to track non-rigid objects. As shown in Figure 4.6, the shape of the gymnast changes significantly in the image sequences. Since the color didn’t change, the tracker still finds the gymnast with most of her body.

![Figure 4.6. Tracking a non-rigidly deforming object.](image)

3) **Occlusion**: When occlusions happen, the evaluation score of the target region will be lower than non-occluding situations. In this case, the learning factor will also become
smaller such that it does not damage the distinctive color models. On the other hand, when the occlusion is over, long-time color model can retrieve the target back. Figure 4.7 shows the occlusion scenarios. If the tracking results are treated as part detected, the estimated target rectangle will not change and waits for next frame. In the case, when the tracking decides that the target to lost, the tracking will be extended to the whole image frame to find the target again. For color based tracker, the illumination change can also be treat as occlusion. As Figure 4.8 shows when there are strong illumination changes, the tracker will treat the tracking results as part tracked or lost and wait the illumination become normal to find the target again.
4) Target Loss and Re-detection: At times, it is inevitable that the tracker will lose the target. Since we still keep the target colors for both long-time and short-time, and these color models are not damaged before lost the target due to the updating algorithm, we can use them to find the target again. As shown in Figure 4.7, the target is retrieved after the tracker loses it and the target reappears.
5) **Comparison with other trackers:** We compared our algorithm with several state-of-art trackers. Table I shows the mean center location error of the tracking results, while Table II shows the ratios of successfully tracked frames respect to the VOC detection criterion with threshold larger than 0.5 [115]. We also compared our algorithm with Pixel tracking [114] which has very good performance for the non-rigid object and the Table 4.3 shows the results. From Table 4.1 we can see that the proposed algorithm has very good performances in almost all sequences. For a larger object, the error is larger, like transformer and board, but for a smaller target, our algorithm can get a much higher precision, like bolt, bird. We believe the reason for this is that tracking based on a pixel
may be affected by noise, especially when the colors change in consecutive frames. However, using silhouette in the tracking and updating steps can limit this effect.

Figure 4.9. Tracking comparison.

From Table 4.2, we can find that the proposed algorithm is among the best trackers. For target which have sharp areas and irregular shapes, the proposed tracker doesn’t perform the best. This may be related to the fact that the proposed algorithm may discard sharp target parts. The target is identified by the silhouettes marked after segmentation step followed by a morphological filter. The filter removes sharp parts and keeps the main body of the target. Then, if the sharp parts are outside the target silhouette, this parts will
be finally removed from our color model. For this reason, for irregular targets, our algorithm can only keep the regular part which will make the tracked target smaller like Figure 8 shows.

Table 4.1. The center location error. (red as best, blue as second).

<table>
<thead>
<tr>
<th>sequence</th>
<th>IVT</th>
<th>Frag</th>
<th>MIL</th>
<th>VTD</th>
<th>TLD</th>
<th>Struck</th>
<th>ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemming</td>
<td>14</td>
<td>84</td>
<td>14</td>
<td>98</td>
<td>104</td>
<td>134</td>
<td>9</td>
</tr>
<tr>
<td>liquor</td>
<td>238</td>
<td>31</td>
<td>165</td>
<td>155</td>
<td>28</td>
<td>124</td>
<td>8</td>
</tr>
<tr>
<td>Singer1</td>
<td>5</td>
<td>20</td>
<td>20</td>
<td>3</td>
<td>5</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>basketball</td>
<td>120</td>
<td>14</td>
<td>104</td>
<td>11</td>
<td>170</td>
<td>153</td>
<td>7</td>
</tr>
<tr>
<td>woman</td>
<td>133</td>
<td>112</td>
<td>120</td>
<td>109</td>
<td>95</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>transformer</td>
<td>130</td>
<td>47</td>
<td>33</td>
<td>43</td>
<td>23</td>
<td>54</td>
<td>13</td>
</tr>
<tr>
<td>bolt</td>
<td>382</td>
<td>100</td>
<td>380</td>
<td>14</td>
<td>90</td>
<td>387</td>
<td>6</td>
</tr>
<tr>
<td>Bird1</td>
<td>230</td>
<td>223</td>
<td>270</td>
<td>250</td>
<td>77</td>
<td>148</td>
<td>6</td>
</tr>
<tr>
<td>Bird2</td>
<td>119</td>
<td>28</td>
<td>18</td>
<td>50</td>
<td>86</td>
<td>88</td>
<td>6</td>
</tr>
<tr>
<td>girl</td>
<td>184</td>
<td>106</td>
<td>55</td>
<td>57</td>
<td>151</td>
<td>119</td>
<td>12</td>
</tr>
<tr>
<td>Surfing1</td>
<td>141</td>
<td>199</td>
<td>319</td>
<td>84</td>
<td>27</td>
<td>265</td>
<td>12</td>
</tr>
<tr>
<td>racecar</td>
<td>340</td>
<td>94</td>
<td>104</td>
<td>196</td>
<td>134</td>
<td>202</td>
<td>8</td>
</tr>
<tr>
<td>gymnastics</td>
<td>15</td>
<td>9</td>
<td>115</td>
<td>-</td>
<td>12</td>
<td>-</td>
<td>10</td>
</tr>
<tr>
<td>board</td>
<td>165</td>
<td>45</td>
<td>66</td>
<td>105</td>
<td>-</td>
<td>-</td>
<td>29</td>
</tr>
</tbody>
</table>
Table 4.2. The successful tracking ratio. (red as best, blue as second).

<table>
<thead>
<tr>
<th>sequence</th>
<th>IVT</th>
<th>Frag</th>
<th>MIL</th>
<th>VTD</th>
<th>TLD</th>
<th>Struck</th>
<th>ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>lemming</td>
<td>0.79</td>
<td>0.51</td>
<td>0.83</td>
<td>0.35</td>
<td>0.27</td>
<td>0.49</td>
<td>0.83</td>
</tr>
<tr>
<td>liquor</td>
<td>0.23</td>
<td>0.79</td>
<td>0.20</td>
<td>0.27</td>
<td>0.80</td>
<td>0.23</td>
<td>0.74</td>
</tr>
<tr>
<td>Singer1</td>
<td>0.93</td>
<td>0.25</td>
<td>0.25</td>
<td>1.00</td>
<td>1.00</td>
<td>0.25</td>
<td>0.81</td>
</tr>
<tr>
<td>basketball</td>
<td>0.11</td>
<td>0.71</td>
<td>0.28</td>
<td>0.85</td>
<td>0.06</td>
<td>0.12</td>
<td>0.98</td>
</tr>
<tr>
<td>woman</td>
<td>0.08</td>
<td>0.07</td>
<td>0.28</td>
<td>0.85</td>
<td>0.06</td>
<td>0.56</td>
<td>0.58</td>
</tr>
<tr>
<td>transformer</td>
<td>0.23</td>
<td>0.31</td>
<td>0.24</td>
<td>0.38</td>
<td>0.35</td>
<td>0.27</td>
<td>0.71</td>
</tr>
<tr>
<td>bolt</td>
<td>0.01</td>
<td>0.09</td>
<td>0.03</td>
<td>0.57</td>
<td>0.14</td>
<td>0.03</td>
<td>0.81</td>
</tr>
<tr>
<td>Bird1</td>
<td>0.01</td>
<td>0.11</td>
<td>0.29</td>
<td>0.02</td>
<td>0.06</td>
<td>0.04</td>
<td>0.22</td>
</tr>
<tr>
<td>Bird2</td>
<td>0.09</td>
<td>0.43</td>
<td>0.83</td>
<td>0.09</td>
<td>0.12</td>
<td>0.14</td>
<td>0.70</td>
</tr>
<tr>
<td>girl</td>
<td>0.07</td>
<td>0.42</td>
<td>0.38</td>
<td>0.55</td>
<td>0.11</td>
<td>0.16</td>
<td>0.94</td>
</tr>
<tr>
<td>Surfing1</td>
<td>0.08</td>
<td>0.10</td>
<td>0.03</td>
<td>0.08</td>
<td>0.40</td>
<td>0.08</td>
<td>0.40</td>
</tr>
<tr>
<td>racecar</td>
<td>0.02</td>
<td>0.15</td>
<td>0.04</td>
<td>0.06</td>
<td>0.03</td>
<td>0.07</td>
<td>0.67</td>
</tr>
</tbody>
</table>

In addition, we also perform comparisons with pixel tracking. Pixel tracking is an algorithm for tracking with pixels and has a very good performance for the non-rigid object. In Tables 4.3, we compare the two trackers. As we can see, for all four test sequences, our algorithm both has equal or better results. Pixel tracking still includes spatial information of the target during the tracking. We think of the object with a high degree of deformations, spatial information does not help to find the target and lowers its performance.
Table 4.3. The proposed algorithm VS pixel track for tracking ratio.

<table>
<thead>
<tr>
<th>sequence</th>
<th>Pixel track</th>
<th>proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>skiing</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>diving</td>
<td>0.89</td>
<td>0.95</td>
</tr>
<tr>
<td>high jump</td>
<td>0.94</td>
<td>0.99</td>
</tr>
<tr>
<td>Gymnastics</td>
<td>0.99</td>
<td>1.00</td>
</tr>
</tbody>
</table>

4.4 Conclusion

In this chapter, a novel way of using color models has been proposed to improve target tracking. The algorithm contains two tracking models, long-time and short-time color model. Long-time model is used to record the target colors and use to retrieve the whole target. Short-time model is counting the target and background color at the same time, and help to find the distinctive target color to track the target robustly. In addition, a dynamic update factor is calculated by the tracking results which make the tracking work for situations as part detection and lost. Since the tracker has no spatial information in its formulation, it works well for non-rigid objects. In our algorithm, we offer a way to check the color tracker at initialization with the track-ability score to help decide whether the tracker is suitable for the scenarios. By test and compares with other trackers, the new tracker shows better or comparable performance at very high rate.
Chapter 5: Unique Feature Vote Tracking

One of the main problems in target tracking is using a unique set of features that species the target. In this chapter, we propose a simple yet efficient method that provides unique "target representation" by generating discriminative non-uniform subspaces from the descriptor space which we refer to as cells. Each cell is attributed with a measure that highlights how likely it describes the target or the background. In addition, we keep a codebook of spatial locations of the descriptors which are mapped to the cell similar to that of the R-Table in generalized Hough transform. Using the uniqueness measure as weight, the target center is estimated by using a modified Hough voting scheme to address non-rigid deformations. Without the loss of generality, we use color as pixel's descriptor and show good performance on the Online Tracking Benchmark (OTB) data set. The experiments show promising tracking performance on occlusion and clutter also other sequences compared to the state-of-art trackers.

5.1 Introduction

Many of the methods, especially the ones based on tracking-by-detection, they only consider the target [2-6]. Despite their concentrated focus on the target features, they have shown good tracking performance [116,117]. These trackers, however, are vulnerable to background clutter and partial or full occlusions. In order to offset this problem, other tracking approaches consider modeling the distinctiveness between the
target and its immediate background [7-11]. These approaches typically train a classifier with labeled background and foreground data to explicitly discriminate the target from the background. In many cases, the classifier based approaches outperform generative models when adequate training data is available [24, 25].

Aside from target tracking, in context target detection and target recognition, Generalized Hough Transform (GHT) had great promise [118, 119]. Lei et al. [118] constructed a codebook via unsupervised clustering of local appearance that implicitly contains the local structure information of a target category. Other methods, such as discriminatively trained patch-based models (DPM) and consensus of exemplars [4, 120] have extended GHT to improve detection performance. Godec et al. [121] have extended the use of Hough transform for non-rigid target tracking by coupling the voting step with segmentation to find the target area. Similarly, authors of [122] and [114] have respectively used key points and pixels to generate vote in the context of tracking.

In this chapter, we seek the answer to the problem of finding a method to generate discriminative information from limited data while eliminating the need to train classifiers. We achieve this by treating the descriptor space as a classifier by non-uniformly dividing it into some cells, and use the measure generated from the mapped descriptors as weight in a voting scheme to locate the target. This appearance-based model can be augmented by including spatial information in the form of a codebook assigned to each cell to facilitate a target center voting scheme. For each new frame, the tracking process starts by generating a descriptor for each pixel, which is mapped to the corresponding cell that specifies descriptor's likelihood to be target or background. This
information is then used along with the spatial codebook to generate a number of votes as target center candidates similar to that of a generalized Hough voting scheme [118, 119]. Once the center is estimated, we use a confidence map to find the bounding box which includes the target. Finally, each cell likelihood measure is updated to model target and background changes. The flow diagram of the tracking algorithm is given in Figure 5.1.

The main contributions of this proposed method are as follows: 1) offers a new approach to estimate the uniqueness of target and background descriptors, and 2) Introduces a joint appearance and spatial voting scheme to estimate the target center

![Flow diagram of the proposed tracking framework](image)

Figure 5.1. The flow diagram of the proposed tracking framework. For each new frame, we extract descriptors for features within the search window (white rectangle) generated based on past target location. The mapped cells and their likelihood measures are used to generate a target center voting map and an appearance confidence map (red circle means the feature from the target, the blue dot means the feature from the background). Finally, the descriptors within the tracked bounding box are used to update the cells.

5.2. Methodology

The goal of this tracking method is to extract and track a unique representation estimated from the descriptor space of a target. Unlike the previous work, we combine feature
selection and clustering steps together. This is achieved by first intelligently dividing the
descriptor space into non-uniform no overlapping cells. Each cell, similar to a cluster
center, describes either target or background. The likelihood measure of a cell is
computed from the number of labeled descriptors that are mapped to the cell. In order to
model spatial information of the mapped descriptors, we use a codebook for each cell,
which is used to vote for the target location in the new frame.

2.2.1. Unique Target Representation

We define unique target representation as a set of cells in the descriptor space that
respectively contain the uniqueness of target and background and a spatial codebook. The
cells relate to subspaces which are generated by recursively dividing the space until each
cell either represents the target or background or neither (see Figure 5.2 for mapping of
pixel descriptors to cells making up space). The cells contain two sets of observations
representing the target and the background descriptors. Considering that a cell $C_i$ contains
the descriptors representing either target or background or neither, it can be considered a
cluster. Hence, the task of extracting unique representation becomes finding unique
clusters.
Figure 5.2. The cells in the descriptor space are generated based on the labeled pixels within the target bounding box (smaller white box), and its surrounding (larger white box). The descriptors generated for each labeled pixel (colored circles inside the boxes) are used to divide the descriptor space into cells to either represent target or background, or neither of them.

Let the target be composed of a set of pixels with descriptors $T = (f_1, f_2, f_3, \ldots)$ and its immediate background contain pixels with descriptors $B = (f_1, f_2, f_3, \ldots)$. A descriptor can be defined as simple as color, or it can be more complicated, such as SIFT, ORB or HoG descriptors. Let the initial target region be marked in the first frame, such that we have initial labels $l_i$ for each pixel descriptor $f_i$. Each descriptor $f_i$ generates a vote in the feature space in favor of its label $l_i$. Then, consider a vote in favor of the target a positive vote $\beta^+(f_i) = \{1|l_i = target\}$. On the contrary, a background vote is considered a negative vote $\beta^-(f_i) = \{-1|l_i = background\}$ which in turns unlearns the cell. As a consequence, the uniqueness of a cell is measured by:

$$d(C_i) = \frac{1}{n} \sum_{j=0}^{n} \beta^{(\cdot,+)}(f_i).$$ (5.1)
where $n$ is the number of background and target labels mapped to the cell. Similar to the equation 4.1, the closer $d(C_i)$ to 1, the more the cell represents the target; otherwise, the closer $d(C_i)$ to -1, the more it represents the background. Hence, a unique target representation is generated as the set of cells that have positive uniqueness:

$$U_{\text{target}} = \{C_i | d(C_i) > \theta_u\},$$  \hspace{0.5cm} (5.2)

where $\theta_u$ represents the threshold that is used to select unique cells. In this model, robust target tracking can only happen when the unique representation $U$ contains many cells with values closer to 1. This requirement can be easily met when the cells are smaller, i.e. small space quantization-bandwidth that will result in higher uniqueness value per cell. Small cell size, however, causes overfitting problem and is not robust to noise in data. Hence, we resolve this problem by introducing a divide and conquer strategy to dynamically partition the feature space into cells of different sizes, as the follow algorithm:

**Non-uniform Cells Generation**

**Input:** $C_i$

**Output:** \{\(C_j \land l_j \mid C_j \in C_i \land l_j = \text{target, background}\)\}

1. **Compute** $d(C_i)$
   
   If $d(C_i) \approx 0$ and $N_i > N_t$, then Divide $C_i$ into $2^k$ cells, repeat 1 for each cell.
   
   Else
   
   If $d(C_i) \approx 1$ then
   
   $l_i = \text{target}$
   
   Else
   
   $l_i = \text{background}$
   
   End if
   
   End if
The algorithm starts by evenly dividing the feature space into $2^k$ cells, $k$ is the dimensionality of the feature space. The algorithm proceeds by checking a cell's uniqueness against threshold, $\theta_u$, and recursively divides it into $2^k$ sub-cells until the uniqueness criteria is met. The recursive division is concluded when the number of mapped descriptors $N_i$ is less than $N_t$ or its uniqueness is larger than $\theta_u$. The pseudo code of the divide and conquer strategy is given in the above algorithm.

**Checking Track-ability**

In order to have a discriminative representation, it is imperative to have cells that support the target appearance, and that a good number of pixels that map to these cells. For this purpose, we check the number of descriptors that mapped to the selected cells upon generation of the target representation using the same idea and equations in Model Checking and Initialization in section 4.2.

We should note that checking track-ability is not absolute and the higher score does not always mean guaranteed tracking for the entire sequence. Hence, the track-ability score is only considered for initialization in the first frame.

**5.2.2. Target Center Voting**

Despite being discriminant in identifying appearance difference between the target and the background, the target representation stated above does not encode spatial information regarding where each observation resides about the target position. In order to alleviate this shortcoming, we consider encoding the spatial information, $s_i = (u_i, v_i)^T$ carried by each descriptor $f_i$ relative to target’s center. Considering that a cell contains multiple descriptors, their respective spatial locations generate a spatial
codebook for the cell. The spatial codebook serves as a look-up table similar to the R-Table GHT and provides a voting mechanism to locate the target center in the new frame. In contrast to the GHT voting, we consider a weighted voting approach which relates to the uniqueness of the target representation. In Figure 5.1, we display the voting surface for the target. The coordinates with the highest vote represent the target center. For the tracking in the new frame, we estimate the best location of the target center $u$ by:

$$score(u) = \sum_i d(C(f_i)) \sum_{s_k \in C(f_i)} v(u, s_k),$$  \hspace{1cm} (5.3)

where $C(f_i)$ is the cell which $f_i$ belongs to. $v(u, s_k)$ is indication function, when $s_k$ vote to $u$ it returns 1, otherwise returns 0.

In some cases, it may be possible that the spatial codebook related to the pixel may not contain precise target center locations due to out-of-plane rotations or non-rigid motion. Such changes in codebook may result in imprecise voting in the target center estimation stage. We overcome this shortcoming by generating weighted spatial voting using Gaussian weighting. For a center candidate, we introduce a weighted voting score where the weight is decided by its distance from the target center candidate:

$$\hat{u}_t = \arg \max(score(u_i)), \hspace{1cm} (5.4)$$

$$score(u_i) = \sum_{u_k \in u_i} w(u_k) \cdot score(u_k), \hspace{1cm} (5.5)$$

$$w(u_k) = \frac{1}{\sqrt{2\pi\sigma}} \exp(- (u_k - u_i)), \hspace{1cm} (5.6)$$

where $u_i$ is the target center candidate, $u_k$ is pixels which around $u_i$. Overtime, the target appearance can undergo changes. Such changes require a mechanism to update the target appearance model. There are three possible cases that can happen all of which relate to changes in the appearance. For each case, the cell may need to be updated to reflect the
changes based on new descriptors observed in the new frame. The first of these is a cell becoming not representative of the target. The second one is a new cell becoming member of the target representation. The last case is when there is a small change in the appearance resulting in slight variation in cell uniqueness. For unique cells, the number of descriptors mapped from the target shouldn’t drastically change. Based on this assumption, if

\[
\frac{\text{Size}(C_{\text{old}})}{\text{Size}(C_{\text{new}})} > \theta_h, \text{ or, } \frac{\text{Size}(C_{\text{old}})}{\text{Size}(C_{\text{new}})} < \theta_l, \tag{5.7}
\]

the unique features in cell \( C \) is considered not to be trusted and are removed from the representation. Here \( \theta_h \) and \( \theta_l \) are high and low threshold defining allowed size changes.

5.2.3. Target Tracking

The target center estimation discussed in the previous section provides the target location but its scale. In order to estimate the target scale, we consider the reverse of center estimation and generate a confidence map from the appearance and spatial information encoded in the target representation.
Figure 5.3. The spatial codebook kept for each cell in the unique target representation. The left figure illustrates a 2D descriptor space. The middle figure represents a number of color coded descriptors computed for target pixels. Their mapping to the feature space introduces an entry for the spatial codebook. For a new frame shown on the right, we estimate the corresponding descriptors and back project the code book to generate votes for the target center.

Let $u_t$ be the new target center and $u_{t-1}$ be the target scale at frame $t - 1$. Each pixel $(u_i, v_i)$ within the search window centered around $u_t$ generates a descriptor $f_i$ with uniqueness value $\theta_u \leq d(C_i) \leq 1$. In addition to appearance, the spatial codebook for $C_i$ provides all acceptable displacements $s_j$ for the mapped descriptor. These domain $s_j$ and range $d(C_i)$ information provide a means to compute the likelihood of the pixel to be a part of the target or the background:

$$v(f_i) = d(C_i) \times \exp\left(-\frac{(u_i-\hat{u})^2+(v_i-\hat{v})^2}{2\sigma^2}\right),$$

(5.8)

where $(\hat{u}, \hat{v}) = \min_{(u_j, v_j)} (u_i - u_j)^2 + (v_i - v_j)^2$ for $(u_j, v_j) \in s_i$ and $-1 < v(f_i) < 1$.

This likelihood value will be computed for each pixel in the new frame which results in a confidence map.
Let the state of the tracked target be composed of \((s_x, s_y, u_x, v_x)\), where \(s\) is scale per axis and \(u\) is the target centroid. At each new frame, the target state initialized by its previous scale and the target centroid estimated in the previous section. The true target state then becomes the state that maximizes the posterior estimate given by:

\[
\hat{X}_t = \arg \max p(X_t | Y_{1:t}) = \arg \max p(Y_t | X_t) \int p(X_t | X_{t-1}) p(X_{t-1} | Y_{1:t-1}) dX_{t-1}, \quad (5.9)
\]

where the new state \(\hat{X}_t\) depends on past previous observations \(Y_{1:t}\) of the target. It can be noted that this formulation can be divided by two parts: the observation model \(p(Y_t | X_t)\) and the motion model \(p(X_t | X_{t-1})\). For the observation model, we use the confidence map:

\[
p(Y_t | X_t) \propto C(X_t).
\]

As for the motion model, we consider the flow of the target center \((\dot{u}_x, \dot{u}_y)\) and changes in the target scale \((\dot{s}_x, \dot{s}_y)\) which is modeled by normal distribution

\[
p(X_t | X_{t-1}) \propto N(\dot{u}_x, \dot{u}_y, \dot{s}_x, \dot{s}_y; \psi),
\]

where \(\psi\) is the covariance matrix.

Considering out of plane motion may change object size independently for each axis, and that the object motion can be assumed to be independent for orthogonal x and y axes, the covariance matrix becomes a diagonal matrix. In our implementation, we compute the maximum posteriori estimate by testing a series of possible scale changes given the target’s previous scale. The test of multiple scales can be swiftly computed using integral image formulation [111] without increasing computational cost.
5.2.4. Representation Update

Upon estimating the position and scale, proposed approach decides whether the target representation needs an update. This decision is made based on how well the previous representation, $U_{object}$ is preserved in the new frame. In order to facilitate this, we estimate a new representation, $\hat{U}$, using the new frame and the estimated target state. In the case the tracking is perfect and the target appearance does not change, we expect a performance measure $\Lambda$ to give the highest possible similarity between $U_{object}$ and $\hat{U}$. In this paper, we compute the performance measure by:

$$\Lambda(U_{object}, \hat{U}) = \frac{1}{n} \sum_{i=1}^{n} \delta(C_{U_{object}}(i), C_{\hat{U}}(i))$$

(5.10)

where $\delta(\cdot)$ is delta dirac function and $\Lambda(\cdot)$ ranges from 0 to 1 respectively denoting bad to good performance. As an example, on how this measure works, let’s assume the target is partially occluded. Depending on the amount of occlusion, the performance measure will drop to a very small value signifying that the model should not be updated. Otherwise, an update to the target representation may be necessary to provide updates to
the changing appearance. We achieve the representation update by updating each cell using:

\[ C_{new}(i) = \Lambda^k C(i) + (1 - \Lambda^k) C_{object}, \]  

(5.11)

where the performance measure serves as the learning factor for the target in the new frame.

5.3. Experiments

The proposed tracking approach does not involve specific descriptors. As stated above, all types of features can be used with the proposed tracking framework. In the experiments discussed in the following section, we use simple color as the descriptor of a pixel. The discussion in this section provides the experimental setup and tracking results for a set of challenging sequences as well as discussion.

5.3.1. Experimental Setup

The results presented in this section is based on the HSV color space where all three values are used as a descriptor. The descriptor space is initially divided into 16*16*16 cells, then a divide and conquer strategy to dynamically partitions the space into a set of non-uniform small cells with threshold \( \theta_u = 0.05 \). The sampling and search window sizes are set as 2.8 times as target bounding box. During tracking, the center voting score threshold is set to \( \theta_v = 0.2 \). The target candidates for certain scales and position is verified using a sliding search window for scale changes of 0.9 to 1.1. For the target motion model, and scale changes we consider only the last 10 recent frames, which provide Gaussian distributions.
The tracking experiment is implemented in C++ with OpenCV library and runs on a laptop with 2.2 GHz CPU and 12GB memory with single thread without further optimization at 20 Hz averagely.

We test the performance of the proposed approach using a large dataset called online tracker benchmark (OTB) [123]. Since the experiment is using color as feature description, 35 color videos of the total 50 are used in the test. The performance scores are provided from the one-pass evaluation (OPE) toolkit. There are two measurements in the evaluation: Precision plot and success plot. The precision plot is used to measure the distance between tracking results and ground truth. The success plot is used to evaluate the overlap score of them. For more details, please refer to [123]. We also compared the proposed tracker on the benchmark with other 34 popular trackers including 29 trackers in [123] and KCF [41]. Since our tracker in the experiment only uses color, we compared it with another recently color based state-of-the-art trackers: TGPR [30].

5.3.2. Experimental Result

As mentioned in section 4.1, the proposed method can check track-ability of the target for the provided sequences. We provide the track-ability of the target using the color descriptor for each sequence, Table 5.1 gives all the sequence scores. And we find out that color is not a very good descriptor for many sequences, and the unique target representation can only represent a small portion of the target. As we discussed above, if the track-ability score is less than 0.2, the algorithm does not track the object. For such sequences, one needs to use more complicated descriptors.
Table 5.1 the track-ability score of each image sequence in OTB50 [123].

<table>
<thead>
<tr>
<th>Name</th>
<th>Fe</th>
<th>Name</th>
<th>Fe</th>
<th>Name</th>
<th>Fe</th>
<th>Name</th>
<th>Fe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basketball</td>
<td>0.301</td>
<td>Crossing</td>
<td>0.407</td>
<td>Girl</td>
<td>0.463</td>
<td>MotorRolling</td>
<td>0.237</td>
</tr>
<tr>
<td>Bolt</td>
<td>0.223</td>
<td>David</td>
<td>0.357</td>
<td>Ironman</td>
<td>0.123</td>
<td>MountainBike</td>
<td>0.318</td>
</tr>
<tr>
<td>Boy</td>
<td>0.533</td>
<td>David3</td>
<td>0.368</td>
<td>Jogging</td>
<td>0.443</td>
<td>Shaking</td>
<td>0.070</td>
</tr>
<tr>
<td>CarDark</td>
<td>0.078</td>
<td>Deer</td>
<td>0.189</td>
<td>Jogging1</td>
<td>0.440</td>
<td>Singer1</td>
<td>0.558</td>
</tr>
<tr>
<td>CarScale</td>
<td>0.382</td>
<td>Doll</td>
<td>0.541</td>
<td>Lemming</td>
<td>0.447</td>
<td>Singer2</td>
<td>0.027</td>
</tr>
<tr>
<td>Coke</td>
<td>0.243</td>
<td>FaceOcc1</td>
<td>0.422</td>
<td>Liquor</td>
<td>0.520</td>
<td>Skating1</td>
<td>0.263</td>
</tr>
<tr>
<td>Couple</td>
<td>0.248</td>
<td>Football1</td>
<td>0.143</td>
<td>Matrix</td>
<td>0.323</td>
<td>Skiing</td>
<td>0.455</td>
</tr>
<tr>
<td>Soccer</td>
<td>0.128</td>
<td>Subway</td>
<td>0.327</td>
<td>Tiger1</td>
<td>0.403</td>
<td>Tiger2</td>
<td>0.544</td>
</tr>
<tr>
<td>Trellis</td>
<td>0.302</td>
<td>Walking</td>
<td>0.526</td>
<td>Walking2</td>
<td>0.425</td>
<td>Woman</td>
<td>0.514</td>
</tr>
</tbody>
</table>

Figure 5.5. Illumination change is causing tracking failure. The trackability score at the beginning of the sequence is 0.263 and is close to 0.2. This causes confusion in the following frames and results in target loss.
Figure 5.6. The tracking results from (OTB).

Continued
Figure 5.6 continued

Success plots of OPE - fast motion (11)

Precision plots of OPE - fast motion (11)
Success plots of OPE - occlusion (22)

Precision plots of OPE - occlusion (22)

Continued
Figure 5.6 continued

Success plots of OPE - background clutter (11)

Precision plots of OPE - background clutter (11)

Continued
Figure 5.6 continued

**Success plots of OPE - deformation (15)**

- ours [0.540]
- TGPR_C [0.519]
- KCF [0.489]
- SCM [0.413]
- DFT [0.390]
- CPF [0.372]
- RS-V [0.367]
- PD-V [0.385]
- Struck [0.354]
- TLD [0.347]

**Precision plots of OPE - deformation (15)**

- ours [0.708]
- TGPR_C [0.691]
- KCF [0.653]
- SCM [0.534]
- DFT [0.520]
- CPF [0.505]
- LOT [0.472]
- Struck [0.469]
- RS-V [0.469]
- SMS [0.457]
Then, the tracking results from (OTB) is given in Figure 5.6. As we can see, the proposed approach is the best tracker compare to all others. Especially in fast motion, motion blur and occlusion sequences. An important observation regarding the target representation is given in Figure 5.7, where the bottle in up row with constant scale is tracked very well; however, the algorithm tries to preserve the object scale despite the changing scale which introduces error in the success error metric. In most cases, the tracker slowly catches up with the size change. In other cases, we observe that background clutter may confuse the tracker to update the scale incorrectly.

Occlusion: Most realistic sequences may contain partial target occlusions. While the trackers based on a holistic target model may have some problem, the proposed target representation models target parts and work under the partial occlusions. Similar to the idea of patch based tracking, proposed approach can track the center of the target which is later used to estimate the target scale illustrated in Figure 5.8, another player partially occludes the tracked player. The proposed representation can estimate the target center with high accuracy which results in high confidence state estimation. In the case when the occlusions happen, the evaluation scores used to update the target representation are lower than none-occlusion cases, which inhibits model update, hence does not damage representation. This property allows the tracker to reacquire the target back once the occlusion is completed.
Figure 5.7. Scale changes can cause inaccuracy in scale estimation. The first row shows a successful scenario where the scale of the tracked target remains constant. The second row shows limited performance due to background clutter masquerading scale change. In both cases, the yellow dot represents estimated target center, the 2nd image shows center voting, the 3rd image shows estimated target scale and position and the 4th image shows target estimate superimposed on the image.

Figure 5.8. Tracking under occlusion that the target is covered by another player. Despite, the voted center is correct, and the resulting confidence map provides adequate information to estimate the target state.
Figure 5.9. Illustration of the removal of unique cells in the representation which are contaminated by background clutter. Each triplet of images shows a consecutive frame from a sequence. In the first image, the background does not cause confusion. The second triplet shows confusing descriptors due to common green color in the background and on target. The last triplet shows that the confusing information is removed from the unique target representation and the target can again be tracked well.

Background clutter: In the case when the background and target have similar descriptors, the tracking task becomes challenging. An implicit advantage of the unique target representation is to suppress cells with confusing descriptors and only focus on unique cells. In such cases, the target state estimation discussed above along with the extracted confidence map help to estimate the target center still at the true position. Once the new location and scale are estimated, for the following frames, the confusing cells no longer affect the tracking. In Figure 5.9, we illustrate this scenario where the descriptors from the green short of the runner are confused with the grass. As can be seen in corresponding confidence maps, the confusing cells in the representation are suppressed resulting in a cleaner target model in the following frames.
5.4. Discussion and Conclusion

In this chapter, we introduce a novel target tracking method that generates a unique target representation by encoding appearance descriptors and their spatial locations in a codebook. In our approach, the descriptor space is automatically divided into multiple cells of nonuniform sizes to uniquely describe the mapped descriptors as target or background. The cells serve as clusters which are used to generate both weights in a voting scheme and confidence map for target state estimation. The target center voting is
modeled as a modified generalized Hough Transform-based on the spatial codebook which encodes relative position of descriptors. During tracking, the method is capable of updating the target representation to eliminate confusing descriptors or enhance new unique descriptors. The experimental results using a simple color descriptor shows comparably superior performance to other state-of-the-art tracking methods.
Chapter 6: Deep Learning based Tracking

In this chapter, we replace the appearance model by a concept model which is learned from large-scale datasets using a deep learning network. The concept model is a combination of high-level semantic information that is learned from myriads of objects with various appearances. In the proposed tracking method, we generate the target's concept by combining the learned object concepts from classification task. We also demonstrate that the last convolutional feature map can be used to generate a heat map to highlight the possible location of the given target in new frames. Finally, in the proposed tracking framework, we utilize the target image, the search image cropped from the new frame and their heat maps as input into a localization network to find the final target position. Compared to the other state-of-the-art trackers, the proposed method shows the comparable and at times better performance in real-time.

6.1 Introduction

Visual tracking is still an open problem in the computer vision community. Single-target tracking is the most fundamental problem in many vision tasks such as surveillance and autonomous navigation. In the past, there have been many successful object trackers that use features and appearance descriptors [2, 3, 4, 6,]. These trackers can be categorized as either generative or discriminative which use appearance-based models to distinguish the target from the background. Researchers have also introduced sophisticated features and
descriptors [124, 125, 126], yet there are still many issues in practical applications. The main limitation of these low-level hand-crafted features is that they only address the texture of the object which may frequently change.

Recently, Deep Neural Networks (DNNs) have demonstrated promising performance in computer vision tasks, and several CNN based trackers have been proposed [50, 117]. However, most of them have only considered the CNNs feature extraction capability and use the traditional methods to do the tracking. The appearance change problem is still haunting the trackers that even use the deeply learned features.

For humans, an object is not just about its appearance at limited view angle, but its concept which may include every appearance about it. Since the CNN has the capability to learn a general semantic representation of objects, we think it can learn some concept as well. Inspired by this idea, we propose a visual tracker which adopts the one-thousand object concepts learned from ImageNet instead of directly modeling the appearance of the novel target. The basic idea is that, with a pre-trained classification network, a novel target can be modeled as a combination of several existing categories. In another word, we calculate and define one more set of weights for the last full connection in the network to define the novel target. This is not an online training and will not change the parameters of the pre-defined network; it uses the learned high-level features to construct a new object concept which is used to define the target. After getting the concept, a heat map, which is generated by fusion of the last convolution feature maps with the concept, is used to highlight the target in new frames. Then, the target image, the search image, and the heat maps are used to find the final target location by the localization network.
The flow diagram can be seen in Figure 6.1. Since we use a pre-trained network, there is no on-line training needed which makes the tracker work at high frequency. The main contributions of this paper include: (1) proposes a method to use object concept from CNNs instead of learning feature appearance in visual tracking; (2) offers an efficient way to define target as combination of pre-learned object categories without on-line training; (3) combines the heat map and target image to form a cascaded tracking network that works at high frequency.

Figure 6.1. The tracking flow of proposition. In the proposed tracking framework, there are two different networks: one is for the target identity concept recognition, and the other is for the target location detection.
6.2 Object Deep Concept

6.2.1 Classification on Unknown Target

The proposed tracking idea is based on the assumption that the CNNs can learn high-level object category concepts. In order to test this assumption, we conducted a number of experiments using the tracking dataset TB50 [123] with the GoogLeNet [127]. GoogLeNet originally designed to perform classification of 1000 object categories in the ImageNet challenge. We note that most of the targets in TB50 are not labeled in the dataset. Our test is to verify if the GoogLeNet can consistently classify TB50 targets that it has not learned before as one or several categories.

Based on the ground truth, we crop targets from all image sequences and classify each cropped image with the GoogLeNet network. We record the top 5 classes for each classification. We record the top 5 categories as the target classification results. Each category’s score is estimated as the ratio of the recorded time and the index of the target image. Two of the results are shown in Figure 6.2. Since the network is not trained with the labels like standing body or the face, it classifies the targets as different objects. In the “singer sequence”, the target classified as space shuttle with the highest score. From this perspective, the network learns the novel object semantics despite scale, illumination and appearance distortions. However, in the image (b), the human face is identified as a band-aid, sunscreen, and Windsor tie with high scores. From the texture point, the face has nothing similar with them. However, all these classified categories are related to skin or face. Hence, when the network finds a band-aid or the similar texture object, it may also find a face.
We also consider the repeatability of this pre-learned semantics. For that purpose, we choose the top 5 classified categories in the previous test as “true classes” that represent a target and for each tested target, if one of the top 5 highest score classes belongs to the selected representative classes, then we mark the classification as correct. The classification accuracy is the ratio of correctly classified against the total number. We have consistently observed that for almost every sequence, the classification accuracy reach 100 percent, which can be seen in the supplemental materials. The results suggest...
that the target has semantic features that spread along the five classes, and during the sequence, the target will have a high response to these semantic features.

6.2.2 The Target Concept

Before we explain how to generate a new target semantic concept, we will first introduce the semantic concept, then define how to generate heat map for specified object category with the concept.

During training, the parameters in the network are adjusted to make sure the output have the same labels. In our approach, we separate the parameters in the network into two. One is used to extract high-level semantic features which are the weights of filters in convolution layers. The other one is the weights of the connector from high-level semantic features to object categories in the full connection layers. The convolution parameters decide how to extract features while the full connection parameters decide how to use these features. So, from this perspective, the object category score is dependent on the weights that decide the connections to the last convolution feature maps in the network. In other words, the object category is defined as a combination of these high-level features, and the connection weights decide this combination. Hence, we define the concept of an object category as a set of weights that connects the category and the last convolution feature map.

Even when the full connection layers lose the spatial information of the high-level features, many classification networks have demonstrated a remarkable object localization ability [128, 129, 130]. In [128], the net’s last convolution layer is replaced by a global average pool (GAP) and the full connection layer is reduced to one to make
the connection between category and high-level features very simple. Following the results of these studies, we combine all the last feature maps by the category weights and generate a class specified heat map (Figure 6.3 shows an example). The heat map can highlight areas that include the high-level features which are trained as components of a given category.

It is impossible to train a network on all possible targets. Considering the conjecture, we made in the last section, unknown target shares many semantic features that were learned from other objects, we use this fact to extract features from CNN and combine them to learn and define new targets. Unlike on-line learning, we don’t change weights in the network. Instead, we select a new combination of the feature maps to generate the concept of a novel target. In order to do this, one can use the idea of sparse coding to generate a new concept with all high-level feature maps, but the process is time-consuming, and what’s more important, the method converges to the same as appearance-based tracking method instead of the target conception. The sparse coding offers a set of weights that make the combination of the feature maps more similar to the appearance of the target. For example, if the target is a cat face, the spare code only generates a concept about the appearance of a cat’s face which is variant even totally different in the following frames. In order to avoid this, we weight the feature maps at the category level. The cat face, which the network does not know before, may be classified as dog or rabbit. On the class level, the concepts about the cat will include dog’s body and limbs, rabbit’s nose and tail, which make the concept more sensible, especially when the target cat shows his arms and body in the next frames. Based on this consideration, we treat the
connections between the feature map and the class category as one set for each category, and we do not change it since it’s already been trained for the category. Let:

\[ W_i^n = \{w_1^n, w_2^n, \ldots, w_m^n\}, \]  

(6.1)

represent the \( n^{th} \) category weights for all \( m \) feature maps in the last convolutional layer.

Then the goal is to present the new object as estimating a set of coefficients that combine \( W^n \) as:

\[ W^{n+1} = \sum_a a_n * W^n. \]  

(6.2)

Here, we use an efficient way to estimate the coefficients while strengthening the same feature parts and weakening the difference. We first use the network to get classification results \( \text{Score} \) and weights \( W_i^n \) that show the mapping of all feature maps for each category. Then we generate a heat map for each high score category as:

\[ H^n = \sum_i W_i^n * M_i, \]  

(6.3)

Since we know the ground truth of the target during the tracking, we can evaluate how good each heat map is for the target by measuring the values inside the target area against the outside as:

\[ a_n = (V_{\text{inside}}(H^n) - V_{\text{outside}}(H^n))/V(H^n), \]  

(6.4)

where \( a_n \) is the final combination coefficient of each category \( n \) and \( V(H^n) \) is the sum of all values in the heat map (see Figure 6.3 for target concept generation). Using this approach, we generate a set of connection weights to represent the target concept from learned object categories without online training.
Figure 6.3. The new target concept estimation. Given a target, we can generate a number of heat maps for top $n$ classes that the target may belong to like the red classes. Then based on the ground truth on the heat maps, the coefficients for the combination of these classes’ weights can be calculated to generate the new concept for the target.

In order to test the performance of the new target concept, we conducted another set of experiments with the same TB50 dataset. From the initial target image, we generate the new concept and find the other four top classified categories as a comparison set. Then we use the new concept to generate the 1001st category. There are tens to hundreds target images in each sequence. In the experiment, we test how many of them are belong to this 1001st category and the other four top categories. In each classification, we assume the target belong to both the top 5 classified categories. The results can be seen in Figure 6.4. As can be observed, except six in fifty-one tracking sequences, all the sequences have scores close to 100 percent, and it is far more accurate than any other class in the initial frame. That suggests that even we only get the tracking target concept from the initial
frame, the network can recognize the target in the following frames as this newly defined class.

![Figure 6.4](image-url)

Figure 6.4. The classification results of the targets in TB50. The vertical axis is the percentage that the target classified as same class in the same image sequence.

### 6.3.3 Heat Map for Target Localization Prediction

Since heat map can highlight the interesting parts of the input image for a specified class, it is helpful for tracking. We can generate the concept of the target in the initial frame, and for following frames, we extract their high-level feature maps, and based on the target concept, generate heat maps to estimate the location of the target. To test the ability of the heat map in prediction, we evaluated if the heat maps generated from the concept helps to localize the target in the new frame.
In this evaluation, we use the dataset from ALOV300+ [131] which has 314 image sequences. In this dataset, approximately every 5th frame of each sequence has a label to locate the target. During the test, we randomly select two frames from a random sequence which has more than one label. In the first image, the target is cropped as target template with some background texture. Using the sample generation idea from [55], we randomly shift and scale the target in the second image as search image to simulate the motion of the target and camera simultaneously. In tracking, we cropped a search image in the new frame based on the target’s previous location instead to track the target in the whole image. Hence, we set the image window size both as two times as the target rectangle. Also, the network need some contextual texture of the target to help the classification and generate the concept. We don’t input the whole image into the network. We test some sizes for the input image and finally select 1:6 time of the target rectangle as best. Also, we choose $n = 100$ as the number of the top classes that are chosen to generate the new concept in (2). The parameter selection test can be seen in the supplemental materials. In Figure 6.5, we show some of the prediction results. As can be seen, the heat map predicts the target location in new frames. Even in the case of appearance blurring and rotation (left top), illumination changes (right top), translation (left bottom), scale change (right bottom), the heat map still can highlight the target area well. The left bottom example also shows us that, the learned concept is category based, it will highlight all the object that belong to the category. So, if there are similar objects in the image, the heat map will highlight all of them. To quantitatively measure the highlighting ability of the heat map, we use equation 6.4 to score the heat map for both the target image and the
search image. Since the image size is twice the target, an even distribution score will be 
−0.5. Hence, for estimate the initial and predict a score, we scale them from 0 to 1 by:

\[ S_{\text{scale}} = (S + 0.5) \times \frac{2}{3}. \] (6.5)

In order to report results, we randomly choose 500 image pairs in each image category in 
the dataset and average the scores as shown in Figure 6.6. In the figure, the blue bars are 
the initial scores of the new concept of the target image, and the orange bars are predicted 
scores on the search image. As we see, the predict scores are proportional to initial scores 
in all sequence categories. That implicates that we can use the initial score to estimate the 
prediction ability of the heat map.

Figure 6.5. The heat map for prediction. In each sub four images, the left top is the target 
image with ground truth to generate the heat map and calculate the target concept also the
initial score of the concept (top left). The lower left image is the search image, and the right bottom image is the heat map generated by the concept of the target image. The ground true in search image is used to calculate the predicted score.

If the initial score is very low, we don’t use the heat map to predict the target.

Additionally, for different scenarios, the confusion has a low score, when there are background clutter, transparency, and moving camera. In these situations, the targets are either affected by the background or the appearance is not stable which makes it hard to distinguish the targets. Even in such challenging situations, the prediction score is still higher than 0.35 which suggest that algorithm can mark the target region successfully.

![Figure 6.6](image.png)

Figure 6.6. The predict and initial score in different situations.

6.4 The Localization Network
The use of concept alone is not adequate to locate the target, the detailed appearance about the concept is also important. Hence, we use both the target image and search image with their corresponding heat maps together as two sets of $height \times width \times 4(R,G,B,HeatMap)$ data blocks to feed to a Siamese localization network to get the tracking results. The architecture of the localization network is similar to CaffeNet[132] which can be seen in Figure 6.7.

![Figure 6.7. The architecture of localization network.](image)

To train the localization network, we keep generating samples from the ALOV300+ and ImageNet 2015 [133]. For the ALOV300+ data, the overlap sequences that are also in our experiment TB50 dataset are removed, and the samples are generated as described in section 6.2, but, we don’t only select continuous frames. Since we want the localization
network to find target location by both the texture and concept information, the search image may not always be similar to the target image which is important in training. For the static images in ImageNet, the search images are randomly cropped with scale, and translation changes respect to the target location. Considering that the location changes are smooth in most cases, the cropping follows a Laplace distribution given by:

\[ f(x|u, b) = \frac{1}{2b} \exp\left(-\frac{|x-u|}{b}\right), \] (6.6)

where we set \( u = 0 \) for both scale and translation changes, \( b_s = 1/5, b_t = 1/15 \) for scale and translation separately. Also, we enforce the scale change to be less than \( \pm 0.4 \) and the center of the translated target is still in the search image similar to the work in [55]. The ImageNet dataset is mainly used to teach localization network to find object boundary and the smooth motion as complementary data in the limited ALOV300+ dataset.

6.5 Deep Concept Tracking

For single object tracking, the target is selected from the first frame with an initial rectangle. First, we define the target and get its concept as described above. For tracking, the target image is cropped as twice as the initial rectangle. When the new image comes, a search image is cropped based on the previous target location with twice the rectangle size. After that, the heat maps of target and search images are feed to the localization network to find the final target position.

Since the initial target image is the only example appearance of the target and the localization network performance is better than if we use the appearance of tracked target in the previous frame, we fix the initial target as a target image in the localization
network. By fixing the initial target, the tracker can efficiently avoid the drift problem. But, when a similar object comes nearby, the concept will highlight those regions and confuse the tracker. So, during the tracking, the target concept needs to be updated:

\[
Concept_n = a \ast Concept_{old} + (1 - a)Concept_{current},
\]

(6.7)

where the a is the learning rate, and \(Concept_{current}\) is the target concept calculated by the current target location. Performing this update will suppress the concept which contain the confusing regions. Hence, in the heat map, the confusion region will not be highlighted. Since the prediction score is proportional to the initial score, we can use equation 6.4 to estimate the tracking quality. If the tracking quality score is lower than a set threshold \(\Lambda\), then we assume there are too many distractors, we keep the target location unchanged to further avoiding drifting.

6.6 Experiment

6.6.1 Experimental Setup

Beside the parameters discussed above, we set the learning rate \(a = 0.5\) and prediction quality check threshold \(\Lambda = 0.3\) for all experiments. We use Caffe as the deep learning tool [132]. We implement the tracking algorithm in Matlab and run it with the NVIDIA graphic card GeForce GTX 950 at average 32 FPS.

We test the performance of the proposed approach using a large dataset referred as the online tracker benchmark (TB50) [123]. The performance scores are provided from the one-pass evaluation (OPE) toolkit. There are two measurements in the evaluation: Precision plot and success plot. The precision plot is used to measure the distance between tracking results and ground truth. The success plot is used to evaluate the
overlap score of them. For more details, please refer to [123]. We also compared the proposed tracker on the benchmark with other 35 popular trackers including 29 trackers in [123] and KCF [41], MEEM [38], DLSSVM [134], and some more recent deep learning based trackers CNNSVM [53], SiamFC-5 [54] and FCNT [50].
Figure 6.8. Average success plot and precision plot for the OPE on TB50.

Continued
Figure 6.8 continued
Figure 6.8 continued
Figure 6.8 continued

Success plots of OPE - background clutter (21)

Precision plots of OPE - background clutter (21)

Continued
Figure 6.8 continued

Success plots of OPE - deformation (19)

Precision plots of OPE - deformation (19)
6.6.2 Experimental Results

The tracking results from TB50 are given in Figure 6.8. More results for different attributes are given in the supplemental materials. As we can see, the proposed approach is one of the best trackers compared to all others. Especially in fast motion, motion blur, in and out plane-rotation and illumination change sequences. We believe the excellent performance is mainly because the tracking target concept from the deep network has very high semantic information which is invariant to rotation, illumination, and translation and scale changes. Use of the concept generated a heat map to highlight target area is robust to this appearance changes as shown in both Figure.6.5 and Figure6.9. In these figures, when the target undergoes serious appearance changes, the concept heat map can still find it correctly. As one can expect, the occlusion and clutter sequences reduce the performance. This may be due to the fact that when the sequence contains seriously confusing situations, both the appearance and concept clue fail to work. We will discuss more in the coming paragraphs.

Compared to other trackers, our method shows excellent performance in overlap score but a slightly lower performance at precision. This can be attributed to the fact that the localization network can find the target boundary and make the output rectangle fit the target very well. But in complicated scenarios, the tracker may lose the target completely which reduces the precision score. Compared other all trackers, ours works in real-time (32 FPS). The DLSSVM runs at 5.4 FPS, FCNT runs at 3 FPS, CNNSVM doesn’t show their time permanence in their paper but should be far away from real-time. Only the
SiamFC compares to ours at 58 FPS, but with better hardware (GTX Titan X) that is more powerful than ours (GTX 950).

Occlusion: Most realistic sequences may contain target occlusions. While the trackers based on a holistic target model may have some problems, the proposed target concept still works under the partial occlusions. Even when there is only a small part of the target is visible, the concept still highlights visible parts. Based on the texture information, the target can be found after recovering from the occlusion. As we can see in Figure 6.10, the tracked player is partially occluded by another player. The concept can still highlight the target part in the heat map. And based on the texture information, the target player can be located correctly.
Figure 6.9. Illustration of good tracking with (a) significant illumination change, (b) texture and edge change, (c) deformation and out plane rotation, (d) fast motion and motion blur.
Background clutter: In the case when the background and target have similar descriptors, the tracking task becomes challenging. If we only use the unchanged concept to detect the target, it is hard to remove the distractors, as shown in the left bottom image in Figure 6.5. However, during the tracking, the concept is updated which can suppress the similar concept and increase the different ones by our concept generation process. When a similar object is observed, since it is not in the target rectangle, their similar concepts will be treat as an outlier and is suppressed in the next frame. Figure 6.11 illustrates an example. In the first row, the concept about the helmet is highlighted for both players. However, after the concept updating, some similar concepts were suppressed in the heat map to help identify the true target.
The Lost Target: Our tracker may have problems in some complicated situations. The tracking is based on the concept and the target appearance information. When they both fail, they tracker may fail. In Figure 6.12, the concept captured from the upper image are mainly about the shape and the edges of the face which are completely lost in the lower image. Also, the face texture disappeared in the search image and left no clue to find the target. Besides that, when a new similar object appears, the concept will be weaker to highlight the target area. At the same time, if the target’s appearance undergoes changes, both of the two tracking cues will be lost. Since proposed tracker does not update in such situations, the target can be retrieved after it reappears.
In this chapter, we introduce a new target tracking method that uses the concepts learned from deep learning network to represent a novel target. The target concept is generated by combining high-level features from the deep network pre-trained on unrelated objects. These high-level features are invariant to scale, rotation and translation changes, even a serious deformation. Also, the concept and high-level features can be used to generate a heat map which highlights potential target area in the search map. In the tracking phase, the initial target image is used as constant texture information in a Siamese localization network. Regardless of only using the constant initialization target example, our method shows good performance in the case when the object appearance undergoes significant changes.

6.7 Conclusion

Figure 6.12. Illustration of concept and appearance information both lost the target.
Chapter 7: Space Tracking for Surgical Navigation and Education in AR system

Recently, augmented reality (AR) technique has become more and more mature in real practice with the development of powerful hardware. Besides in game and industrial field which the AR can build an amazing mixed reality world between the virtual and actual objects, these attractive features are of high potential in the medical field as well. For the low-level applications, a wearable AR device can be employed as a medical training tool or a surgical navigation system. For the high-level applications, AR technique can significantly boost the development of robot-assisted surgery. By combining the tracking and AR technology, this work in this chapter developed a mixed reality system integrating a HoloLens and a three-dimensional (3D) point tracking module, which can be applied as a medical training and navigation system on tumor resection surgery. In the system, a stereo camera is used to track the 3D location of a scalpel and transfer the coordinate to the HoloLens via a wireless network. On the HoloLens side, its coordinate system and virtual model may (for navigation) or not (for training) register to the real scenario. For the training, the virtual model can be displayed at any place and scale. And the whole operation can be displayed at all angels. For the navigation, the virtual model is overlapped with the actual scenario. The virtual scalpel moves along the true object to guide the operator’s motion. Finally, the experiment demonstrates the practical usefulness of this AR system as well as the precise localization ability of the tracking system.
7.1 Introduction

AR has become more and more powerful and practical in the real lives [138, 139]. With the AR system, the user can understand the environment with more information that may come from the device’s sensors or the digital knowledge from the internet. AR has enormous potential uses in many applications even in the industry, like in the field of construction, decoration, product design, especially in medical operation. In [140], the Google Glass is used to display the tumor information over the operation space which greatly helps the surgical operation. Similar, an AR system is introduced in [141] to localize small tumors accurately.

The work here also demonstrates the potential applications of AR technique in a medical procedure. For a training system, display of a full range of surgical procedure is a perfect way to instruct medical students. From this angle, the AR system has a significant advantage in playing holographic information in 3D space. At another side, remote teleguide surgery is in huge demand. After the operation scenario is digitalized, experts at worldwide can help to guide the surgery or directly operate on the remote side of a robot surgery assistant. So, from these two aspects, the work uses a HoloLens and a stereo camera to generate a medical augmented reality system for a tumor resection surgery.
Figure 7.1 The medical navigation and education augmented reality system.

7.2 The Augmented Reality and HoloLens

From the Wikipedia, augmented reality (AR) is a live direct or indirect view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data.

Comparing to general display, AR provides an extra information for users which is an overlap on the current environment. Right now, AR can provide a natural way for people
to interact with virtual objects, just like the interacting with the environment in daily lives. The AR techniques provide a novel approach to visualize virtual objects in 3D space and display holographic 3D models, which offering user a more natural and immersive way to interact with them.

At another level, the virtual objects may interact with the real world. That means the extra information is not just a fixed overlap, but a synthetic content upon the reality and that is referred to Mixed Reality (MR). In some literature, the MR and AR have different meanings, but in the dissertation here, we prefer to define MR as a special situation of AR. So, in the following sections, we use AR to refer both AR/MR. The main difference between AR and virtual reality (VR) is whether rely on the real environment. VR is independent on the environment while AR offers a response to the real world. Beside regular color camera, AR device usually has much more sensors than VR, like the depth camera, infrared camera, GPS, Lidar and so on.

HoloLens is the most powerful AR product at current. It uses an optical see-through head-mounted display, with a resolution of 1268*720 per eye. The device contains four microphones, one ambient light sensor, one two-million-pixel photo/video camera, one depth camera and four environmental understanding cameras as well as an IMU as its primary tracking sensor. Using these sensors, HoloLens can recognize a voice, surround 3D environment and human motions like gaze and gesture tracking and analysis.
The most impressive feature of HoloLens is its spatial mapping ability. The spatial mapping can generate the 3D surface of around environment. To analyze the scene and understand it is critical for HoloLens to display holographic objects in the space. And only from a precise location, the holographic objects can be displayed correctly in the space. By using the spatial mapping result, IMU and environmental understanding cameras, HoloLens can rebuild the entire environment and find the locating of itself in this environment.

7.3 System Component

The whole system consists of operation table, stereo cameras, the scalpel with lights and the HoloLens as Figure 7.1 shows above. Each part is corresponding to different
functions. The whole system can also be divided as registration, tracking, display and all these things can be included in two branches: the stereo tracking and the HoloLens presenting. The working flow of the AR system is described in Figure 7.3.

![Flowchart](image)

Figure 7.3. The working flow of the two branches of the AR system.

### 7.3.1 Registration

The registration means register different coordinate systems into a universal one. In the operation table, there are four markers used as fiducials or ground control points to set up the coordinate system as the universal one. The center of the top left marker is defined as the original point and the three perpendicular axes are X, Y, and Z which are illustrated in the Figure 7.4. In the surgical navigation and education system, all the other coordinate need to be registered with it.
Since the tracking module is based on the camera coordinate, the stereo cameras’ locations need to be calibrated with the four fiducials on the operation table as Figure 7.4 shows above. After the calibration, the scalpel tip point is then tracked in this universal coordinate system. Another registration for the coordinate system is the object coordinate in HoloLens, like the virtual arm, which also needs to be registered to the operation table. Only the camera and HoloLens are both aligned to the operation table, the tracked scalpel location can be transformed correctly to the virtual scenarios.

For camera calibration, this work uses algorithm and equations from section 3.4 and the OpenCV library [135]. To ensure integrity, the camera calibration processing is described
as follow. Firstly, 32 checkerboard images at different view angles are captured for each camera for the intrinsic parameters using the OpenCV camera calibration functions. Then, we manually find the center of the four fiducials in camera images which corresponding to their ground truth locations. Then, the intrinsic and extrinsic parameters are calculated and refined by using bundle adjustment as described in section 3.4.1. and 3.4.2.

To simulate a 3D object in the physical world, HoloLens uses a right-handed Cartesian coordinate system with three perpendicular axes: X, Y, and Z, and the Y axis aligned to gravity pointing up. The HoloLens’ coordinate system is decided by its initial status which decided by the camera. The camera center will be the original point. However, HoloLens’ coordinate system is not fixed after its initialization. HoloLens tries to make its coordinate system as perfect as possible during the whole running time. So, the coordinate system keeps updating while the HoloLens is mapping the environment. Since the operation table is a relatively simple place for mapping, the initial coordinate system for the operation space can be assumed has high accuracy and will not change during the whole surgery. To register the HoloLens to operation board, the fiducials are first found in HoloLens’ color image. Then based on the translation relationship between the color image coordinate and depth image coordinate, fiducials’ 3D locations in HoloLens can be retrieved to establish the coordinate translation between HoloLens and the operation table.
7.3.2 Tracking Module

Two cameras are installed on the top of the operation board to form a stereo camera as figure 7.1 shows. As described in section 3.4, a space point’s 3d location can be calculated by finding its projections in the stereo camera. However, directly track the scalpel can be complicated and time-consuming. To simplify the tracking problem, a small holder is added on the top of the scalpel to help locate the tip of it. The holder is generated by a 3D printer with high dimension precision. There are three forks, and each holds a color light (blue, green, and red). During tracking, the stereo cameras first detect and track these lights in 2D images and then use equations from section 3.4 to estimate the 3D location of the points further to retrieve the tip of the scalpel. The tracking system is running around 20 fps which can be improved further.

Figure 7.5. The scalpel with LED lights on the top.
Since the tracker is tracking active light, camera’s exposure time is set to a small value to avoid distractors and increase the image capture rate. During the tracking, the images are first segmented by a threshold to find the highlight parts as potential light points.

\[ I_{th} = \begin{cases} I_{x,y}, & I_{x,y} \geq \theta_A \\ 0, & I_{x,y} < \theta_A \end{cases} \]  \hspace{1cm} (7.1)

where \( I_{th} \) is the image after segmentation with threshold value \( \theta_A \). After a Gaussian filter, contours are used to find the blocks that remain in the threshold image. Furthermore, these blocks are filtered by size which can help to remove the distractor and noises caused by reflect area on the operation table. After these steps, each light block is treated as the candidate. To identify the color of each block, we first calculate the block’s average color for each channel \((C_r, C_g, C_b)\), then use the following equation to estimate the main color of the block:

\[ C_{main} = \text{Max}(S_r, S_g, S_b), \]  \hspace{1cm} (7.2)

where \( S_r = C_r^2 - C_g * C_b \), \( S_g = C_g^2 - C_r * C_b \), \( S_b = C_b^2 - C_g * C_r \). In order to find corresponding light pairs, epipolar constraint introduced in section 3.4 is used to find the same color light at the other image that used to form the stereo matching block.

For a precise localization in space, the matching points in the image should have sub-pixel accuracy. Due to the dynamic noise of cameras, the light block may have some changes during the tracking. For the purpose of robustly locate the light, we use the weighted light distribution to estimate the center of the light:

\[ L_i(x, y) = \frac{\sum_{j \in L_i} w_j \cdot (x_j, y_j)}{\sum_{j \in L_i} w_j}, \]  \hspace{1cm} (7.3)
where $L_i(x, y)$ is the center of light block $i$, $w_j$ is gray value of the pixel at $(x_j, y_j)$. With each light’s sub-pixel image location, we can use the 3D location retrieve algorithm as described in section 3.4 to estimate their space locations:

In the best situation, there should be exactly three pairs of matching points. But, due to some errors, there may more or less than three pairs. If the matching pairs are less than three, then the tracking is failed. For this situation of more than three, post-processing is considered to remove the wrong pairs. The post-processing first considers the possibility of the 3D location of the light, if it is too far away from the operation board, then it should be removed. Also, the three lights should be close to each other, if the 3D point is far away from the average location, then it should be removed. After this post-processing, if we still have more than three matching points, the tracking is assumed as failed.

![Figure 7.6. The localization of LED lights.](image)
After the system calculates each light’s location, it can get the perpendicular line of the lights plan and get the scalpel tip’s location just add a distance from the center of the three lights along the perpendicular line. The location of scalpel’s tip is calculated by the three LED lights’ locations as:

\[
L_{\text{center}} = \frac{(L_r + L_g + L_b)}{3},
\]

\( T = (L_r - L_g) \cdot (L_r - L_b), (\cdot) \text{ is cross operation}, \)

\( T_{\text{Normal}} = T / \text{norm}(T), \)

\( \text{Tip} = T_{\text{Normal}} \times D_{\text{scalpel}} + L_{\text{center}}, \)

where \( L_r, L_g, L_b \) are the locations of red, green and blue light, \( L_{\text{center}} \) is the center of these lights. \( T \) and \( T_{\text{Normal}} \) are the normal vector and normalized normal vector separately, \( D_{\text{scalpel}} \) is the distance from the light center to tip of the scalpel as the value \( b \) in Figure 7.5.

From the above equations, we can know that the error from the lights’ location will be magnified at the tip which is multiplied by the distance \( D_{\text{scalpel}} \). Hence, to relieve this problem, the scalpel should be short, and the branch of lights should be longer. Besides this, a smooth method can also make the tip stable in practice. To smooth the trace, the tip’s location is calculated as the average of the \( n = 10 \) previous locations plus the current one. This can greatly smooth the trace of the scalpel tip. However, the smooth processing will make the motion has a delay which shows in Figure 7.7.
Figure 7.7. The smooth of tip’s trace and the delay of the position. At the upper row, the solid circle is the measured location; the dashed circle is the smoothed location for the solid circle. At the lower row, the new location is calculated from the previous smoothed locations and the new measured location.

7.3.3 Virtual model and scenario displaying

Figure 7.8. The virtual arm model.
To simplify the work, we generate the 3D model of the scenario ahead. This virtual 3D model including the operation table, the experiment object (here is an arm), and the scalpel. The arm and the operation table are fixed to each other, during the complete processing, their relative location is keeping unchanged. That’s because all the procedure is under the operation table’s coordinate system, so the experiment object’s location cannot be changed on the table.

Before experiments, this virtual 3D model is loaded to the HoloLens. For the education purpose, the virtual model does not need to be registered to the real operation table. And it can be displayed at anywhere for the tutorial. Then, when the 3D tracking module accurately acquires space coordinate and orientation of the scalpel, the information is transmitting to the HoloLens via a private wireless network. At the end of HoloLens, the virtual scalpel is moved dynamically in its 3D space which along with the 3D model of the arm. The motions just followed the real scalpel which captured by the cameras. Since the HoloLens shows the holographic information of all the 3D models, this true-to-life scenario is best for the education.

On the other side, for the surgical navigation purpose, the virtual scalpel must have attached to the true experiment object. So, during the experiment, the coordinate of the 3D model must be registered to the true operation table as section 7.3.1 mentioned. Then, the movement of the scalpel can be seen on the real scenarios for the remote teleguide surgery.
7.4 Experiment

To demonstrate the ability of the system in training and teleguide, we carried out three experiments. At the first one, we want to see if the virtual scenario that HoloLens displayed is the same as the true scenario. Then, we attached the virtual scenario to the real one and to see if the pre-defined motion of the scalpels can move correctly along the true experiment object. For the last one, the localization capability of the stereo camera is tested.

7.4.1 The Medical Training

In one of our scenarios for this application, a surgeon (the trainer) is doing a tissue resection demonstration with a scalpel. The 3D tracking module accurately acquires the scalpel location, then transmits it to HoloLens. Students with HoloLens can observe the whole processing with a virtual scalpel moving along a 3D model of the real subject in all views.

In the experiment, the camera's intrinsic parameters are first calibrated by a 7x12 checkerboard with the grid size 2.89cm. The operation table’s coordinate is setting as Figure 7.4. The centers of four black and white markers are treated as ground control points for the cameras extrinsic parameter calibration.

To check the performance of the representation of the 3D scenario, we moved the virtual model to the front of the true subject and displayed their movements simultaneously. And in the processing, the trainer is moving the scalpel along the side of the hole on the rubber arm. A video recording can be seen at https://youtu.be/zw0v6JS7bbU and Figure 7.9 shows
some locations of the virtual and true scalpel. As we can see, the virtual scalpel is moving along the virtual arm exactly like the true one moved along the side of the hole.

Figure 7.9. The demo of display virtual operation scenario with HoloLens.

7.4.2 The Surgical Navigation

If we want some untrained person to follow a procedure to accomplish a surgical, the navigation on the real subject is defiantly better than just a video tutorial. With the same setup, the tracing path of the scalpel tip is displayed in the mixed reality world, which can be viewed in stereo in HoloLens. After registration of the virtual object with the real one, the virtual scalpel is aligned and fixed with the real subject, so that the trainee can follow the tracing path and move his/her head freely to get a better understanding of the procedure. More information can be added as hints and notes virtually during the operation.

The tracing path of the scalpel can be pre-recorded for the training purpose or it can be lively received by an expert from a remote side. In the experiment, we pre-recorded the
trace that shows the area which should be removed in the simulate tissues. With the HoloLens, we followed the trace on the real rubber arm to draw a line as the intersection part. A video recording can be seen at https://youtu.be/uRBbaSe80D0 and the Figure 7.10 shows the virtual scalpel drawing the trace and the operator flowering the trace. Finlay, we compare the two areas to see how likely they are.

Figure 7.10. The demo of operation navigation with HoloLens. Image (a) and (b) display the navigation of the virtual scalpel. Image (c) shows the scalpel trace (black) drawing by a human which following the virtual scalpel. The red line is the ground truth that used to record the virtual scalpel’s motion.
As we can see in Figure 7.10, the compare results show us the guiding track works very well to help us remove the un-see tissues. However, we also noticed that in some place, the trace and the real intersection line is not overlapped well. That may be caused by the error from camera tracking and the registration. More quantity measurement about the localization precision will be discussed below.

7.4.3 Localization Precision.

To precisely measure the space localization error. We put the scalped on a straight track to simulate the motion along a line, like the Figure 7.11 shows below. Then we move the track over the operation border at different rotation to measure the precision of the light localization by the stereo cameras.

Figure 7.11. The setup for the experiment of location measurement.
First, we rotate the plant base to simulate an out of horizontal plan rotation, and the Figure 7.12 shows the rotation of the track. At each rotation angle, the track is fixed first, then, the scalpel is moving along the track, and each light is localized by the camera at the 20cm interval, total 14 positions. Then, we move the track over the board at several different positions to cover a larger part of the space. We also repeat the whole process at x and y-direction separately just as the Figure 7.12 shows bellow.

**Figure 7.12.** The track’s locations for the experiment of localization measurement. The blue circle is the positions that measured, the red circle is the center of the points in the same track, and the lines are the fitting results from these positions.

There are no ground control points in the operation space, however, for each track’s movement, the calculated space points should be on the same line. So, we first fit a line
based on these points, then to see their average and standard deviations of each point’s error respect to the fitting line. The results are showing at table 7.1.

Table 7.1. The average absolute errors of lights to the fitting lines. A is the angle of the scalpel, P.A is the different positions’ average errors, L.A. is the different lights’ average errors. The values are in cm.

<table>
<thead>
<tr>
<th>A</th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>L.A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R</td>
<td>0.082</td>
<td>0.072</td>
<td>0.052</td>
<td>0.083</td>
<td>0.067</td>
<td>0.066</td>
<td>0.050</td>
<td>0.048</td>
<td>0.078</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>0.065</td>
<td>0.081</td>
<td>0.066</td>
<td>0.079</td>
<td>0.064</td>
<td>0.053</td>
<td>0.075</td>
<td>0.082</td>
<td>0.054</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.068</td>
<td>0.080</td>
<td>0.081</td>
<td>0.100</td>
<td>0.076</td>
<td>0.094</td>
<td>0.087</td>
<td>0.073</td>
<td>0.095</td>
</tr>
<tr>
<td>2</td>
<td>R</td>
<td>0.069</td>
<td>0.051</td>
<td>0.060</td>
<td>0.063</td>
<td>0.077</td>
<td>0.055</td>
<td>0.058</td>
<td>0.056</td>
<td>0.055</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>0.069</td>
<td>0.058</td>
<td>0.053</td>
<td>0.062</td>
<td>0.064</td>
<td>0.059</td>
<td>0.054</td>
<td>0.055</td>
<td>0.053</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.049</td>
<td>0.057</td>
<td>0.069</td>
<td>0.063</td>
<td>0.066</td>
<td>0.054</td>
<td>0.064</td>
<td>0.064</td>
<td>0.061</td>
</tr>
<tr>
<td>P. A</td>
<td>0.067</td>
<td>0.067</td>
<td>0.064</td>
<td>0.075</td>
<td>0.069</td>
<td>0.064</td>
<td>0.065</td>
<td>0.063</td>
<td>0.066</td>
<td>0.066</td>
</tr>
</tbody>
</table>

From Table 7.1, we can see that all the localization measures have high correctness and precision. The average of absolute errors in all the positions is close to 0.07 mm, which is relatively small for the surgical demonstration and navigation. For different lights, as we can see, the different color doesn’t show too much difference. Also, for the nine positions, the errors are relatively evenly distributed. That means the color and position of the lights only have limited affection on the precision of light localization.

However, the results from the two different angles have obvious differences. The angle which is less facing the cameras has little larger errors than the one that more facing to the cameras. That’s because the view angle significantly effects the segmentation of the lights. From a sharp angle, the noise may change the segmentation center greatly, but, when close to the vertical angle, the affection becomes smaller and smaller. So, in practice, it’s better to make the LED lights facing the cameras.
Table 7.2. The standard deviation of errors that lights locations to the fitting lines. A is the angle of the scalpel, P.A is the different positions’ average errors, L.A. is the different lights’ average errors. All the values are in cm.

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
<th>P8</th>
<th>P9</th>
<th>L.A.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R</td>
<td>0.055</td>
<td>0.042</td>
<td>0.021</td>
<td>0.049</td>
<td>0.044</td>
<td>0.036</td>
<td>0.021</td>
<td>0.038</td>
<td>0.042</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>0.035</td>
<td>0.039</td>
<td>0.038</td>
<td>0.037</td>
<td>0.036</td>
<td>0.028</td>
<td>0.031</td>
<td>0.051</td>
<td>0.026</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.030</td>
<td>0.045</td>
<td>0.039</td>
<td>0.051</td>
<td>0.050</td>
<td>0.047</td>
<td>0.040</td>
<td>0.052</td>
<td>0.058</td>
</tr>
<tr>
<td>2</td>
<td>R</td>
<td>0.032</td>
<td>0.025</td>
<td>0.021</td>
<td>0.026</td>
<td>0.035</td>
<td>0.033</td>
<td>0.037</td>
<td>0.042</td>
<td>0.026</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>0.050</td>
<td>0.032</td>
<td>0.044</td>
<td>0.038</td>
<td>0.030</td>
<td>0.036</td>
<td>0.015</td>
<td>0.028</td>
<td>0.027</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>0.024</td>
<td>0.025</td>
<td>0.024</td>
<td>0.036</td>
<td>0.045</td>
<td>0.035</td>
<td>0.032</td>
<td>0.040</td>
<td>0.035</td>
</tr>
<tr>
<td></td>
<td>P.A.</td>
<td>0.038</td>
<td>0.034</td>
<td>0.031</td>
<td>0.040</td>
<td>0.040</td>
<td>0.036</td>
<td>0.030</td>
<td>0.041</td>
<td>0.036</td>
</tr>
</tbody>
</table>

From Table 7.2, the standard deviation values are not small as expected. The larger standard deviation means the data is not close to the fitting lines and the error distribution is relatively random. From this, it may infer that the random camera noise is the main factor that causes the localization error. Since the explore time is short, it is easy to have noise at pixel level which makes the estimation of the center of lights unstable all the way. That’s also the reason of using historical location to smooth the tracking location as post-process. On another side, even after camera calibration, there are still some errors from the lenses distortion and inaccuracy of the camera’s extrinsic parameters.

7.5 Conclusion

In this chapter, an AR based surgical navigation and education system is proposed. The goal of the system is to calculate the location of the scalpel in real time and display it in the HoloLens. Two cameras are formed a stereo camera to calculate the location of the LED lights which amount on the scalpel to help identify its position. Experiments about the localization errors show the tracking model has a high accuracy. Also, the displaying of the virtual scenarios in the real world is impressive and helpful for the surgical
navigation and education. Right now, the virtual model is fixed which can’t update with the real world. So, the next work will be dynamic generating the virtual models according to the actual objects.
Chapter 8: Conclusion

This chapter summarizes the main research and contributions of the dissertation and discusses limitations and suggestions for future work.

8.1 Summary and Contributions

Visual tracking is one of the most important tasks in computer vision applications. In this dissertation, three tracking methods are proposed trying to deal with the tracking problems for different targets and various scenarios.

First, for a simple tracking scenario, an efficient tracking method with distinctive color and silhouette is proposed. It uses dynamic color models to represent the target. Also, with the color segmentation, the target’s silhouette is detected to improve the appearance model. Without complicated and time-consuming features, the efficient tracking method is simple and easy, and the performance is excellent with high execution rate for various practical scenarios.

Next, based on the similar tracking idea, a unique feature vote tracking algorithm is further developed to track the rigid object in occlusion and cluster background. This work divides the feature space into many small sub-spaces as storage cells for feature descriptions. These cells record the feature’s uniqueness and spatial information to help find the target again. This method also used a novel tracking method with feature’s confidence and spatial information to estimate target area.
Next, deep learning and the convolutional neural network has been proposed to track the object by the learned high-level semantic features. Object are representing as a combination of these high-level features and generate a hot map of the new coming frame that shows the possible distribution of the target.

Beside 2D tracking, two cameras are formed a stereo camera to track the 3D space location of a scalpel. And the localization experiment shows the accuracy of the 3D space tracking. By combining the HoloLens, the tracking model shows the enormous potential in augmented reality like the surgical education and navigation.

8.2 Discussion and Future Work

From low-level to high-level feature, this dissertation offers three different methods to track an object in different scenarios. However, there are still many limitations of these methods. For low-level features, the changes of the appearance are always un-expectable. No matter how elaborates the feature descriptors are, it is almost impossible to cover all the appearance changes in the practice. On the other hand, the high-level feature is too abstract for an accurate positioning. Also, the online training efficient of the unknown object needs to be further improved. Despite these shortcomings, the visual tracking in specified applications can achieve relative high precision location information with the help of special tracking designs like using markers and LED lights.

In the future, visual tracking will not be just tracking problem, but also detection and segmentation. Object detection can directly give the location of the target. Generation object detection methods like You only look once (YOLO)[145], faster RCNN[146] and single shot multibox detector (SSD) [147], will be one of the fundamental parts of the
future visual trackers. Combining detection and tracking will make the computer vision tasks meaningful and more intelligent. Figure 8.1 shows the detection methods used in SSD.

Figure 8.1. The detection framework of single shot multibox detector [147].

For the future tracking problems, besides the target’s general location, the precise status of the target is also required. To get that information, the segmentation technique must be included to find the exact target area. Recently, a data set named Densely Annotated Video Segmentation (DAVIS) has been introduced by [148] [149]. The data set consists of fifty high quality, full HD video sequences, spanning multiple occurrences of common video object segmentation challenges such as occlusions, motion-blur and appearance changes [149]. In the newest version (2017), the new dataset, the sequences have more than one annotated object and there are total 10459 annotated frames and 376 objects [148]. Figure 8.2 shows two example frames with the corresponding overlaid object annotations. As we can see, the annotation is at pixel level and the task is so-called semi-
supervised video object segmentation. By given a video sequence and the mask of the objects in the first frame, the algorithm should output the masks of those objects in the rest of the frames.

Figure 8.2. The example annotations of the DAVIS 2017 dataset [148].
So, in the future, the visual tracking work will focus on how to use object detection and segmentation methods to boost the tracking performance at pixel level.
References


