TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

KF	Kalman Filter
KCF	Kernelized Correlation Filters
DCF	Dual Correlation Filters
CHT	Circle Hough Transform
GCHT	Gradient Circle Hough Transform
CCOEFF	Correlation Coefficient
CCOEFFN	Normalized Correlation Coefficient
CCORR	Cross Correlation
CCORRN	Normalized Cross Correlation
SQDIFF	Square Differences
SQDIFFN	Normalized Square Differences
HOG	Histogram of Oriented Gradient
GPT	Global Projection Transformation
SURF	Speeded-Up Robust Features
RANSAC	Random Sample Consensus
FOV	Field of View
GHT	Generalized Hough Transform
LKA2C	LuKas-Kanade Adapted for Coastal Changes
NDWI	Normalized Difference Water Index

DSAS	Digital Shoreline Analysis System
NYU	New York University
RGB	Red, Green, Blue
HSV	Hue, Saturation, Value
CMYK	Cyan, Magenta, Yellow, Black
MCSS	Multi-Color Scheme System
SIFT	Scale-Invariant Feature Transform
DMP	Deformable Part Models
IPST	Incremental Pictorial Structures
UAV	Unmanned Aerial Vehicle
DCN	Deep Comparison Network
RCNN	Recursive Convolutional Neural Networks

LISTE OF SYMBOLS AND UNITS OF MEASUREMENTS

fps	Frames per second
cm	Centimeter
μm	Micrometer
Hz	Hertz

INTRODUCTION

When a person watches a video he/she could easily distinguish between objects and realize what they are. Our brain could even keep tracking different objects in real time. Until today nobody has discovered how our brain works in this purpose or how it learns to do such tasks that most advanced algorithms cannot nearly do the same job.

By advancing technology, now computers can run same algorithms thousands of times faster than they used to do only two decades ago (Piguet (2018)), and reaching the goal of having performance as fast as human mind seems to be feasible. But speed is the most reachable feature. We are also interested to know how we perform tasks. Computers have different physical structure than humans. Even if we know how we think, it might not be possible to make computers learn the same way we do.

There are many reasons why tracking and detection of objects have high demands both in industry and research. But the goal is to let machines understand the surrounding environment like humans do. One very interesting and novel application that detects visual features to learn the environment and enhance the accuracy of GPS [1] is VPS (Visual Positioning System) recently introduced by Kaware (2018) used in "Google Maps" and "Waymo" autonomous cars. We are at golden age of developing autonomous cars. The cars that can see obstacles, track them or even predict their behavior in the near future. Detection and tracking of each part of human body is what is needed to reconstruct the 3D model of it (Kazemi *et al.* (2013)). These results could be used for other applications such as tele-immersion where a 3D model of each individual is reconstructed in a virtual simulated environment such as a virtual conference.

The main application we use to design our method and algorithm is detection and tracking of markers attached to shoulders, elbows and wrists of a physically impaired athlete while driving a three-wheel racing wheelchair. A Go-Pro forth generation camera is installed above the front

[1] Global Positioning System

wheel and captures at 240 fps[2] with the resolution of 1280×720. In the verification chapter, we compare our method with three other well-known algorithms in the conditions that satisfies our main application. Thus, videos of a person on wheelchair moving arms are used in the verification chapter.

Detection and tracking are open issues in general (Maciejewski *et al.* (2019)). There are many algorithms that propose solutions for specific applications. For instance if the problem is to find the position of the sun, a simple algorithm that searches for the brightest area is one solution. There are many algorithms that could detect a human face, or track one specific face between many (Raheem *et al.* (2019)) or detect and track a flying object (Agrawal & Dean (2018)).

Changing behavior of objects and surrounding environment makes the task of detection and tracking challenging (Chattopadhyay *et al.* (2019)). The object being tracked may rotate, illumination might change, reflection could appear, occlusions may happen, motion blur could happen, background might change and etc. All that happens because everything even the camera could be in motion in a video sequence.

In this research the focus is to detect and track sphere markers and find the pixel based 3D position of them and our assumption is that the true position is given in the very first frame. Most other researches work for specific objects and perform only detection or tracking but in this work we do both for each single frame. The desired output is the size and position of seen marker that is equivalent to 3D position of marker with respect to camera. This research contributes to following results:

- We implement recent and well-known methods such as Kernelized Correlation Filters (KCF), Channel and Spatial Reliability (CSRT), Boosting, and others and compare their accuracy and reliability in our main application in detection and tracking of sphere markers.

[2] frames per second

- We optimize the Circle Hough Transform (CHT) function based on its circle edge detection inputs considering the main application of detection of sphere markers.
- We develop a color based detection method and optimize the color ranges to minimize false object detection and verify the results of edge detection technique.
- We fuse the Histogram of Oriented Gradient (HOG) and KCF to boost the robustness of KCF against partial occlusion.
- We Develop a point tracker to fix the KCF incapabilities in scale variation to propose a more proper template size to the detection unit.
- We use six different template matching units to verify the detection results. Each method votes for correctness of detection and different scenarios are designed to consider the required action based on given votes.
- We implement a Kalman Filter and develop and extend it to 3D position prediction. Kalman filter is firstly used to smooth the final results and remove fluctuations and noise. Secondly, KF predicts the position of the target in the next frame to reduce the chance of failure in the future frame. Thirdly, KF would correct the results if template matching verification unit does not verify the presence of the target.

The purpose of this study is to build a base infrastructure that could be used in many applications such as reconstructing the 3D model of the human body using only one camera and multiple markers attached to joints. Most relevant studies are based on using multiple cameras in specific studios designed to reconstruct the 3D model used for applications such as tele-immersion (Duncan *et al.* (2019)). Although these methods seem to work in an equipped lab with special processors, the person being captured needs to stay at a certain place and move slow enough for the algorithm in order not to lose the track of body (Duncan *et al.* (2019)). In

this study we designed an algorithm that does not require multiple cameras and expensive processors. To gain more accuracy in fast motion, the camera will capture video with 240 frames per second frequency. Since the it's not necessary for our algorithm to be used for online tracking (real-time tracking is not necessary), we can pay as much performance as is necessary to increase the accuracy and robustness. The one drawback of marker based detection algorithms is their inherent incapability to be robust against total occlusion.
The result of this study could be also used for other applications such as for robot arms playing ping pong, soccer robotics, etc.
The goal of tracking an object in reality only tries to find a similar image patch of one frame in the next one (Keuper *et al.* (2018)). It does not necessarily understand what that image patch contains. There could be a part of an object, multiple objects or just some part of background that tracking algorithm follows. But detection part tends to find a specific object [3].

In Chapter one, we discuss the recent and similar methods in details that are well-known in the literature. In the Chapter two, we explain our own proposed method and show the results when we applied it to the experimental data. In the third Chapter, we compare our method with three other candidates in terms of accuracy and failure. The same method will validate the results of the proposed method. In the Chapter four, we give a conclusion of our procedure and what the possible future works could be.

[3] In our study it is a sphere marker

CHAPTER 1

LITERATURE REVIEW

Detection and tracking of objects in a video sequence is a task many researchers have been focused on during past 40 years. There are various theories and algorithms that have been developed for general and specific purposes. Tracking markers with a known shape or just a moving object with no specific structure and recently we see trackers which learn the shape of the tracking area or detection techniques that recognize the name or type of the object it's detecting.

The core of most tracking methods is template matching. It is the task of finding the position of a subimage called a template inside a large main frame that is called a search area or region Yang *et al.* (2018). By shifting the template image over all possible places in the search area and measuring the similarity between the template and the selected window in search area, we will find a place in the large image where the template has the most similarity with it Yang *et al.* (2018).

1.1 Template Matching

In this section we discuss the building blocks for template matching introduced in the literature and will be used later in our own research partially or with modifications.

1.1.1 Integrated HOG

Template matching was subject of much research since 1960s (Guo *et al.* (2019)). Most older techniques were based on correlation matching methods and are suitable for "whole-to-whole" template matching (Zhang *et al.* (2017)). In more general circumstances image matching is challenging because of complex backgrounds and noise. Zhang *et al.* (2017) proposed an

HOG (Histogram of Oriented Gradients) patterns for the improved GPT (Global Projection Transformation) matching to gain the robustness against noise and background by using norm normalization. Investigates using the Graffiti dataset revealed that this suggested approach comparing with the original GPT correlation matching and the integration of Speeded Up Robust Features (SURF) feature descriptor and Random Sample Consensus (RANSAC) method is capable of an excellent matching (Zhang *et al.* (2017)). Moreover, the computational cost of the suggested approach decreased dramatically.

Image 1 in Figure 1.1 is a template sample of size 160 $\times$ 126 pixels, Images 2 to 6 are input

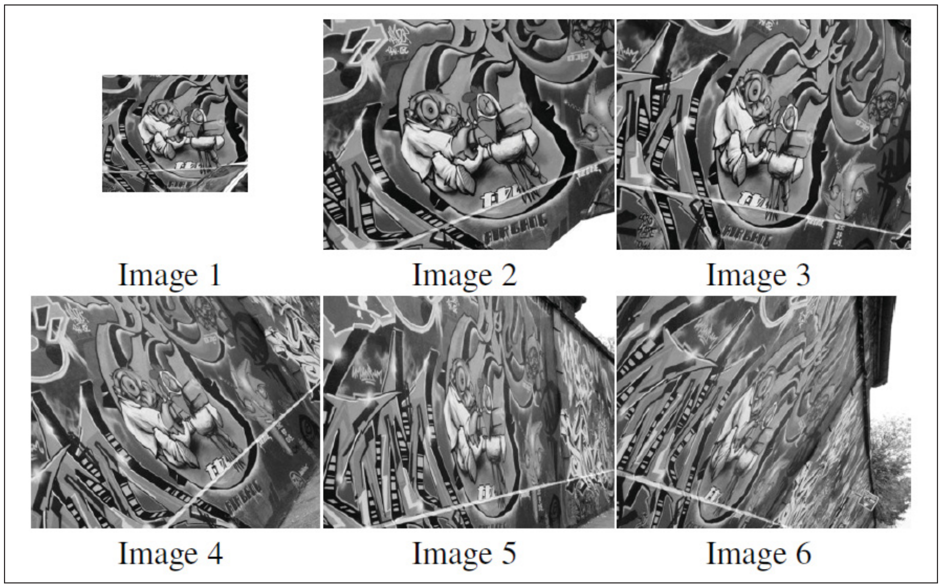


Figure 1.1 Graffiti dataset. Reference: Zhang *et al.* (2017) page 5

images different views of a search area shows the proposed methods that uses HOG is more effective than the conventional Non-Stationary Gaussian Processes Tomography (NSGPT) solution (Zhang *et al.* (2017)). The two main improvements are:

- Proposed technique requires significantly less iterations to get to the maximum Correlation
- If field of view (FOV) is wide, suggested method reaches higher correlation compared to the method that isn't integrated with HOG.

Histogram of Oriented Gradients (HOG) provides a very useful tool that increases the accuracy for more general FOVs and improves the performance of template matching and tracking. Thus, HOG is combined with Kernelized Correlation Filters (KCF) for our own purpose in tracking and it will be discussed later in Chapter 2.

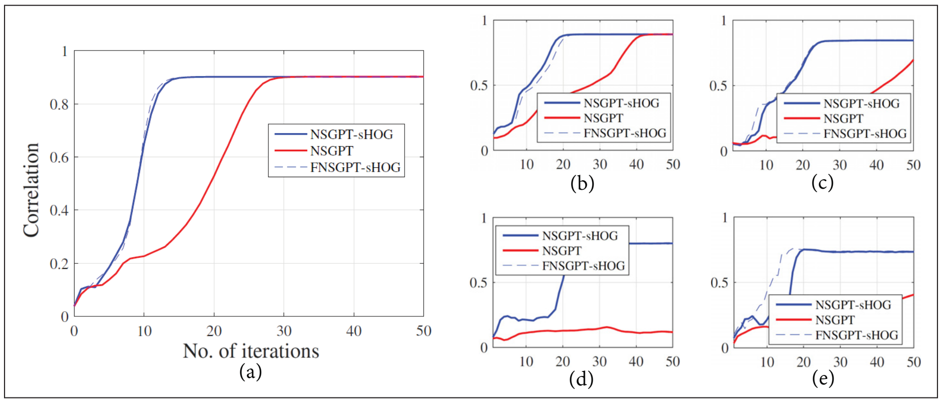


Figure 1.2 Comparison of correlation values and number of iterations in GPT matching with and without HOG between Image 1 against a) Image 2 b) Image 3 c) Image 4 d) Image 5 and e) Image 6.

Figure 1.2 shows how HOG lets NSGPT to achieve higher correlation in fewer iterations.

1.1.2 Memory Efficiency

Mun & Kim (2017) modified GHT (Generalized Hough Transform) to decrease its computational complexity and memory requirement and enhance its performance under rigid motion. In the suggested method, orientation and displacement are considered individually by using a multi-stage structure, and the displacement collector–that uses more memory than others–is downsampled without reducing detection accuracy. Additionally, an adaptive weight scheme is used to make the template position more trustworthy. Experimental test outcomes express that the suggested scheme has benefits in memory requirement and computational cost comparing with conventional GHT, and pose estimation becomes more steady.

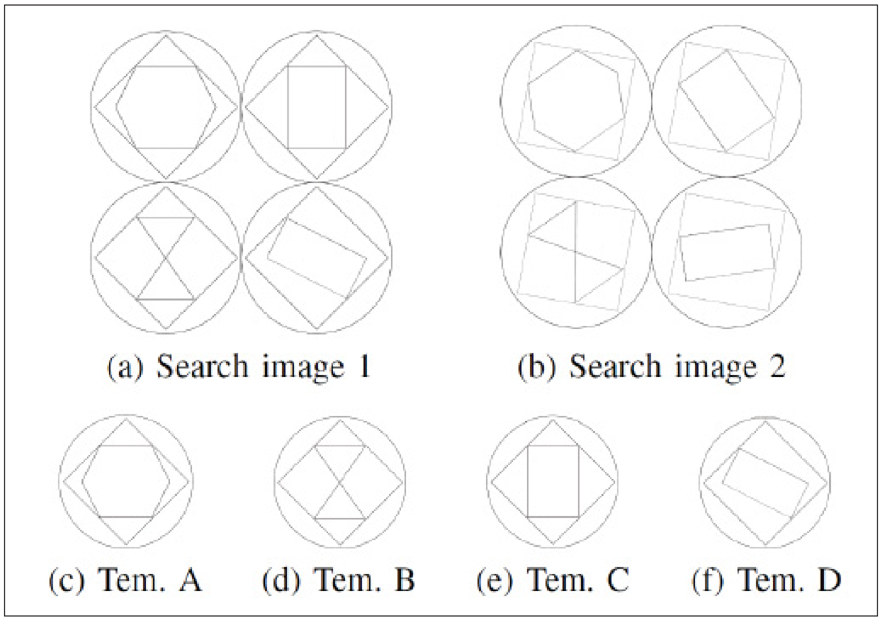


Figure 1.3 Confirming the successfulness of adaptive weighting scheme. (a) Search area with no rotation. (b) Search area rotated 325 degree. (c)-(f) Template patches.

Figure 1.3 shows robustness against rotation and tables 1.1 and 1.2 shows the process of voting to find the best match template for each region.

Table 1.1 Voting outcomes for constant weighting method. Reference: Mun & Kim (2017) page 6

		Constant weight				
	Template	**Region A**	**Region b**	**Region C**	**Region D**	**Rate**
search image 1	A	1389	1199	1104	906	1.158
	B	1310	1388	1228	976	1.060
	C	890	816	1873	692	2.104
	D	585	525	527	635	1.085
search image 2	A	186	166	166	150	1.120
	B	199	200	183	140	1.005
	C	186	185	207	156	1.113
	D	167	166	151	180	1.078
Average		-	-	-	-	1.216

Table 1.2 Voting outcomes for adaptive weighting method. Reference: Mun & Kim (2017) page 6

		Adaptive weight				
	Template	**Region A**	**Region b**	**Region C**	**Region D**	**Rate**
search image 1	A	460.01	657.15	337.98	261.69	1.288
	B	388.10	521.12	360.64	280.76	1.343
	C	256.61	238.32	851.26	196.33	3.317
	D	169.11	148.00	146.15	219.04	1.295
search image 2	A	131.67	109.53	1165.52	72.16	1.130
	B	132.37	146.19	117.28	76.61	1.104
	C	147.21	116.51	257.44	78.7	1.749
	D	96.25	94.56	64.71	110.16	1.145
Average		-	-	-	-	1.546

GHT is robust against partial occlusions, chaos, arbitrary brightness variation, and noise compared with state-of-art pattern matching methods (Mun & Kim (2017)). On the other hand, it has very high memory requirements to implement in product inspection. Mun & Kim (2017) suggests a slim GHT suitable for product inspection that decrease memory requirements and computational complexity. Moreover, its reliability is by making its weight growth scheme adaptive, it is particularly effective when the template size is too small or when analogous patches have appeared in the search area reference image. In this simulation, it is claimed that a defined stability is improved by about 27 %. Moreover, slim GHTs memory requirements and computational costs are about 0.002% and 6% of CGHTs, correspondingly.

1.1.3 Enhanced Normalized Cross Correlation

Pontecorvo & Redding (2017) discusses a particular example of non-periodic conversion symmetry. In this method they propose to observe consistent and overlapping regions of self-resemblance through a non-urban scene and proposed a method that automatically detects various poles[1], or their shadows. The approach does not depend on having a prior pole sample or knowledge of its precise size. By using normalized cross-correlation, analogous areas across the entire photographed picture are found. By estimating the size of the pole and its given or

[1] Standing poles such as electricity poles and pylons

obtained alignment, the blobs could be refined. The suggested technique then shows a mutual amount of self-similarity between similar patches by clustering together all image patches that have commonly overlapping blobs. For non-urban areas, it is likely to detect identical or analogous poles. Experimental results on real aerial imagery show that this method with only small number of false alarms can potentially detect almost any pole , and performs with greater performance compared with state-of-art template-matching methods (Pontecorvo & Redding (2017)). The following limitations should be considered while applying the algorithm:

- The algorithm to detect blobs and filter background objects must assume configurations of pole-shaped objects.
- The pole size should be assumed as given a priori in the main image and the sensor metadata.
- In case of few self-similar designated objects in the captured picture, the self-similarity detection methods may not be able to perform as expected (The worst-case scenario)
- Ideally, minimum of 4 or 5 targets of interest should be visible in the image.
- There is no easy approach to sample an $H \times W$ image densely [2], where HW correlation images have to be considered. It requires the total memory of $(H \times H \times W \times W)$ tensor should be stored. This requirement could be decreased by using a stride but it has to be chosen carefully to make sure the target is not ignored sinking among the sample pixels

Pontecorvo & Redding (2017) have offered a different pole detector in airborne electro-optical imagery based on the idea of image-wide non-periodic translation symmetry. At first they computed the 4D tensors of thresholded correlation images for a subsample of image pixels by assuming that the camera inspecting geometry is known along with the estimated pole size. Secondly, they established a sequence of filtering steps to eliminate any blobs not shaped and oriented like the expected pole from the thresholded correlation pictures. In conclusion, a clustering technique is developed to collect all pixels with enough numbers of commonly overlying

[2] high sampling rate

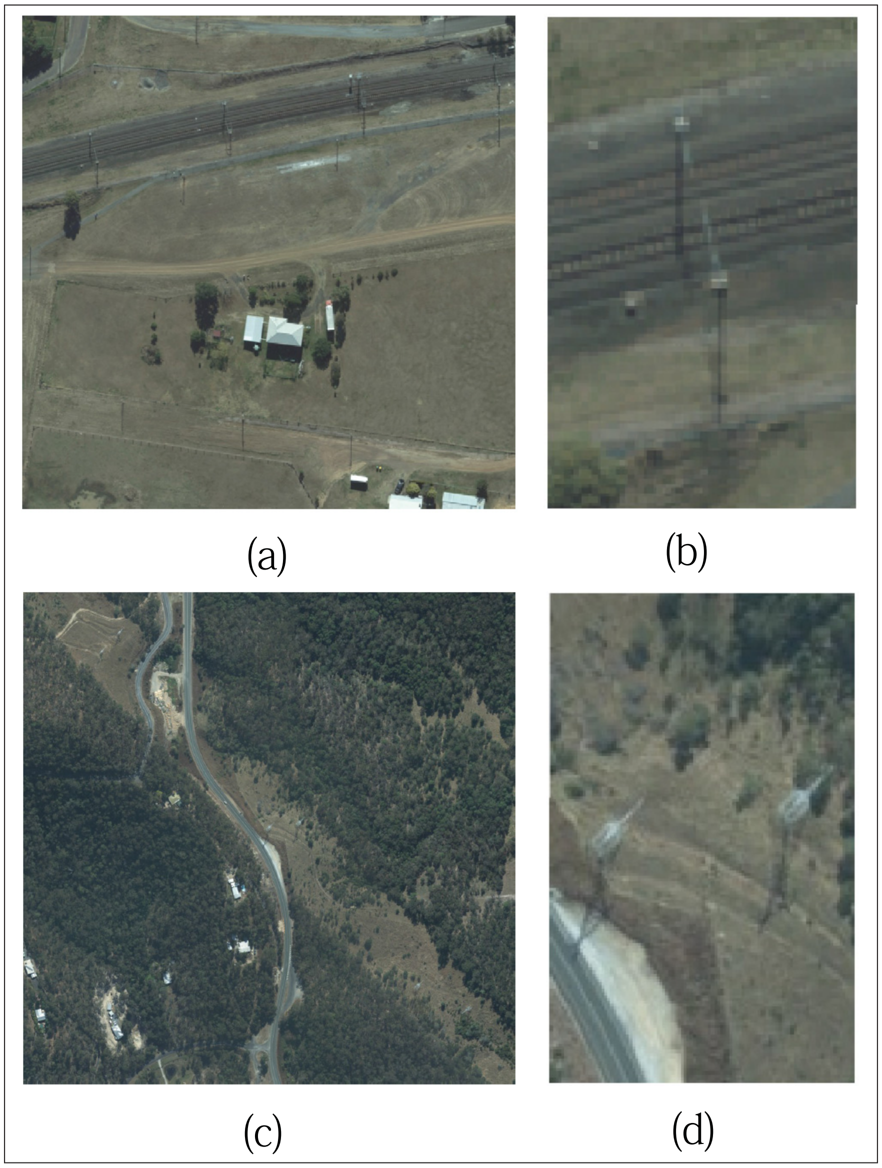


Figure 1.4 a) Rural area image with visible pole shadows around the dirt roads and rail tracks, b) Close up views of rail poles, c) Rural area image with electricity poles and pylons. d) Close up views of electricity poles or pylons. Reference: Pontecorvo & Redding (2017) page 5

filtered blobs. They then showed that most of the poles available in the image are detectable by the largest clusters with just a few false alarms by using this method for the detection of poles in two airborne images of non-urban sights. Experiments show this method detected more targeted objects with fewer false alarms and it's compared to state-of-art template matching method, which assume a target template is presented a priori.

1.2 Edge Detection

One well-known method of detecting an object is by recognizing it by its edges (Goodsitt *et al.* (2019)). An object does not normally share the same color, saturation and brightness with background or surrounding objects. Thus there is an instant change in pixels color, saturation and brightness at the border with background in one side and the desired object at the other side. Multiple methods are proposed in the literature to detect these edges and the applications are varied (Goodsitt *et al.* (2019)). Shadow, reflection, motion blur and low resolution could make the task of edge detection challenging. Here we discuss some effective methods introduced in the literature which will be used in our own method with some modifications.

1.2.1 Canny Method

Bouchahma *et al.* (2017) presented a method called LuKas-Kanade Adapted for Coastal Changes (LKA2C). This technique automatically analyzes and detects shoreline gradual alterations captured by satellite. It measures the changes by analyzing the shoreline images around the study area (Bouchahma *et al.* (2017)). The SURF algorithm is the base of this suggested technique. Then, Canny edge detection was used on segmentation of NDWI (Normalized Difference Water Index) image components to detect the edge of shorelines. Finally, the pyramidal Lukas-Kanade optical flow algorithm was modified and used to measure and find the amounts of alterations. Experiments on real satellite images captured the island of Djerba in Tunisia proved the usefulness of the suggested technique.
This is not the first use of Canny edge detection (Bouchahma *et al.* (2017)), but the method Bouchahma *et al.* (2017) developed to modify images to get the most efficiency of Canny is quite informative.

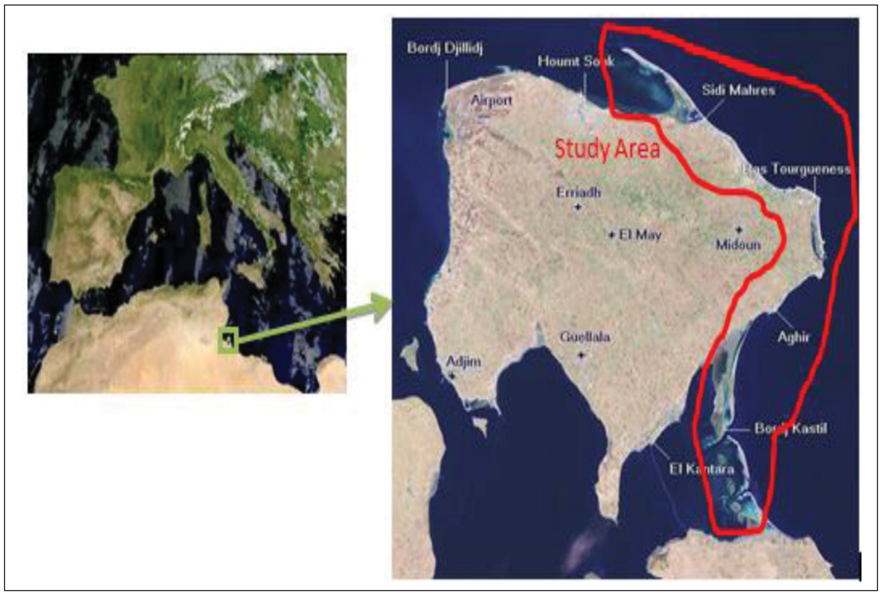


Figure 1.5 Study area captured by satellite. Reference: Bouchahma *et al.* (2017) page 2

Conclusion: In this study, satellite images are used to detect and objectify shorelines. for 1984 and 2015 by segmentation of the NDWI components with histogram based on automatic thresholding method. The shorelines change around Rass Errmal, Aghir and Elkestil were calculated by an algorithm called LKA2C. the algorithm start by detecting the area of interest using SURF as a registration technique (Bouchahma *et al.* (2017)). Then, the shorelines have been extracted using Canny edge detector. The measurement of changes has been calculated using Lukas-Kanade algorithm. To validate the approach the result is compared using a manual approach based on DSAS (Digital Shoreline Analysis System).

In figure 1.5 the selected shoreline is shown in a satellite photo and the figure 1.6 shows how the shoreline edges change over time.

1.2.2 Customized Efficiency

Edge detection is the core component of most object detection and image segmentation techniques (Chen *et al.* (2016)). Local structures are usually a set of image patches separated by local edges such as T-junctions in the image. Dollár & Zitnick (2015) proposed a compu-

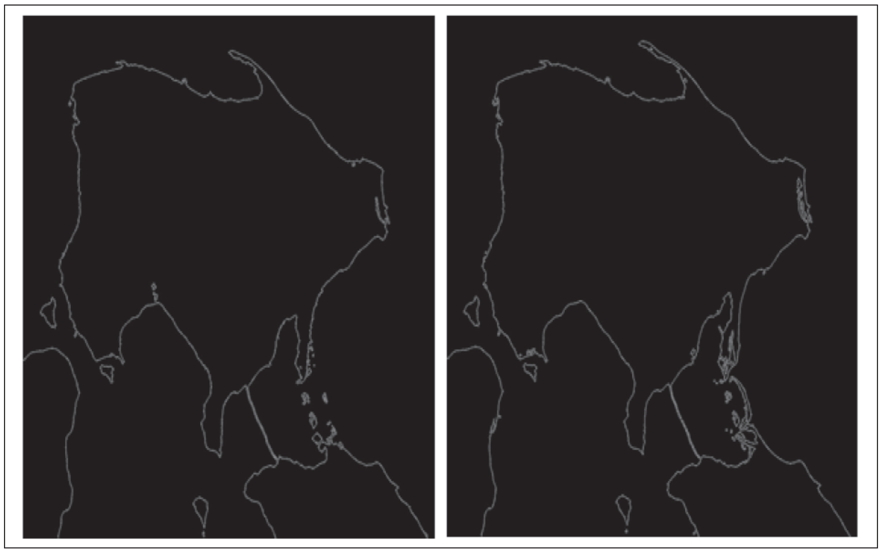


Figure 1.6 Extracted Edges of Shorelines. Reference: Bouchahma *et al.* (2017) page 3

tationally efficient edge detector by learning from edge structures acquired in localized image patches. The proposed new method tries to learn decision trees that robustly maps the structured patterns to a discrete space on which standard information gain measures may be evaluated. The outcome is an algorithm that achieves real-time performance that is orders of magnitude faster than many other approaches, while achieving state-of-the-art edge detection results on the BSDS500 Segmentation dataset and NYU Depth (Dollár & Zitnick (2015)). Finally, it is showed potential of the suggested method as a general purpose edge detector by showing that their models generalize well across datasets.

This method is capable of real-time frame rates (30 fps) and reaching state-of-the-art accuracy (Dollár & Zitnick (2015)). It could make new applications capable of needing more efficient high-quality edge detection. For example, it could be well suited for video segmentation and time sensitive object classification tasks such as pedestrian detection. Structured decision trees could be learned by the suggested method and could be useful for various problems. The direct and fast inference process is ideal for applications demanding computational efficiency. Several vision applications contain structured data, there is substantial potential for structured

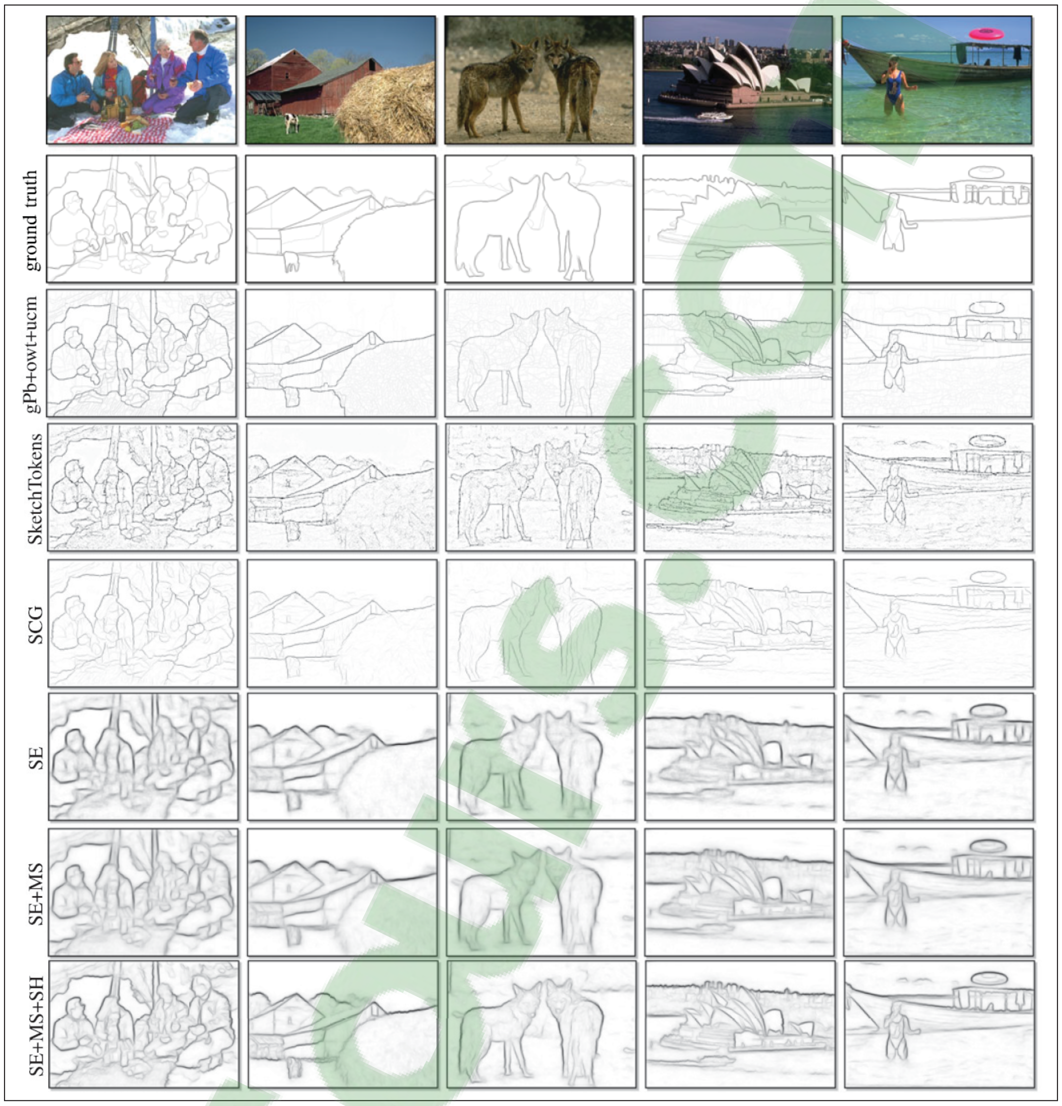


Figure 1.7 Results of edge detection on the BSDS500 dataset on five different sample pictures. The first row is the original image, the second row is ground truth. The third row show results for gPb-owt-ucm. The next two rows demonstrate results for Sketch Tokens, and SCG. The last four rows contain proposed results for variants of SE. Reference: Dollár & Zitnick (2015) page 6

forests in other applications. In conclusion, Dollár & Zitnick (2015) suggest a structured learning approach to edge detection and define a general purpose technique for learning structured arbitrary decision forest that robustly uses structured labels to choose splits in the trees. The state-of-the-art precisions are demonstrate on two datasets, while it is many times quicker than most competing state-of-the-art techniques.

1.3 Color Detection

One specific feature of an object is its color since colors do not change during a video sequence. Hence, one effective approach is to detect an object based on its color. An important assumption should be uniqueness of a color assigned to the object with respect to surrounding environment. This assumption could limit the generality of any-color backgrounds and let algorithms fail if there is similar colors close to the target. One method to make a color-detector more robust to such problem is to combine it with other methods such as edge detectors.

1.3.1 Integration of Edge and Color

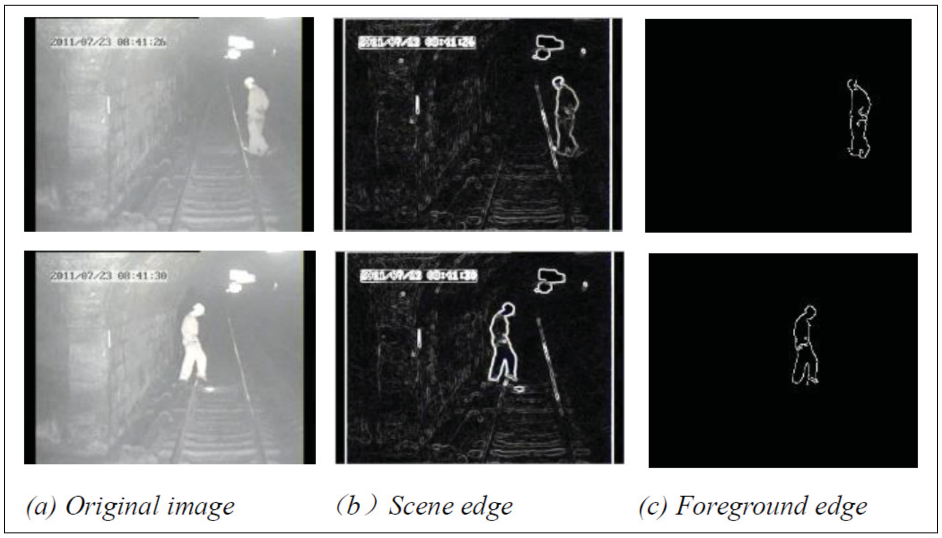


Figure 1.8 Edge detection unit. Reference: Xian *et al.* (2017) page 3

In a coal mine, object and background have both similar gray under low lighting, and quick illumination variations can cause false target detection (Xian *et al.* (2017)). Due to the fact that the detection technique based on Gaussian mixture model simply evaluate the color information, problems of undetected objects and false detections appear (Xian *et al.* (2017)). An enhanced object detection method is proposed by Xian *et al.* (2017), which linearly integrates the color and edge information. By using Gaussian Mixture Model fitting the color information of background, extracting edge information of the image as supplement of color information, and classified edge as foreground and background edge, then the color background subtraction model and edge background subtraction model are established and normalized, finally the two information are linearly integrated to detect the miners and other objects. The simulation results (Xian *et al.* (2017)) prove that the suggested algorithm can efficiently detect the object of similar gray with the background and eliminates the false moving object caused by sudden brightness variation, and thus increase the accuracy in the detected object.

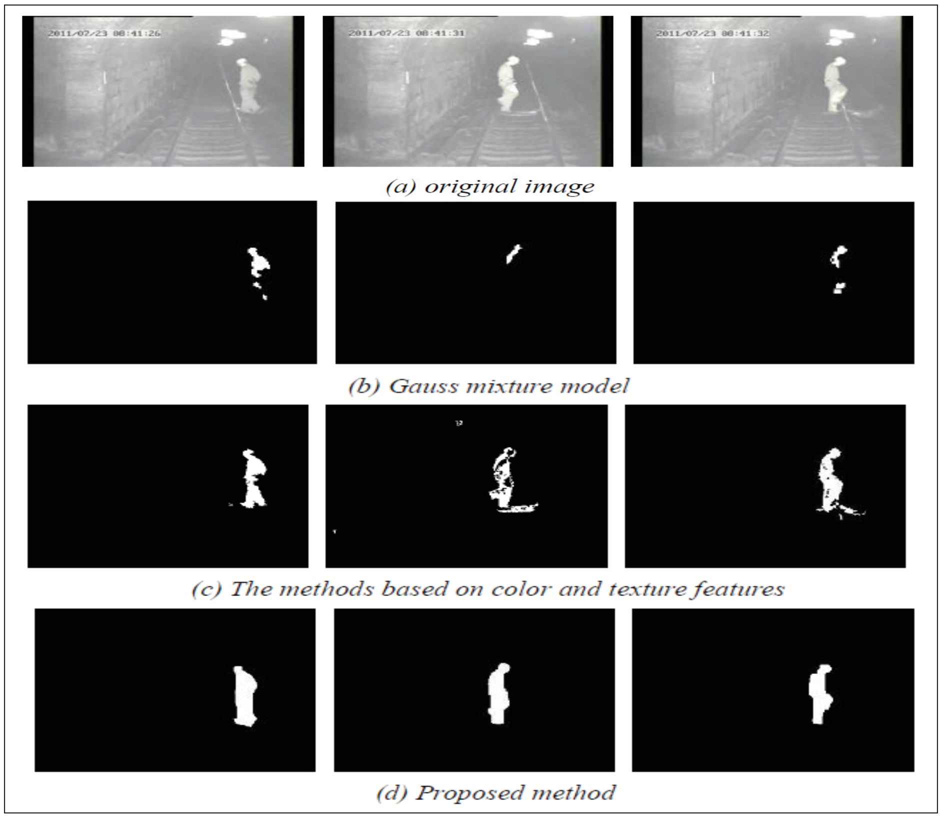


Figure 1.9 Experimental results comparison. Reference: Xian *et al.* (2017) page 5

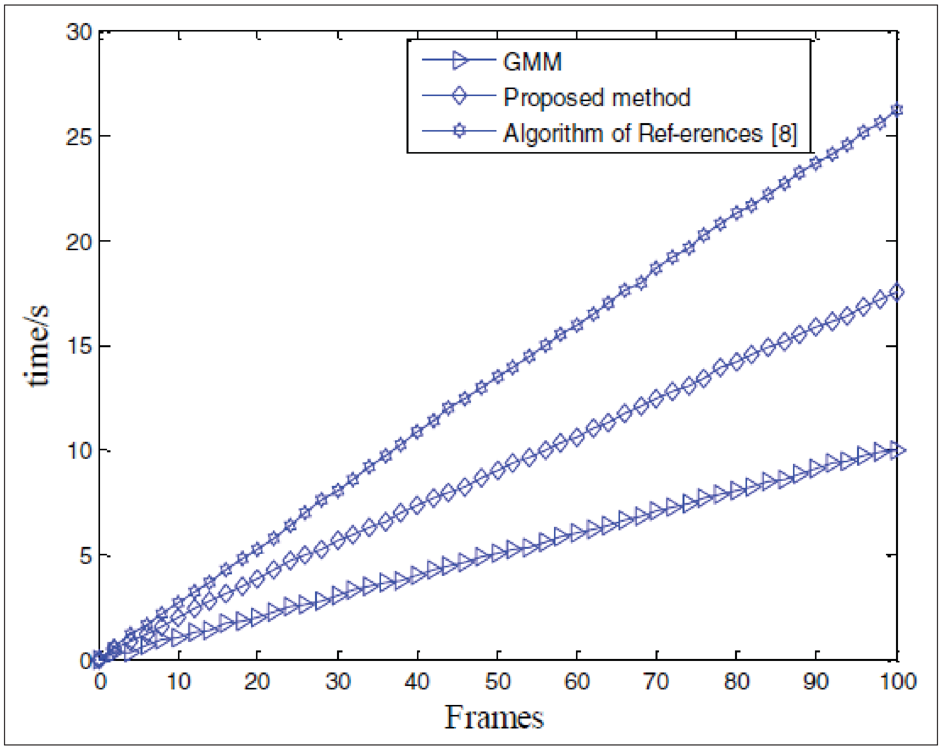


Figure 1.10 Time cost analysis. Reference: Xian *et al.* (2017) page 5

Xian *et al.* (2017) proposed a new approach based on integration of color and edge information to detect the miner as target in mine environment. Experimental results indicate that it can eliminate the false moving object caused by sudden brightness change and detect the true moving target efficiently. The weakness of suggested method is that the threshold is determined manually, which means the value may not be optimal. By use of the statistic information of the picture, an adaptive threshold technique could bring us to optimum results.

1.3.2 Multi-color Spaces Combination

Pichai & Kumar (2017) proposed a novel color detection method by combining RGB, HSV and CMYK color schemes for digital images. This approach is a preprocessing algorithm to analyze images before getting used by applications such as face detection and object recognition. Object positioning task could be done by using Multi-Color Scheme System (MCSS). MCSS is a known method in digital image processing for human body part segmentation. It detects skin color based on processing patterns through neighborhood pixels. It is verified that this method outperforms the state-of-art color object localization techniques. It is also proved that the detection accuracy and stability has an edge over the best facial color detection algorithms so far. Compared with skin texture detection algorithms that are based on facial-parts geometries, the proposed method claims to have less computational complexity A set of experiments is designed to give an intuition for comparing and evaluating the integration of multi-color systems used in MCSS. RGB, Hue and CMYK are the three multi-color band frequently used in MCSS. Thus the skin detection algorithm is fused with multi-color scheme and it's showed that this method is more precise than state-of-art single channel color space skin detection techniques. Because of its performance and accuracy, this technique is confirmed to be superior to the recently reported competitive methods.

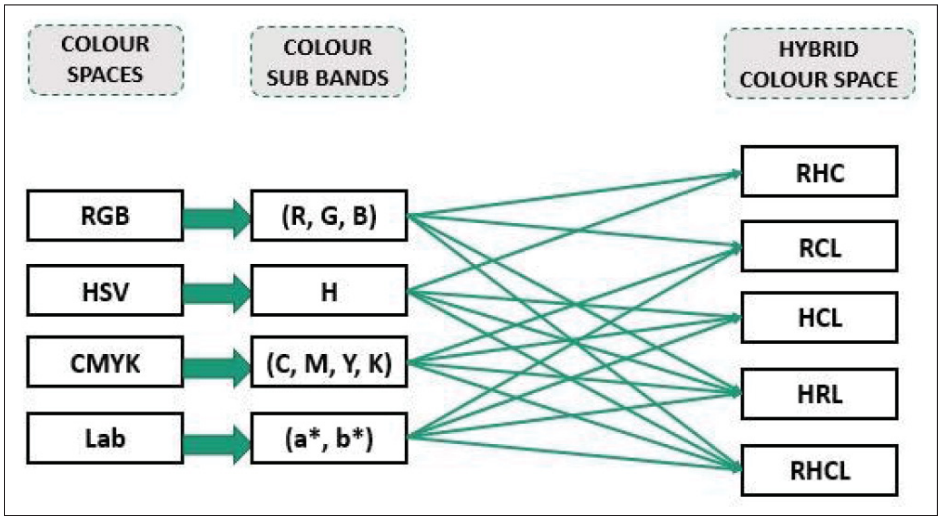


Figure 1.11 MCSS Diagram and Structure.
Reference:Pichai & Kumar (2017) page 3

Figure 1.11 demonstrates the technical process of producing different hybrid color spaces and figure 1.12 compares the results of mapping a sample image to all those spaces. At first, the input RGB picture is transformed to CIE-Lab, HSV and CMYK. The chosen components for this study are:

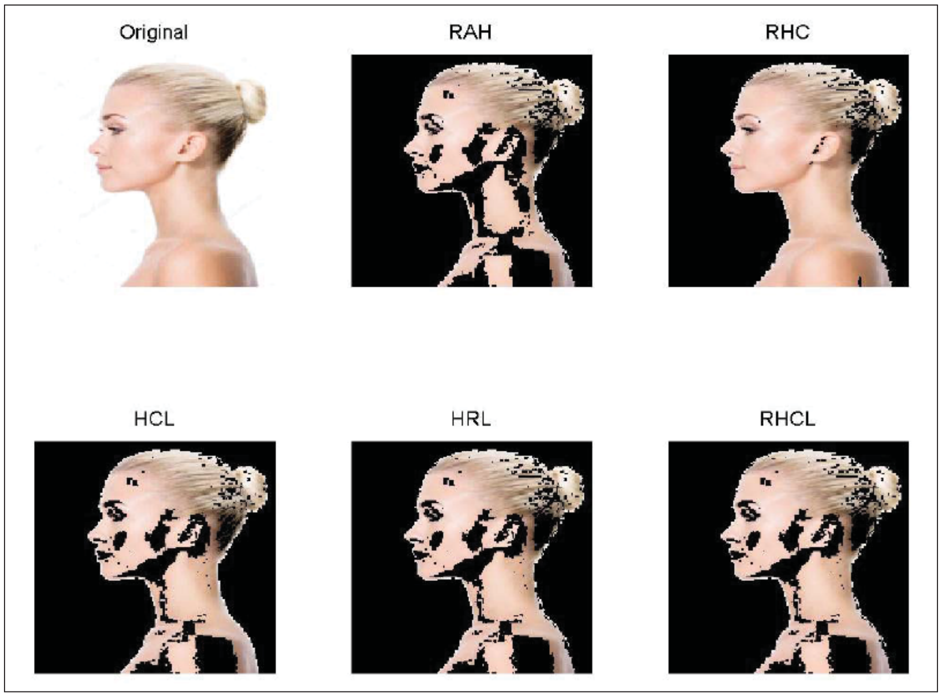


Figure 1.12 Sample from Profile database. Reference: Pichai & Kumar (2017) page 4

- a*b* elements of CIE-Lab
- Hue element of HSV
- Cyan, Magenta,Yellow and Black components of CMYK

Then HRL, RHCL, RHC, RCL and HCL are the five hybrid spaces created by the components above. Experiments verified that for the purpose of skin identification, RHC shows more accuracy compared to all the created multi-color spaces. It also outperforms the Rahman's method designed for color-based face detection. Because of its merits, this skin detection method finds place in face and emotion detection and face identification.

1.4 Circle Detection

The goal of this study is to detect and track sphere marker(s) attached to human body. We can now start taking a closer look to methods that are designed for specific objects. It's circle in our case.

1.4.1 Gradient Hough Circle Transform

Cornelia & Setyawan (2017) proposes a novel technique for detecting circular objects in response to the new alterations into the principles on Kontes Robot Sepak Bola Indonesia (KRSBI) division for KRI 2017. The suggested algorithm by Cornelia & Setyawan (2017) is based on both the contour and Gradient Hough Circle Transform (GCHT) that is designed to detect round objects regardless of the color. The objects is firstly detected by the contour based method and then it would be verified by GCHD. The GCHD could be used as a back up technique if contour fails. The investigation shows that the suggested algorithm can reach ball detection accuracy up to 100% for orange ball and up to 96% for white ball if balls and camera are not moving and lightning conditions are predetermined. In a video sequence of objects moving randomly but on a fixed brightness properties, the algorithm reached an accuracy of 93.3% for orange ball and 96.6% for white ball. Furthermore, the processing time consumed in the experiments are short, for an average frequency of 37.76 FPS.

Cornelia & Setyawan (2017) has proposed a novel circular object detection method designed for soccer robots based on the Contour and GCHT techniques. Experimental results indicate that the suggested method works well in detecting both white and orange round objects on green background. In this work, Cornelia & Setyawan (2017) claimed that they improve the performance and stability of the algorithm. Particularly, the older algorithm should have been enhanced so that it could more reliably find the white balls at far more distances. Moreover, to be made more adapted to real-world possibilities, there are circumstances in which partial occlusion happens by other robots. Hence, Cornelia & Setyawan (2017) also enhanced the reliability of algorithm to better deal with occlusion problem.

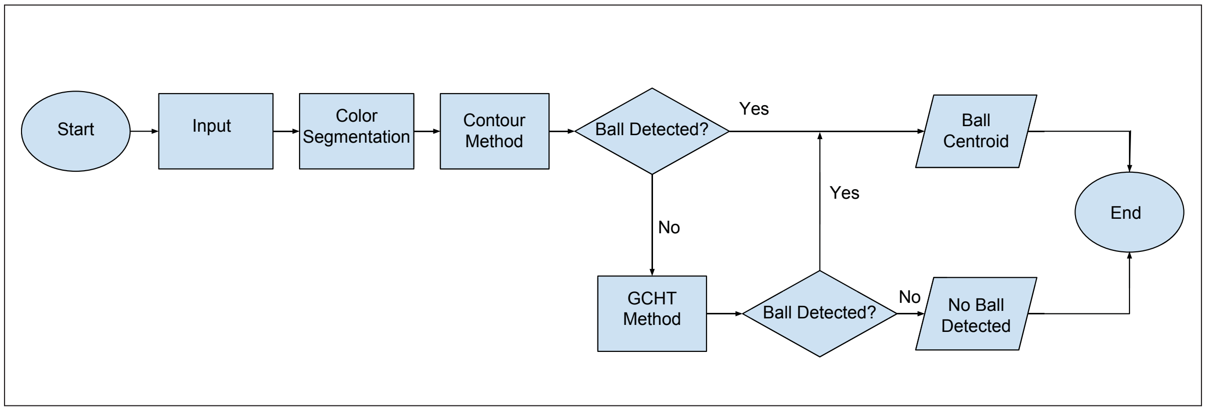


Figure 1.13 Flowchart R2C-R9 conventional algorithm. Reference: Cornelia & Setyawan (2017) page 2

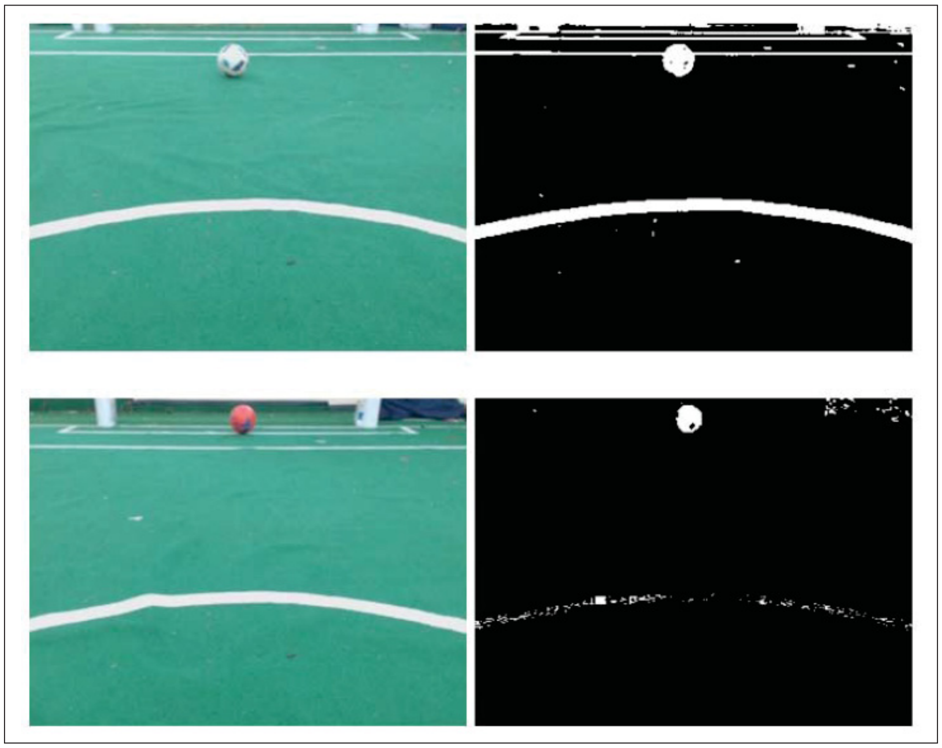


Figure 1.14 Testing an orange and a white ball with similar sizes and different distances. Reference: Cornelia & Setyawan (2017) page 6

Table 1.3 compares the results of this method in two balls with different colors, white and orange.

Table 1.3 Performance comparison for white and orange ball for contour and GCHT ball detection technique with random placement. Reference: Cornelia & Setyawan (2017) page 6

Color	Frame	Detection	False Rate (%)	CF	GCHT	FPS
Orange	30	28	6.66	25	3	39.16
White	30	29	3.33	25	4	36.79

1.4.2 Threshold Segmentation

Flexible printed circuit board (FPC) is a common substrate for packaging integrated circuits (ICs). Detecting the circles rapidly on FPCs in computer vision is very important to assess the quality of FPCs during its manufacturing. Luo *et al.* (2017) introduced a fast circle detection method based on a threshold segmentation technique and a validation check is suggested. An image is initially segmented by an adaptive heuristic threshold to take closed contours; then circle candidates are gotten by removing apparent non-circle contours; and eventually, circle candidates are further confirmed to be circles based on Helmholtz principle. Test results indicate that the proposed technique is very fast and has high detection accuracy to detect the circles on FPC pictures. Figure 1.15 compares the result of the method proposed and the EDCircles.

1.4.3 Reshaped Circles in Real 3D World

Spherical object detection is of great importance in various areas, for example computer vision, image processing, pattern recognition, etc (Chen *et al.* (2017)). Most current techniques for circle detection are 2D-based algorithms, which are designed such that only perfect 2D circles in the image plane can be detected (Chen *et al.* (2017)). Round circle shape objects in 3D real world are typically ellipses in 2D images instead of standard 2D circles [3]. Subsequently, most common approaches according to 2D images are incapable to separate physical space circles from ellipses. To solve this problem, Chen *et al.* (2017) proposes a 3D circular object detection technique according to binocular stereo vision. At first, it detects and fits ellipses/circles in stereo pictures, then achieves sub-pixel-level disparity data between two pictures by stereo

[3] depending of the different angles they expose to camera

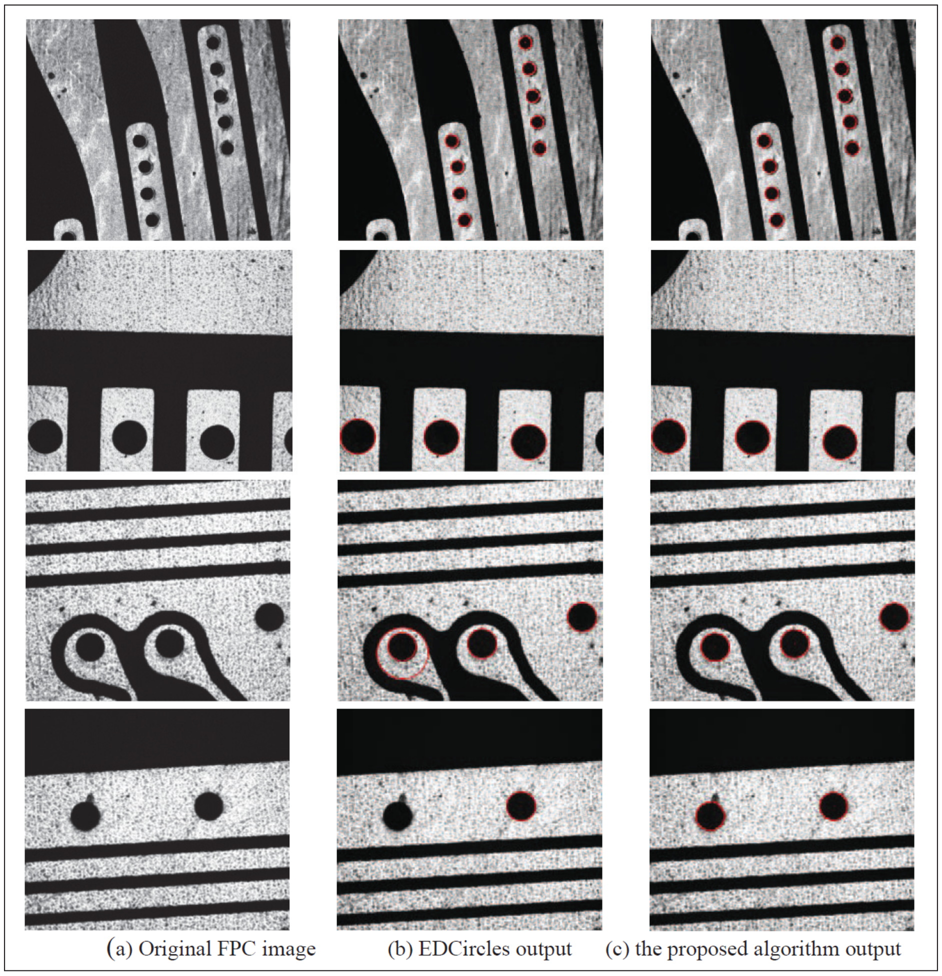


Figure 1.15 Comparing different algorithm results. Reference: Luo *et al.* (2017) page 4

matching with the required mathematical models. Then, based on the binocular stereo vision model, the disparity data is reversely projected into 3D space. By the preset thresholds for parameters to assess the extent deviated from round and coplanar objects, space circular objects can finally be detected based on those 3D data. Several experimental results reveal that the suggested technique can detect round objects well with high accuracy and efficiency.

1.4.4 Soccer Robot Example

Ball detection and tracking is the most important task in “Automatic Soccer Robot Competition” (Shah *et al.* (2018)). A common method used color to track ball(s) (Dewi *et al.* (2019)).

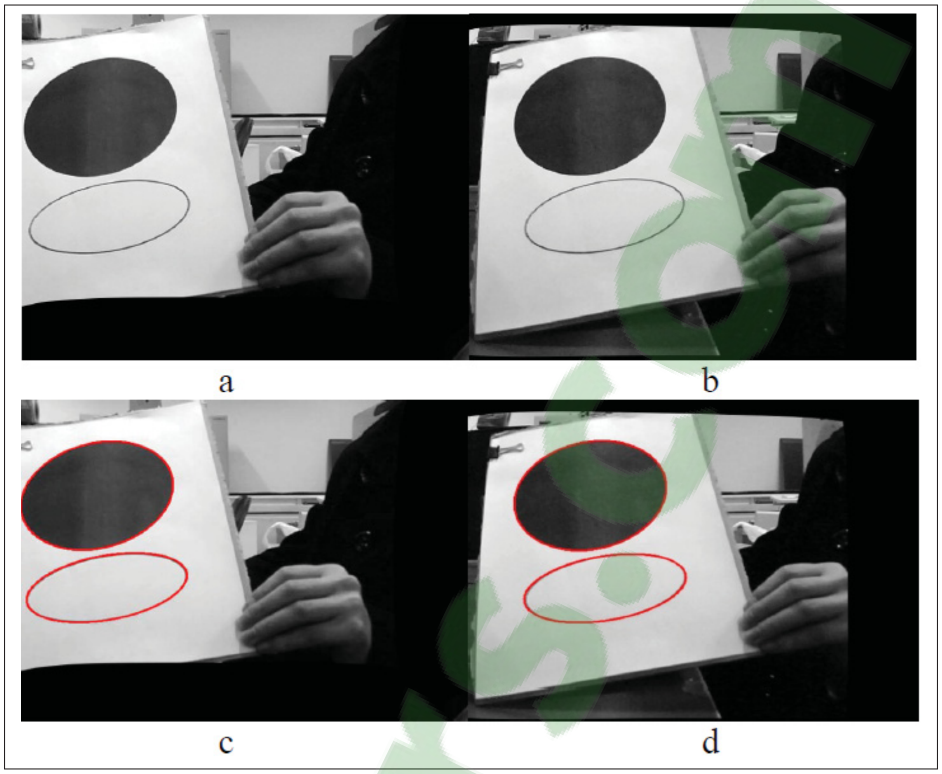


Figure 1.16 The 3D circles detection results a) First original sample image; b) Second original sample image; c) The detection result for first sample; d) The detection result for second sample. Reference: Chen *et al.* (2017) page 4

The main issue of color based technique is that the color values of a ball can change based on the illumination condition of the environment. The brightness problem make the operator manually change the color range used to track the ball. This way is not optimal because the operator has to test it many times and results have normally more error. Putri *et al.* (2017) proposes a method where the robot is able to find the color range of the ball automatically. The approach is to use "gradient hough circle" to detect the ball in the image and then get all the ball's pixel values. Then standard deviation is used to the ball's color values data to remove outliers. Finally, from the new data, lowest and highest value for the color range is obtained and used in the color-based object tracking.

Color-based tracking is a well-known technique for tracking single color objects. The biggest challenge is to find an optimized range of colors (Sarkar & Martin (2019)). By using the

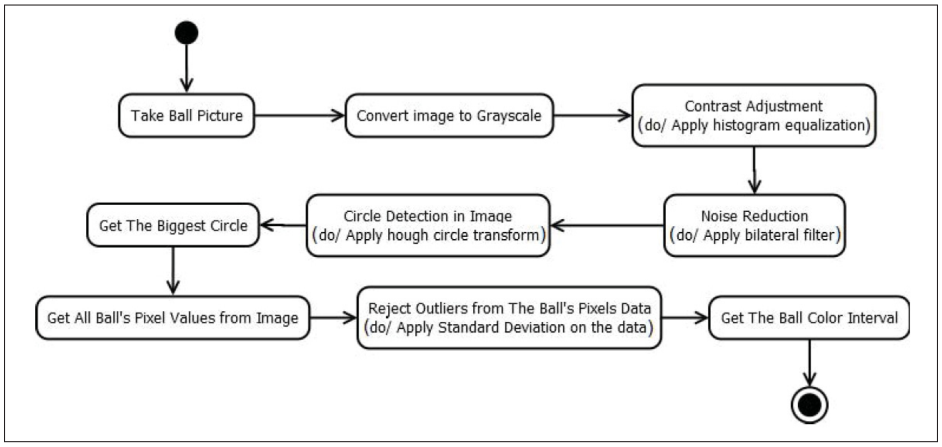

Take Ball Picture
Convert image to Grayscale
Contrast Adjustment (do/ Apply histogram equalization)
Noise Reduction (do/ Apply bilateral filter)
Circle Detection in Image (do/ Apply hough circle transform)
Get The Biggest Circle
Get All Ball's Pixel Values from Image
Reject Outliers from The Ball's Pixels Data (do/ Apply Standard Deviation on the data)
Get The Ball Color Interval

Figure 1.17 Proposed Method. Reference: Putri *et al.* (2017) page 1

methods above, processor can automatically do color calibration faster by itself instead of defining it manually. Figure 1.18 shows how the proposed method defines a color mask and filter colors based on the color of the ball.

1.5 Corner Detection

In image processing, corner is created where two edges meet with different directions (Haggui *et al.* (2018)). Despite of edge detection, corner detection outcomes are typically a set of a few and countable points (Haggui *et al.* (2018)). If an object shape is not deformable, it could be useful if we use corners of an object to identify and separate it from background and other objects in an image. However illumination change and rotation of the target and motion blur are main reasons that can cause substantial error for detecting corners (Haggui *et al.* (2018)).

1.5.1 3D Corner Neighborhood Estimation

The efficiency and repeatability of a corner detector defines its usefulness in a real-world application. The repeatability is vital because an identical scene viewed from different locations must produce features which match to the same real-world 3D positions. Efficiency is essential because this controls if the detector combined with other processing units can operate at real-

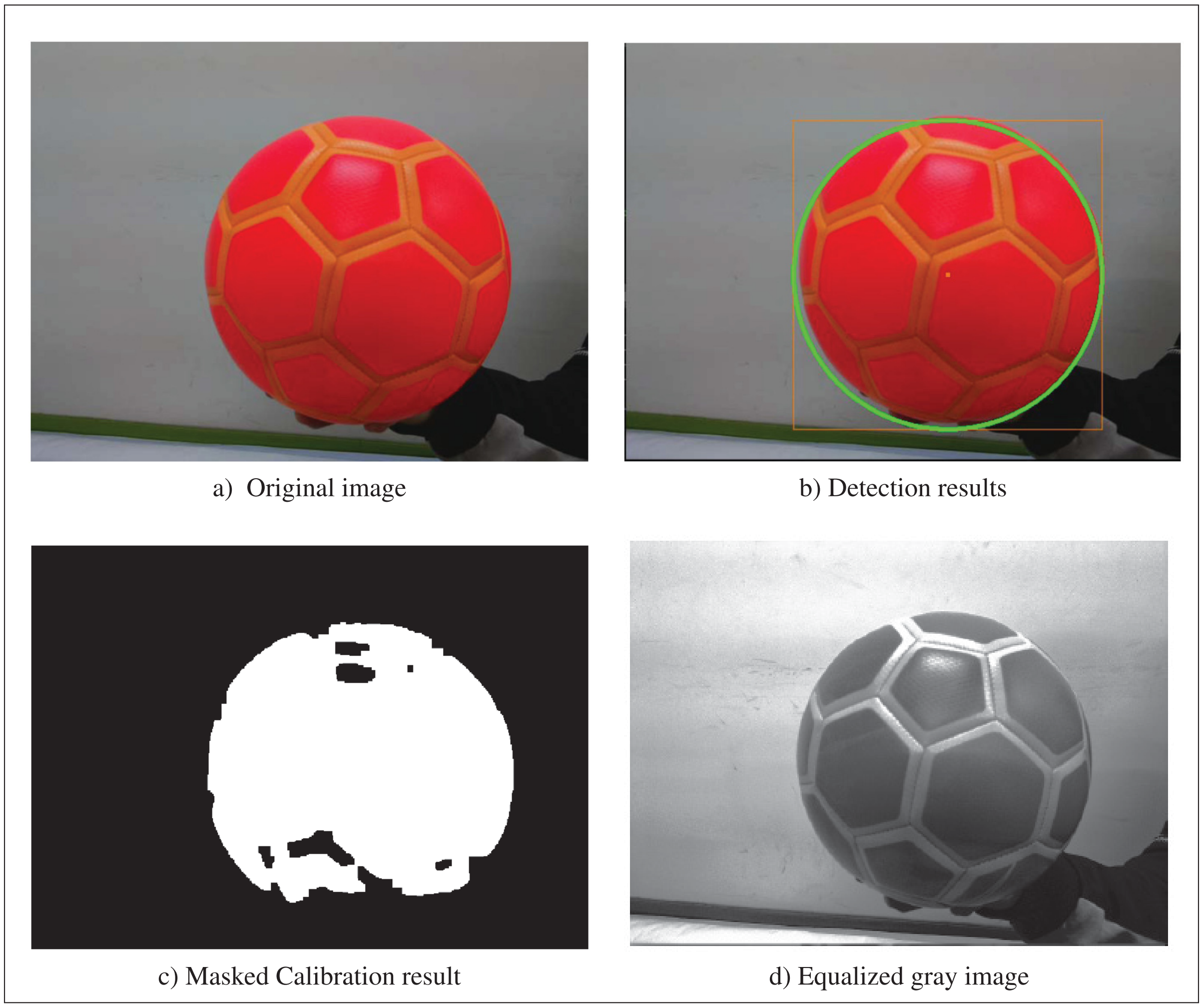


Figure 1.18 Proposed circle detection technique. Reference: Putri *et al.* (2017) pages 2,3 and 5

time 60 frame rate. First advancement described by Rosten *et al.* (2010) is a novel heuristic for feature detection, Rosten *et al.* (2010) designed a feature detector using machine learning that can process live PAL video using less than 5% of the available processing time. In comparison, most other detectors cannot even run at 60 fps frame rate (Harris detector 115% and SIFT 195%). Next advancement obtained by generalizing the detector, letting it to get adjusted for repeatability, with just slight loss of efficiency. At the end, a precise comparison of corner detectors according to the repeatability criterion is applied to three dimensional scenes. It is shown that, even with being principally made for speed, on strict experiments, the proposed accurate detector significantly performs better than current feature detectors. Finally, the comparison reveals that using machine learning produces major enhancements in repeatability, achieving a detector that is both very fast and precise. 1.19 demonstrates the process of defining a corner based on twelve-point technique.

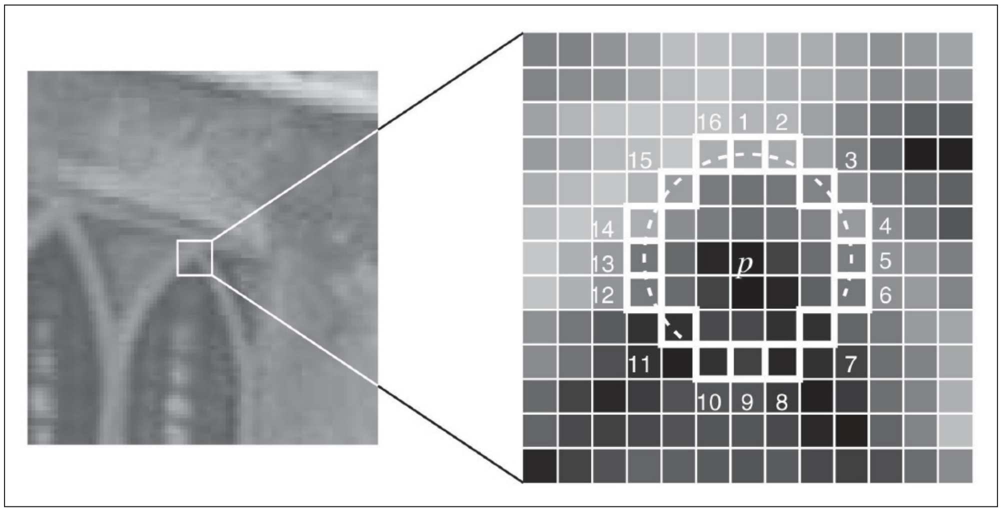


Figure 1.19 Twelve-point technique in an image patch. The cropped section on the left image is the area considered to test the twelve-point method. The square marked as p at the center of the chosen template is a candidate pixel to be at a corner of an object. The 12 adjacent pixels around the pixel p that are marked by dash border are more white than the candidate corner p more than a defined threshold. Reference: Rosten *et al.* (2010) page 5

Rosten *et al.* (2010) proposed a FAST family of detectors. By using machine learning, the very simple and repeatable segment test heuristic is modified into a FAST-9 detector that has matchless running speed. By simplifying the detector and removing predetermined ideas about how a corner should be defined, the suggested method is capable of optimizing a detector directly to advance its repeatability, producing the FAST-ER detector. Despite being very efficient, FAST-ER has significant enhancements in repeatability over FAST-9 (particularly in noisy pictures). The outcome is a detector that is not only computationally efficient, but also has improved repeatability results and is more reliable with changes in corner density than other current detectors.

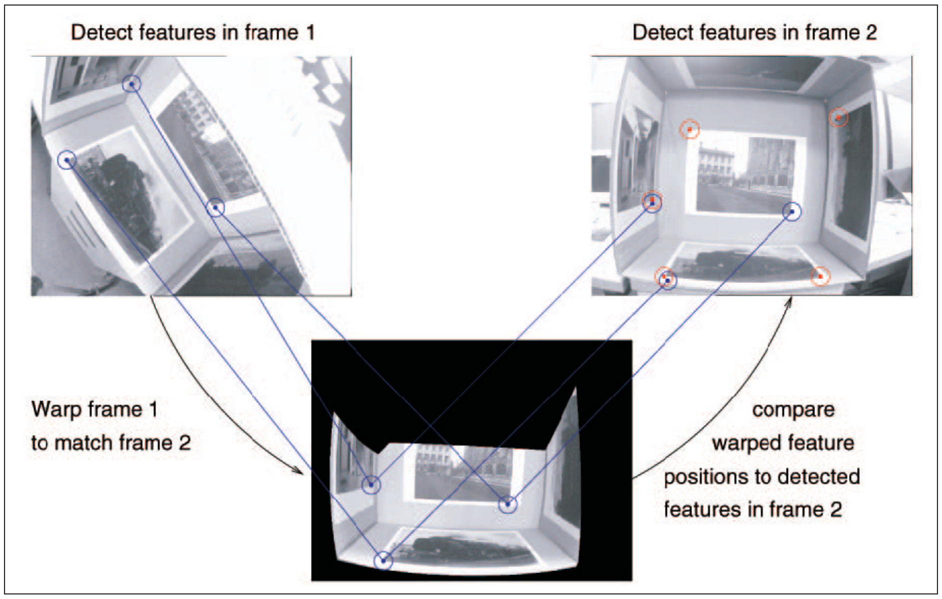
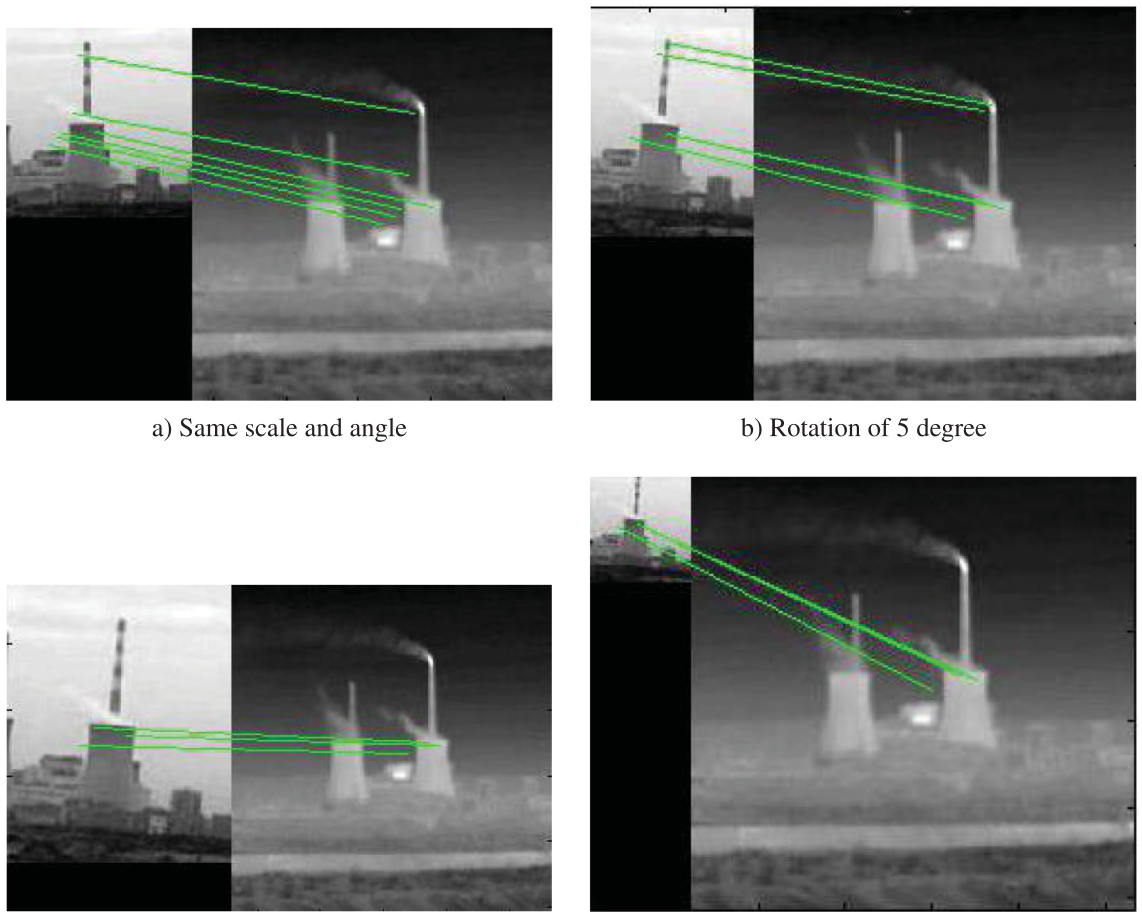

Detect features in frame 1
Detect features in frame 2
Warp frame 1 to match frame 2
compare warped feature positions to detected features in frame 2

Figure 1.20 Repeatability tested by changing the views.
Reference: Rosten *et al.* (2010) page 7

These results bring an interesting point about corner detection methods: Too much belief in perception can be misleading (Rosten *et al.* (2010)). This technique rather than focusing on how the algorithm has to do the job, concentrates on what performance measure is supposed to be optimized and it yields good results that makes it a detector that compares favorably to present detectors (Rosten *et al.* (2010)).

1.5.2 Corner Detection And Template Matching Integration

Gao & Cai (2017) proposed a novel template matching technique based on multi-scale corner point detection. This algorithm is claimed to solve image matching difficulties in infrared and visible pictures. After creating the Gaussian scale space, at first, Canny is used to find the multi-scale edges in the Gaussian space and the curvature scale space (CSS) is built, then the corner feature points are determined and the multi-scale CSS detector is molded. Secondly, to prevent the gradient flip while constructing the feature points gradient vector in both visible and infrared pictures, gradient direction angle of feature point neighborhood is narrowed. Also the direction is adjusted by using adjacent projection, the main direction of feature points is gained from histogram of the gradient direction. A 64-dimensional feature point descriptors space is created and normalized. It is verified that in experimental results, the proposed algorithm can successfully match the infrared and visible images, the matching result remains precise in the occurrence of rotation, zoom scale and illumination changes. Figure 1.21 shows the results of matching templates in different conditions.



a) Same scale and angle

b) Rotation of 5 degree

c) Magnifying 1.5 times

d) Shrinkage of 1.5 times

Figure 1.21 Template matching results in different conditions. Reference: Gao & Cai (2017) page 5

1.5.3 Corner Detection in Curved Images

Yu *et al.* (2017) proposed a new approach to reshape and reconstruct curved images captured by fisheye lenses and designed a framework to define new features for corner detection with less distortion. There are two problems that has to be solved. First, resolution is not uniformly distributed through space and then, spherical polar coordinates has non-linear and non-uniform distribution over Cartesian space. These problems are solved by making an adjustment in the Yin-Yang grid (Yu *et al.* (2017)), which is an overset grid containing two latitude/longitude coordinate systems. Effectiveness of this novel method is verified by running experiments on real and synthetic pictures.

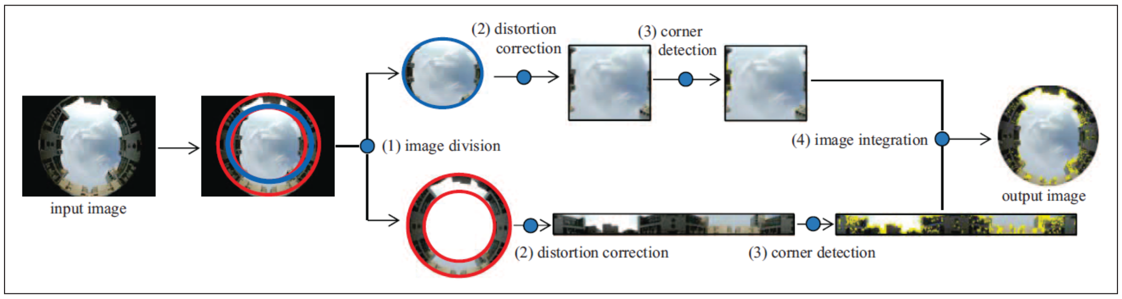
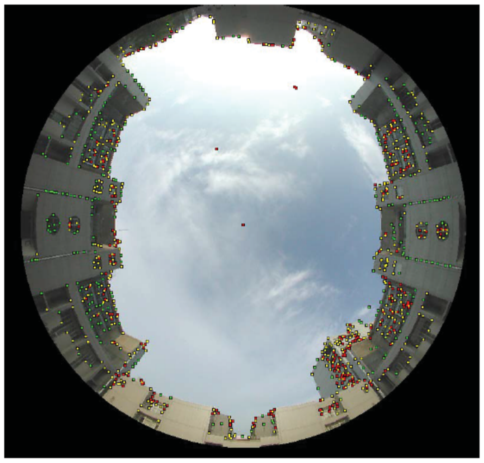


Figure 1.22 An overview of proposed method. Reference: Yu *et al.* (2017) page 2

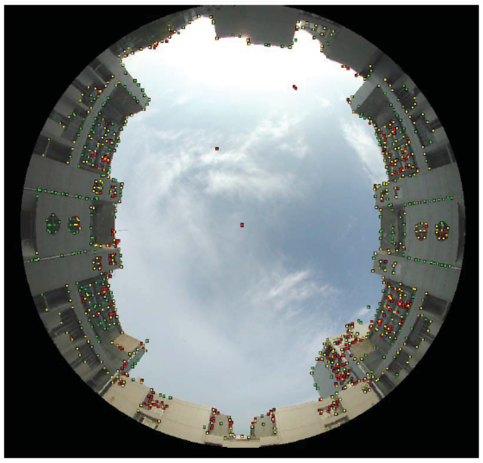
Figure 1.23 compares the effectiveness of the two discussed methods in detecting corners.

1.6 Tracking

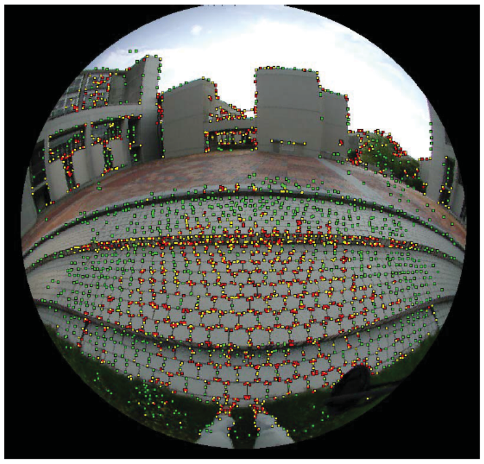
Tracking takes into account when there is motion and multiple frames. Trackers are faster compared to detection algorithms (Divakaran *et al.* (2018)). They will be less likely to lose an object and more robust to motion blur, brightness changes and occlusion. Trackers don't necessarily understand the image patches they follow. They are normally designed to find a similar image patch in different frames. Robust object tracking based on visual features is one of the most challenging subjects in computer vision.



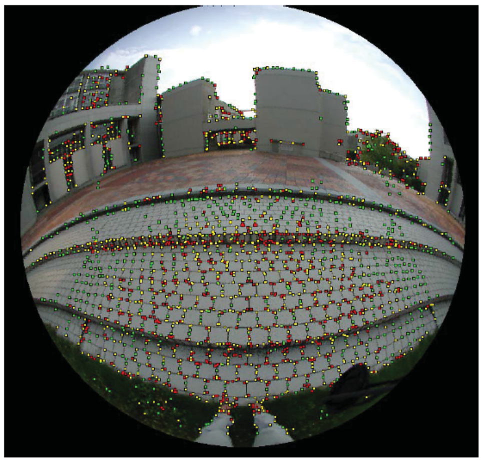
a) ShiTomasi (cyan and yellow spots) and proposed- ShiTomasi (purple and yellow spots)



b) Harris (cyan and yellow spots) and proposed- Harris (purple and yellow spots)



c) ShiTomasi (cyan and yellow spots) and proposed- ShiTomasi (purple and yellow spots)



d) Harris (cyan and yellow spots) and proposed- Harris (purple and yellow spots)

Figure 1.23 Comparing four different corner detection approaches in two scenes. Reference: Yu *et al.* (2017)

1.6.1 Part-Base Tracker

Many studies in computer science about trackers focused on model-free tracking techniques (Chrysos *et al.* (2018)). In such methods, a square boundary box around the object is given in the very first frame and then the algorithm follows the target in the rest of the video (Chrysos *et al.* (2018)). The first problem that challenges these algorithms is when the object deforms. The boundary box scenario for objects that are not always showing one face to camera is sub-optimal. Using a part-base method instead by training discriminative Deformable Part Models (DPM) is a solution (Chrysos *et al.* (2018)). The main challenge of this method is the difficulty of training discriminative DPMs with only a few number of training examples. Chrysos *et al.* (2018) suggested that a reproductive model is a better fit for the task. An Incremental Pictorial Structures (IPST) is proposed which advances pictorial structures by incremental updates to gain robustness and get adjusted for object adaptations. The results of a set of experiments verified in details the proposed IPST outperforms the state of art model-free algorithms in the applications of tracking human and animal face and body.

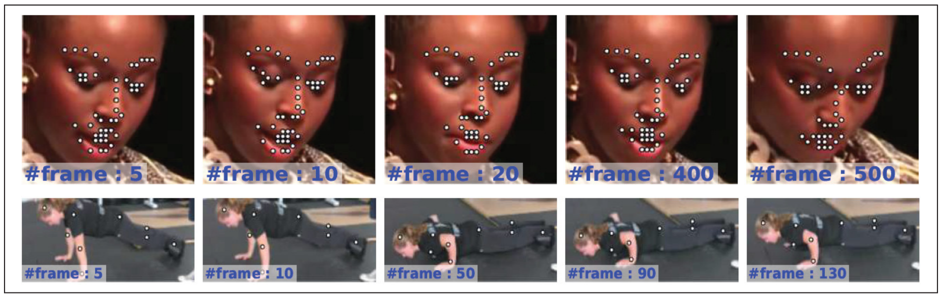
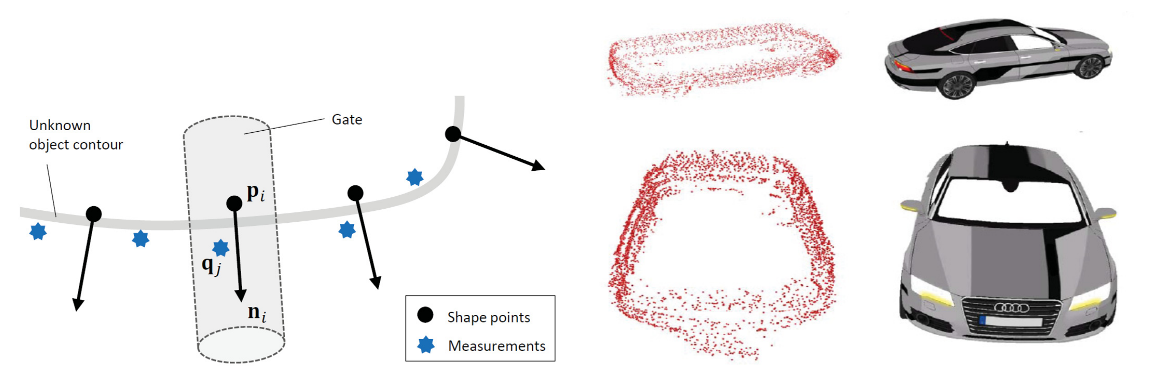


Figure 1.24 IPST proposed stable tracker. Reference: Chrysos *et al.* (2018) page 7

1.6.2 3D Cloud Model

Awareness of surrounding environment is a necessary pre-requirement for autonomous cars to operate (Kraemer *et al.* (2017)). Specifically, trustworthy information about the motion and dimension of objects is crucial for a safe and convenient driving policies (You *et al.* (2019)). The capability of precise contour measurements, for instance by laser scanners, let us estimate object geometry accurately which has advantage for the tracking in occurrence of partial occlusion and high performance processing. Kraemer *et al.* (2017) propose a batch formulation of the object tracking problem which allows to approximate object motion and form as a cloud of gathered 3D points at the same time. The suggested method is shown to be able to produce very precise motion estimation and reconstructions of object 3D shape from only three layers of scan data.



a) Shape integration using gating. b) Reconstructed 3D point cloud view.

Figure 1.25 Illustration of proposed method and results. Reference: Kraemer *et al.* (2017) page 3 and 5

1.6.3 UAVs Collision Avoidance

In recent years, interpreting images based on visual features and extracting objects receive lots of attention (Chee & Teoh (2019)). A number of algorithms doing such tasks are proposed and made collision avoidance of robots and object tracking feasible. These algorithms have to be accurate, robust to illumination and appearance changes, fast and computationally efficient. Methods based on handcrafted heuristics and constraints are widely employed for tracking UAVs (Chaudhary *et al.* (2017)). These methods limit the UAVs capabilities and are typically used for single task application (Chaudhary *et al.* (2017)). Figure 1.26 shows the process of the experiment for the proposed method.

Chaudhary *et al.* (2017) studied the challenges in tracking and successfully landed an autonomous unmanned aerial vehicle (UAV) on a fast moving object, they also presented a tracking algorithm based on visual appearance that fuses cross correlation filters with deep

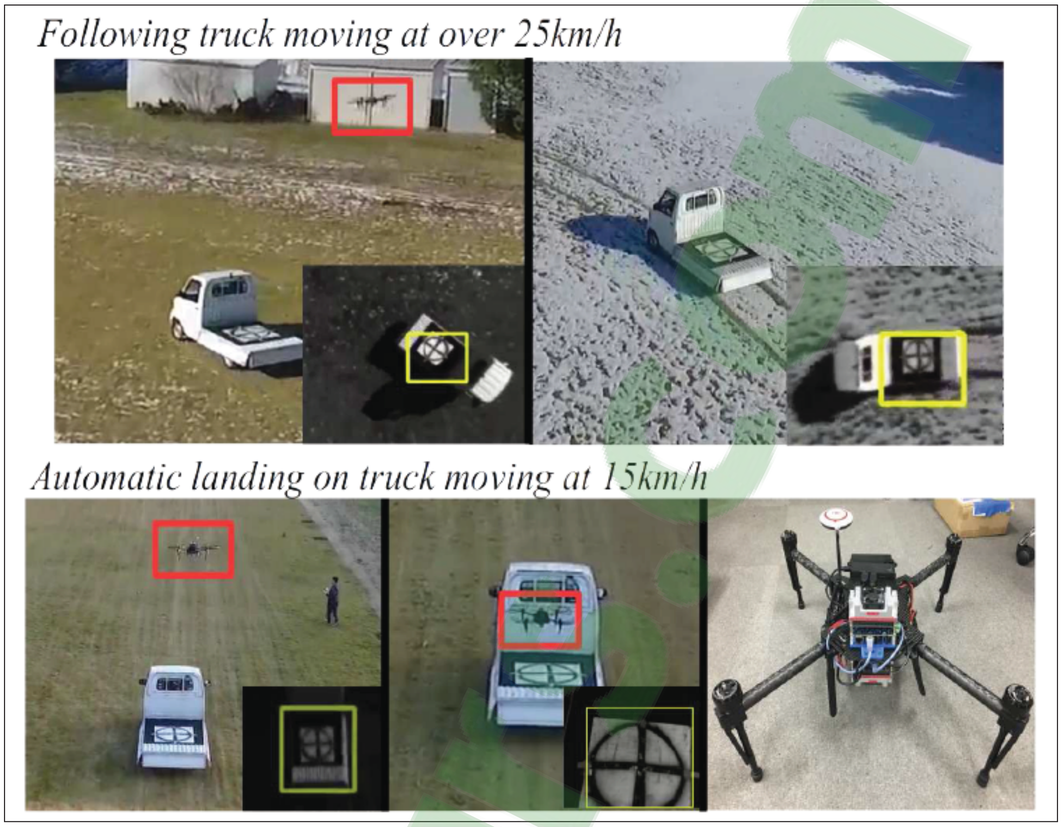

Following truck moving at over 25km/h
Automatic landing on truck moving at 15km/h

Figure 1.26 The yellow bounding box indicates the results of our tracking. Red box is the position of the flying uav. Reference: Chaudhary *et al.* (2017) page 1

comparison network for instantaneous accurate tracking. The algorithm translates each frame using an online learning model by searching the neighboring area detected from last frame. Instead of using pyramidal technique, a Deep Comparison Network (DCN) is established to predict scale variations. DCN is established to estimate scale variation and compensate tracker shifting using cross correlation filters in a single network calculation. The network is trained by end-to-end method which tries to learn an influential matching tool for localizing target using a defined template. Moreover, the algorithm can find and detect a lost object due to total occlusion without requiring a separate detector. Figure 1.27 shows the procedure how to recover a lost object using a predefined template.

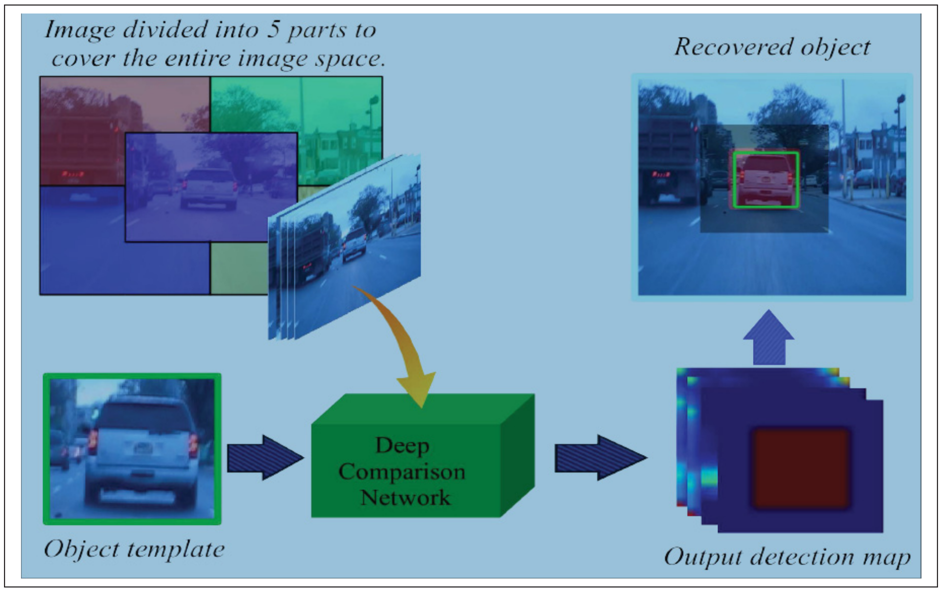


Figure 1.27 Structure of the fast detection pipeline proposed to recover lost targets using a predefined template. Reference: Chaudhary *et al.* (2017) page 5

1.6.4 Kernelized Correlation Filters

The core component of nearly all modern trackers is a discriminative classifier, created to discriminate between the main object and the neighboring environment (Henriques *et al.* (2015a)). To deal with natural image variations, this classifier is characteristically trained by translated and scaled training model template images (Henriques *et al.* (2015a)). Redundancies influence these samples and it needs to be optimized by an adjustment: any repeated or overlapped pixels are considered to be identical. According to this simple statement Henriques *et al.* (2015a) suggested an analytic prototypical for thousands of translated template patches datasets. By presenting that the output data matrix is circulant, it could be diagonalized with the Discrete Fourier Transform, decreasing both storage and computational cost by a number of orders of magnitude. Interestingly, this formulation is correspondent to a correlation filter for linear regression, used by some of the fastest competitive algorithms (Henriques *et al.* (2015a)). For kernel regression, though, a new kernelized correlation filter (KCF) is proposed, that unlike other kernel trackers, it has the exact equal complexity as its linear counterpart. Additionally, Henriques *et al.* (2015a) suggested an extremely fast multi-channel extension of linear correlation filters as well, through a linear kernelized filter, that is called dual correlation filter (DCF). Both KCF and DCF outperform top-ranking tracking methods such as Struck or TLD on more than 50 videos benchmark (Henriques *et al.* (2015a)), except for running at hundreds of frames per second (FPS) and being created in only a few lines of code.

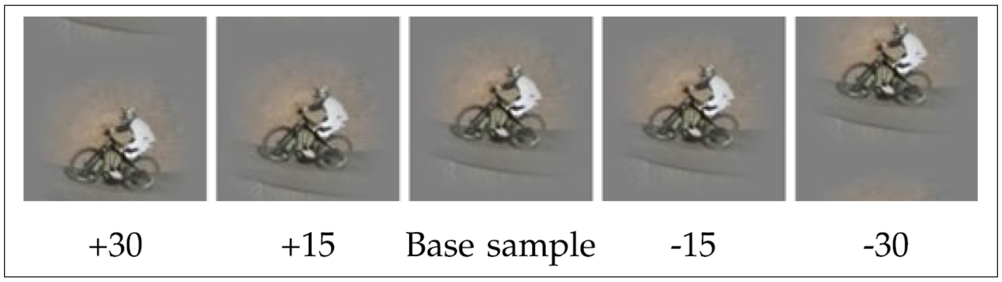


Figure 1.28 Vertical cyclic shifts in a base image patch. The formulated Fourier domain theory let us train a smart tracker with arbitrary cyclic shift of the template image, both vertical and horizontal. Reference: Henriques *et al.* (2015a) page 4

Figure 1.28 shows vertical shifts in the base sample template.

1.6.5 Extended KCF

Robust object tracking based on visual features is one of the most challenging subjects in computer vision (Lu *et al.* (2017)). Correlation filters showed a great robustness against circular shifts and therefore receive lots of attention (Lu *et al.* (2017)). What makes most of the correlation filter-based trackers unreliable in the task of tracking is that they fix the scale of the object in every single frame and use only one template patch to update the filters. Lu *et al.* (2017) claimed to improve the robustness of the kernelized correlation filters (KCF) for the task of tracking, by establishing a fast scale pyramid approach to get to a better solution for the scale fluctuation problems. Moreover, a sparse model selection system on template sets is presented to solve the problem of noisy and unclean templates in single template techniques. Suggested method is tested on OTB-2013 dataset and the experimental results verifies the robustness of this novel approach. The proposed tracker reaches assured performance in both accuracy and computational cost and speed comparing with the common state-of-the-art trackers.

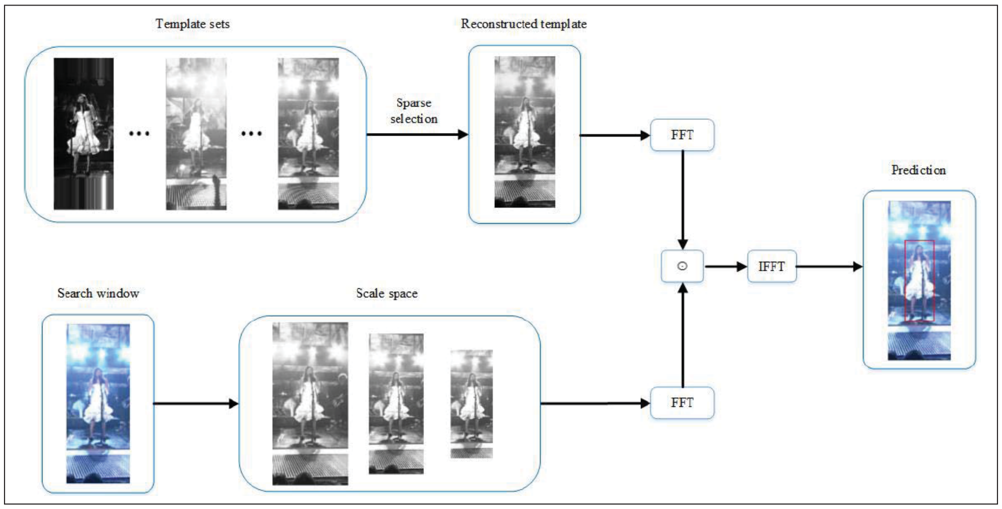


Figure 1.29 Template reconstruction and its convolution with the search area and the approach to find the position of the maximum response (similarity) is illustrated. Reference: Lu *et al.* (2017) page 3

Figure 1.29 demonstrates the process the Extended KCF performs to match the best prediction in the search area to the template.

1.7 Kalman Filter and Extended Kalman Filter

Kalman filter is a linear advance tool that predicts the next state of a system based on previous predictions and measurements assuming the position of the target is determined in a sequence of previous frames. The first thing that is concluded from definition above is that kalman filter prediction gets more accurate as it gets more observations during time (El-Ghoboushi *et al.* (2018)). We use kalman filter to reduce detection noise and fluctuations and to predict the next state of the target to measure the reliability of the algorithm when in the verification chapter.

1.7.1 Kalman filter for tracking prediction and denoising

At the present time, numerous researches are focused on the process of tracking aircraft and air traffic surveillance systems (El-Ghoboushi *et al.* (2018)). Multilateration air traffic surveillance system is known as one of the hot spot zones for scientists. El-Ghoboushi *et al.* (2018) proposed one Non-Bayesian method for single, and an extension for multiple aircrafts tracking that considers the appearance of clutter. As you can see in figure 1.30, the suggested approach deals with latitude and longitude (geographical coordinates).

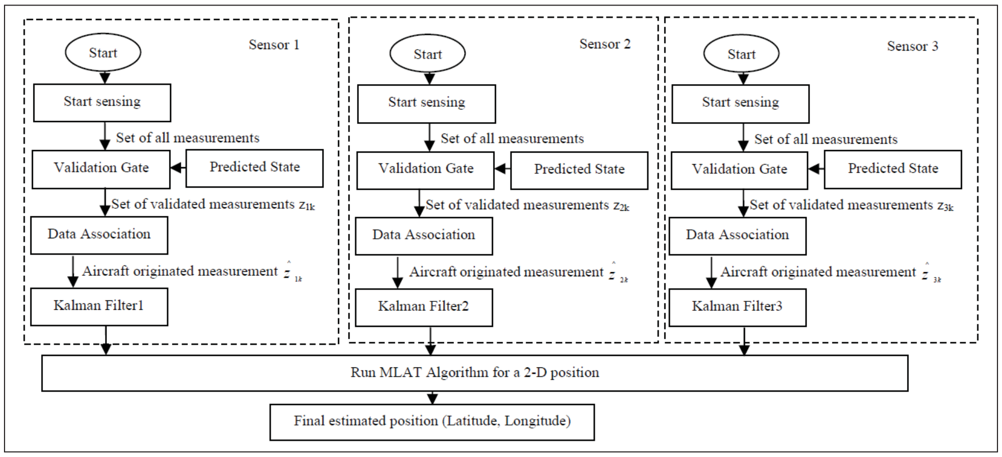
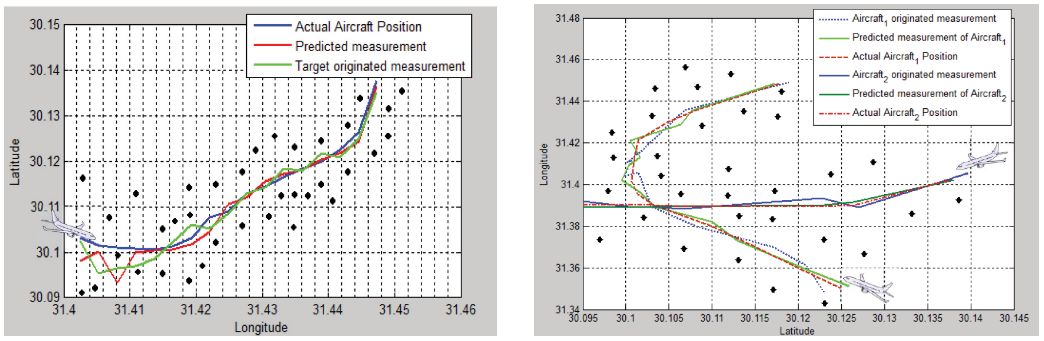

Sensor 1
Start
Start sensing
Set of all measurements
Validation Gate
Predicted State
Set of validated measurements z_{1k}
Data Association
Aircraft originated measurement $\hat{z}_{1k}$
Kalman Filter1
Sensor 2
Start
Start sensing
Set of all measurements
Validation Gate
Predicted State
Set of validated measurements z_{2k}
Data Association
Aircraft originated measurement $\hat{z}_{2k}$
Kalman Filter2
Sensor 3
Start
Start sensing
Set of all measurements
Validation Gate
Predicted State
Set of validated measurements z_{3k}
Data Association
Aircraft originated measurement $\hat{z}_{3k}$
Kalman Filter3
Run MLAT Algorithm for a 2-D position
Final estimated position (Latitude, Longitude)

Figure 1.30 The structure for a single aircraft tracking in clutter. Reference: El-Ghoboushi *et al.* (2018) page 3

First, a suggested single aircraft tracking technique in clutter is reviewed. A modified Kalman filter in the predicted state is used. The proposed predicted state of aircraft is presented which depends on the state in the last three iteration spots. Monitored by a suggested validation gate and data association analysis. The used Validation gate threshold can change and adapt and it is not necessarily constant. Then a novel multi-aircrafts tracking modeled in clutter is proposed and analyzed. In conclusion, a validation technique is designed with simulation of both scenarios by using a Multilateration network at Cairo International Airport by considering a pilot area. These results show how kalman filter is useful in denoising a signal, prediction of future estimated position of the target and lowering the effect of failing sensor.



a) Single aircraft is tracked in clutter b) Two aircrafts are tracked in clutter

Figure 1.31 Reference: El-Ghoboushi *et al.* (2018) pages 5 and 6.

1.8 Conclusion

Template matching is the basic element of most detection and tracking methods. The approach of finding the best matched template is different in different techniques. One fast and reliable discussed approach is KCF know for its accuracy of 2D template detection, reliability during partial occlusion and speed, but it lacks lightning endurance and detecting changes in the size of the object. Template matching also gives us clues how corner detection and edge detection work. During this study template matching proved that there is a relationship between the size of the template and the size of the target that is important to match performance and accuracy of detection methods and tracking methods.

Color detection is only useful if we use color markers. Hopefully this is a variable under our control. Moreover, some colors would be found easier than others. In the next chapter we discuss how to use the HSV converted image to distinguish different colors.

Circle detection methods are vary such as GHCT built in Python and C++ and less accurate low performance "imfindcircle" in MATLAB. Obviously, we would choose both accuracy and performance if we could. Thus we do this research with Python for its simple syntax, high performance and accuracy over built in and/or written algorithms by the author.

Tracking techniques had an evolutionary path from beginning path throw decades of researches, from point trackers to KCF. We will modify KCF to get the best result based on our own target's shape and visual features. Finally, we would like to filter the final results for the reason of:

- Smooth the path of the target during multiple frames,
- Predict the presence of the target in the coming frame to avoid losing the object,
- Guess the position of the object if the main algorithm fails to find id due to total or partial occlusion.

In the next chapter we propose a method to use and modify discussed techniques and develop other techniques to enhance the accuracy and performance of tracking and detection.

CHAPTER 2

TRACKING AND DETECTION OF A SPHERE MARKER

2.1 Introduction

The human brain performs tasks in ways it cannot express. We do not know how we distinguish between objects, understand the pattern of visual change and finally track objects while looking at them. Computers do not work in the same way. They don't understand a picture as a human does. What a machine sees in a picture is a 2D or 3D matrix of numbers expressing the color and brightness of pixels[1]. In order to let computers detect and track objects in video sequences, we have to do calculations on discrete pixels and find mathematical differences between an object and the surrounding environment.

In this work, detection and tracking of sphere markers are discussed. Sphere markers are widely used because their shape does not change when they rotate or move. By following sphere markers attached to any object, we can track the position of that object regardless of its shape and color. Before digging into mathematics and algorithms we need to define tracking and detection.

Detection: Refers to finding a specific object based on its visual (mathematical) properties in an image. The image could contain many other objects. Detection of an object in a video (sequence of frames) might happen at each frame[2] regardless of appearance of the object in other frames.

[1] Pixel is smallest unit of a raster/digitized photo

[2] A video is made by a sequence of frames. Each frame is an image captured at a certain time

Tracking: Tracking in a video is the process of following a selected part of a frame in the subsequent frames. Trackers do not necessarily understand what the selected part is. It could be any object or no object. They capture the selected part, keep it in mind and try to find similar patches of image in the following frame(s).
The reason why we run both tracking and detection is because tracking algorithms are faster in finding the region of interest (ROI) and unlikely to lose an object while detection algorithms are more accurate and able to distinguish objects from each other in relatively small image patches. So we find an area containing the desired object by tracking and then search for exact object in that area by running a detection algorithm. We run a "Kalman Filter" algorithm at the end to reduce the noise and smooth the motion path.

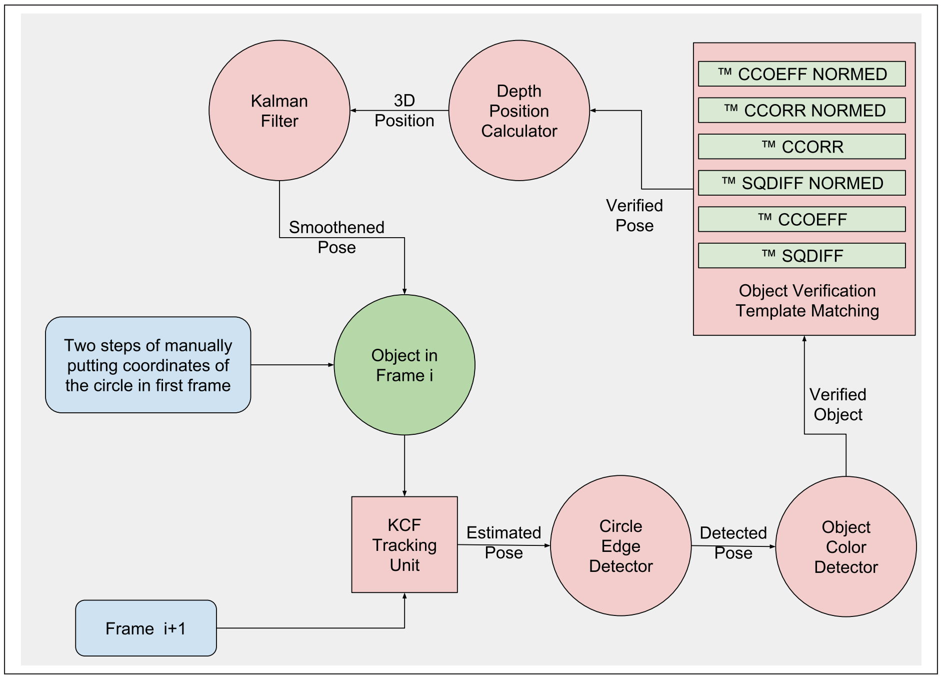


Figure 2.1 Schematic view of algorithm

Figure 2.1 shows an overview of the methodology for this task. It gives a schematic understanding how the algorithm is designed and how the method manages to reach the desired results. It starts with the first frame in a video sequence and given coordinates of target on it. Tracker is trained by cropping a template containing the marker. Since the marker is the main part that needs to be tracked, template should be small enough in order not to be considered as background. On the other hand, fast motion could cause the marker not to be detected as a circle if the target gets out of the template image partially. Thus the template size must have a minimum amount. Next, an edge circle-detector searches the template patch for a circle shape object. The algorithm to do such task is optimized to minimize the error caused by detection of possible displaced circles. A color based object detection is also designed to double check the detection as an independent process and verifying the presence of the object. It defines a range of colors as the signature of the object and minimize it to keep it exclusive to marker. Once the position and size of the marker is detected, it should be converted to 3D location of the marker with respect to the camera. By using the characteristics such as focal length, field of view and sensor size that is described for “GoPro Hero 4 Silver" by its manufacturer, 3D position is obtained. A Kalman Filter algorithm that is a common technique to reduce measurement noise the detected path way is designed and made compatible with three-dimensional motion space. At the end of the methodology a validation technique is designed and used to verify the validation of final results.

2.2 Objective

This project is defined to develop a computationally inexpensive method to find the pixel-wise position of the joints of human body using sphere markers. It could be used later in other projects such as: 3D reconstruction, 3D virtual reality, physiological purposes and etc. The suggested method proposes to use spherical markers to make the task of detection achievable. If markers are not used, most other complex techniques including neural network algorithms to date seem to lose accuracy over time. They are also significantly more expensive computationally. The most recent algorithm presented in 2018 called "Masked YOLO[3]" Wong *et al.* (2018) developed by "Darknet" laboratory which uses RCNN[4] with mask around the tracked object

[3] You Only Look Once

[4] Recursive Convolutional Neural Networks

has low accuracy for relatively small objects like markers (with only a few pixels width [5]) or joints far from camera and processing time as high as tens of seconds per frame on the same machine which can process almost hundred frames per second with the developed algorithm in this study. However this method is limited to spherical markers which should have known colors.

Since almost all developed methods either fail to adjust and follow changes in size of the marker or fail to fully recover the detected size of a marker circling periodically, an improvement is needed in order not to lose much of accuracy over time. We will also compare the accuracy of proposed method and three advanced and recent algorithms to challenge robustness of all methods.

The result of this study will be ground truth for studies on racing wheelchair. Hence, we use “GoPro" camera capturing videos with 240 frames per second and has the resolution of 720×1280 pixels.

We make four valid assumptions to make reaching the goals of this study achievable:

1. We choose different colors for markers that could pass near each other to avoid miss-detection.
2. In order to reduce the chance of losing the target, markers are recommended to have colors that are not available in the surrounding environment.
3. One of the most challenging issues with tracking and detection algorithms is their incapability in keeping accuracy for relatively small objects. Hence, the size of the markers should be at the largest amount that does not add any limitation to movement of body parts.
4. To reduce the effect of motion blur, it is more proper to capture video frames by high frequency. Today, “GoPro" cameras are capable of recording videos at 240 fps which is suitable for this study.

[5] These methods are known for their high accuracy in classifying the type of the target but not its location

With these assumptions we control the factors that can affect the accuracy or cause miss-detection or even losing the targets. But not all elements could be controllable. Such as: occlusion, illumination changes, vibration, reflection, and so on. Each of these challenges could cause multiple issues. For example occlusion could make changes in detected target size, could cause the algorithm fail to recognize the true spherical shape of the marker or even could cause losing the object entirely. The goal of this study is not to propose a method that is absolutely robust against all possible challenges. We will present a method achievable in a master study that brings accuracy to a higher level compare to state-of-art methods for some challenges such as partial occlusion and brightness changes. Other challenges need more advanced equipments and more research investigations and are out of scope of a master study. This method should suggest a mechanism to pause the proposed algorithm and ask for the true position again if it senses that it's losing the primary object. Losing the target could happen because of four main reasons:

1. In case occlusion, the target might not be visible for a number of frames.
2. In case of passing by an object with similar shape, the algorithm may start pursuing a wrong object by mistake.
3. In case of passing by an object with similar color, the algorithm might continue but following a different object by mistake,
4. Accumulated error over time causes the algorithm losing the position of the marker.

Once the marker is spotted at the end of task of detection and tracking, there could be a small fluctuations in the set of true detected positions. It is called overfitting[6] and it would happen because we record videos in a discrete time frame. Reaching the position of the target is not achievable in all infinite moments of time-line. In order to overcome overfitting, we developed and modified a “Kalman Filter" to make the path of passing target smoother. It would sacrifice

[6] Because detection gives the best found position in a single frame (training instance) regardless of what are the positions of the target in other frames (test instances)

a bit of accuracy on training samples. However “Kalman Filter" is optional and could be used in case of requiring a smooth movement of the marker.
There are many programming languages that could support most functionality of the algorithm we present. C++, python, java, c# and others. We would try to implement everything by python for three main reasons.

- Python has relatively simple syntax,
- It is open-source and free. Hence, millions of researchers worked with it and discussed any possible issue,
- It is well developed and optimized and has high performance,
- There are plenty of similar works and optimized algorithms available for free implemented by Python.

Python is both functional and object oriented programming (OOP) that suits our needs and capable of efficient programming by reducing the number of lines of codes. The main algorithm is written in about 1000 lines. But all other algorithms that we wrote to test, compare and validate the results are written in more than 30,000 lines of codes.

2.3 Tracking

2.3.1 Introduction

The algorithm starts by asking for inputs on the very first frame. The position and size of the marker are the basic inputs and should be given initially. Then the tracker unit starts to learn an image patch containing the given input area. This template will be searched on the next frame in neighborhood areas around the location it is on current frame. Tracker will search for the template itself and all its shifted made samples. Except for the first frame, the first step the algorithm takes when a new frame is given, is to track the image patch taken from previous frame and send the position to detector unit. In this section we will discuss how tracker does such a task.

2.3.2 Learning Tracker

Tracking objects based on visual features is a challenging and critical task in the field of Computer Vision. There are many applications for it, including Video Surveillance, 3D Reconstruction, Human Computer Interaction, Autonomous Driving and Navigation Systems. The process of tracking a single object is to use the position of a particular object in the first frame and estimate the states of the object without manually selecting the object in the next frames. There are many factors that make tracking an object challenging such as motion blur, partial occlusions, object or camera rotations, moving camera, object position changes, camera vibration, complex background, brightness variations, scale variations, etc. In recent years, numerous tracking methods have been suggested to tackle these challenges, and a number of approaches have been developed to increase the accuracy and performance of tracking, but there is no single method yet that can solve all the challenges in real-time.

There are two general classes of trackers, Generative and Discriminative. The goal of Generative methods is to search in the current frame for the most similar region to the template built in the last frame. What makes Generative methods unreliable is changes in the template appearance. If shape, color or illumination of the target changes, it rotates or camera goes around the object or motion blur happens, the similarity factor would be reduced drastically. Hence, Generative trackers have lower efficiency because they are not robust enough to changes in the object and template appearance.

Discriminative trackers try to learn the template and search area in the last frame to predict the object state in the following frame. In recent years, Correlation Filters (CF) are attracting researchers due to the outstanding results. Correlation filter runs correlation operation and finds the maximum response. But they usually require a large number of instances in their training set which makes the computational cost incredibly high. Kernelized Correlation Filter

(KCF) successfully increases the Circulant Structure Kernel (CSK) tracker performance and efficiency by using Histogram of Oriented Gradient and Multichannel Methods.

2.3.2.1 Kernel Filters

A brief overview of Linear Regression is presented, before we start a discussion about Non-Linear Regression and Kernel Trick. Linear Regression known also as Ridge Regression provides a closed-form solution and makes it simpler than Support Vector Machines (SVM) with comparable performance (Lu *et al.* (2017)).

Linear Regression

The goal in ridge regression is to find an optimized function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ that minimizes the sum of squares error over training examples x_i and the target y_i where λ is the L2 regularization parameter.

$$W_{opt} = \underset{w}{\operatorname{argmin}} \sum_{i}^{n} (f(x_i) - y_i)^2 + \lambda \sum_{j}^{m} w_i^2 \tag{2.1}$$

Where in (2.1) "n" is the number of instances, "m" is the number of features or dimension of the function f. The closed-form solution for the equation (2.1) is:

$$W_{opt} = (X^T X + \lambda I)^{-1} X^T Y \tag{2.2}$$

Later in this section we will work with the Fourier domain. Hence, the closed-form solution for complex values will be needed. Let assume $X^H = (X^*)^T$ is the Hermitian transpose of matrix X. Thus the complex solution is:

$$W_{opt} = (X^H X + \lambda I)^{-1} X^H Y \tag{2.3}$$

Cyclic Shift Let assume we have a one-dimensional vector $m \times 1$ presented as $\mathbf{x} = [x_1, x_2, \cdots, x_m]^T$. The i'th shift of $\mathbf{x}$ can be created as shown in equation (2.4).

$$P^i\mathbf{x} = [x_{m-i+1}, \cdots, x_m, x_1, \cdots, x_{m-i}] \tag{2.4}$$

Where P^i is the permutation matrix. P^1 is presented below.

$$P^1 = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{2.5}$$

Circulant Matrix:

Circulant matrix is generated by shifting the vector $\mathbf{x}$ over all its elements until we reach $\mathbf{x}$ again.

$$X = C(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix} \tag{2.6}$$

Training matrix X is diagonal and can be written as

$$X = F \operatorname{diag}(\hat{\mathbf{x}}) F^H \tag{2.7}$$

Where F is constant and independent matrix and $\hat{\mathbf{x}}$ is discrete Fourier transform (DFT) of vector $\mathbf{x}$. Matrix F is unique and defined as a constant that computes the discrete Fourier transform of any arbitrary vector $\mathbf{s}$ as $\mathscr{F}(\mathbf{s}) = \sqrt{n}F\mathbf{s}$. It's a valid claim because DFT is linear.

To reconstruct the equation (2.3) in Fourier domain we first need to find the $X^H X$, it is calculated as:

$$X^H X = F\text{diag}(\hat{\mathbf{x}}^* \odot \mathbf{x}^*)F^H \tag{2.8}$$

Where $F^H F = I$ and $\odot$ denotes inner (element-wise) product. By putting all together we find the linear regression weights as:

$$\hat{\mathbf{w}} = \frac{\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}}}{\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda} \tag{2.9}$$

Comparing two equations (2.3) and (2.9)[7] shows instead of computing the inverse of matrix in order of $O(n^3)$ we only need to compute DFT in $O(n \log n)$ and inner product and division in $O(n)$ which reduces computational cost and time by several orders of magnitude.

2.3.2.2 Kernel Trick

Kernel trick is a method is used for multi dimensional feature space where the number of features are considerable. Let's assume $\phi(\mathbf{x})$ is a non-linear feature space with input vector $\mathbf{x}$. Then the solution $\mathbf{w}$ is a linear function of new defined instances:

$$\mathbf{w} = \sum_i \alpha_i \phi(\mathbf{x}_i) \tag{2.10}$$

The inner product of two non-linearized samples $\mathbf{x}_i$ and $\mathbf{x}_j$ is the correspondence element in kernel matrix K:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \odot \phi(\mathbf{x}_j) \tag{2.11}$$

Each element of matrix K is in fact correlation between two instances. Hence, the kernel matrix is a correlation matrix.

[7] This equation is disscussed in Henriques *et al.* (2015a)

In multi-dimensional spaces, it's recommended to use kernel trick since there is no need to deal with features no matter how many dimension they are. The new optimization solution for dual space is α which is giving weight to instances. This is the weakness of this method where computational cost increases as the number of instances grow. α could be obtained as:

$$\alpha = (K + \lambda I)^{-1}\mathbf{y} \tag{2.12}$$

Henriques *et al.* (2015a) proved that for any permutation matrix M the kernel matrix K is circulant if $k(\mathbf{x}_i, \mathbf{x}_j) = k(M\mathbf{x}_i, M\mathbf{x}_j)$. Let's call the first row of matrix K as $\mathbf{k}^{xx}$ then the matrix K is:

$$K = C(\mathbf{k}^{xx}) \tag{2.13}$$

The α solution for the optimization function $f(\mathbf{z})$ is:

$$f(\mathbf{z}) = \mathbf{w}^T\mathbf{z} = \sum_{i=1}^{n} = \alpha_i k(\mathbf{z}, \mathbf{x}_i) \tag{2.14}$$

$$\hat{\alpha} = \frac{\hat{y}}{\mathbf{k}^{\hat{\mathbf{x}}\mathbf{x}} + \lambda} \tag{2.15}$$

To detect the template containing the object, $f(\mathbf{z})$ needs to be computed for many candidate template patches for each frame. Revaud *et al.* (2013) and Kiani Galoogahi *et al.* (2013) showed we benefit from one more advantage when we work in dual space.

Linear and Non-linear Kernels

In dual space we are capable of sum over multiple channels in the Fourier domain. We need to consider scale changes and orientation variation for every frame and each candidate. Thus dual solution is providing powerful tool when we are using HOG. $\mathbf{k}^{\mathbf{x}\mathbf{x}'}$ for linear kernel and multiple channel is computed as:

$$\mathbf{k}^{\mathbf{x}\mathbf{x}'} = \mathscr{F}^{-1}\big(\sum_c \hat{\mathbf{x}}_c^* \odot \hat{\mathbf{x}'}_c^*\big) \tag{2.16}$$

One useful example of non-linear multi-channel high dimensional feature space is Gaussian kernel integrated with Radial Basis Function (RBF). Once the $\mathbf{k}^{\mathbf{x}\mathbf{x}'}$ is computed in the non-linear equation (2.17), $alpha$ could be obtained in the linear method described before.

$$\mathbf{k}^{\mathbf{x}\mathbf{x}'} = \exp\Big(-\frac{1}{\sigma^2}\big(||\mathbf{x}||^2 + ||\mathbf{x}'||^2 - 2\mathscr{F}^{-1}\big(\sum_c \hat{\mathbf{x}}_c^* \odot \hat{\mathbf{x}'}_c^*\big)\big)\Big) \tag{2.17}$$

Figure 2.2 shows how multi-channel tracking using HOG helps gaining robustness against partial occlusion.

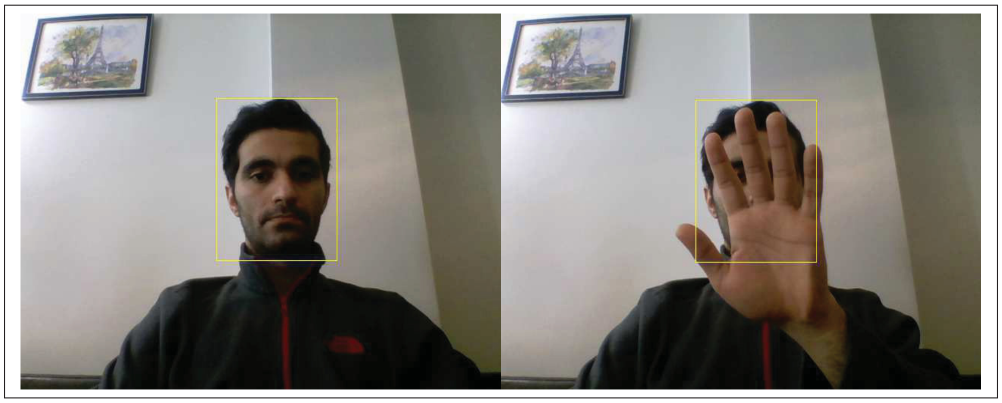


Figure 2.2 Partial Occlusion While Tracking

Figure 2.3 shows that using learning method aimed by multi-channel solution, object deformation causes no error during tracking.

It is proven that KCF tracker is robust against illumination variation (Henriques *et al.* (2015a)), object deformation, motion blur and partial occlusion. One drawback of this advance learning method is its low sensitivity to changes in marker size. On the other hand, KCF tracker would only detect changes in target size if this changes are more than a significantly large threshold.

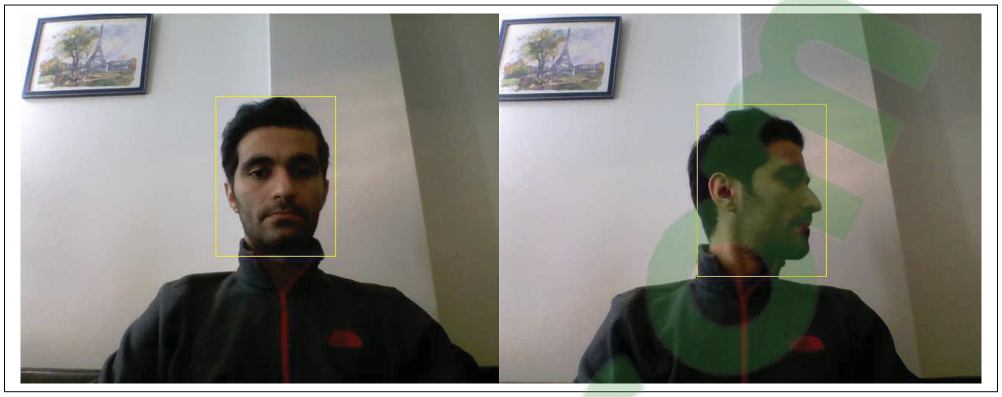


Figure 2.3 Deformation of Primary Object During Tracking Time

2.3.3 Point Tracker

KCF tracker follows the position of the target with good enough accuracy (it will be shown) in two-dimensional space. It is confirmed that KCF has an edge over state-of-art modern trackers in terms of accuracy and performance. It is claimed that it can perform tracking task with hundreds of frames per second. But its not robust to scale variation of the target.

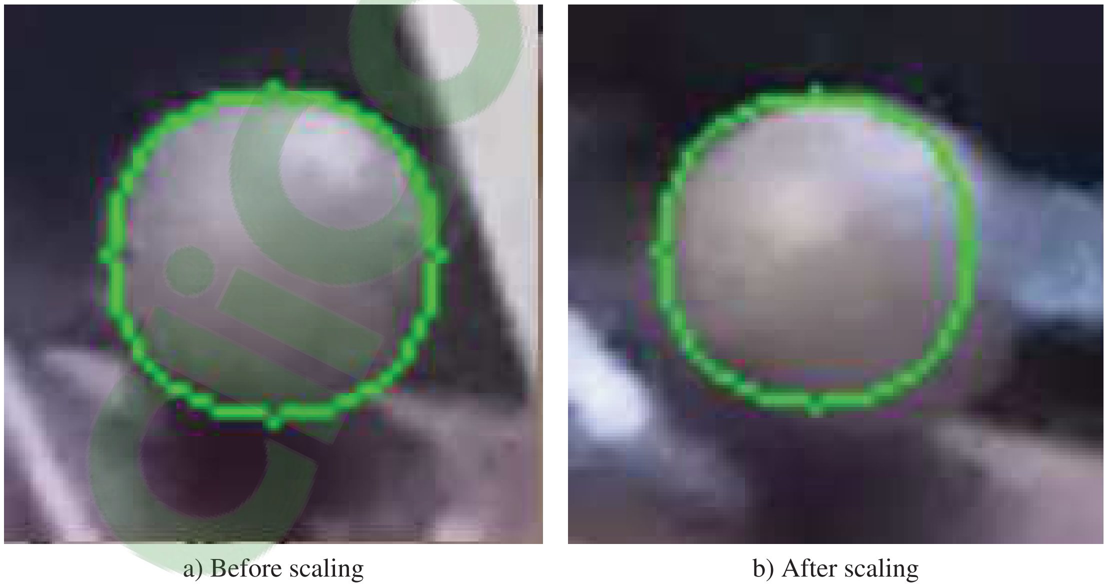


Figure 2.4 The effect of scale variation on KCF tracker

Figure 2.4 shows how changes in target size is not followed by KCF tracker. Figure 2.4a is captured before any variation change and Figure 2.4b is captured after the target got closer to camera. This figure proves that KCF is unable to adjust to the size of the market rapidly and hence makes the estimation of the third dimension with an unacceptable error.
To make the tracker more sensitive to scale variation, a method Optical Flow Point Tracking is integrated with KCF. We developed a method only to adjust the scale since the position in 2D is already determined by KCF. In this technique the Lucas-Kanade method is used to define a number of feature points on the marker. The algorithm searchs for the same features in the next frame. The changes in geometric location between feature points is calculated between every two adjacent frames in the video. To do such task, a function called "GeometryScale" is developed. Lets assume n points are defined on the marker on the 2D position (x,y). The middle point is defined as $P_m = (x_m, y_m) = (\frac{\sum_i x_i}{n}, \frac{\sum_i y_i}{n})$. Thus the radius of the marker on the second frame could be obtained as follow:

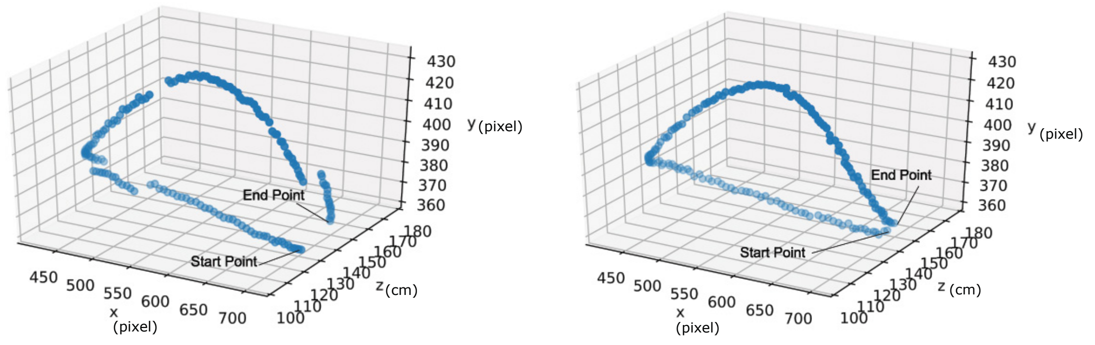
$$r_{as} = r_{bs}\frac{\sum_i \sqrt{(x_{i,as} - x_{m,as})^2 + (y_{i,as} - y_{m,as})^2}}{\sum_i \sqrt{(x_{i,bs} - x_{m,bs})^2 + (y_{i,bs} - y_{m,bs})^2}} \tag{2.18}$$

where r_{as} and r_{bs} are radius of the marker after and before scaling respectively.

2.3.3.1 Results

Applying this function will adjust scale changes and makes it extremely sensitive to changes at marker size. Figure 2.5 shows the 3D position of the marker in a sequence of frames without and with using point trackers.

Figure 2.5a is stating the fact that KCF tracker is not sensitive to small changes in marker size. Hence it jumps to another size when the changes are more than a large threshold. Figure 2.5b shows the result after using the point tracker to adjust the size the marker on every frame.



a) KCF b) KCF with point tracker

Figure 2.5 The modification on scale variation

2.4 Detection

The process of Object Detection used in this thesis includes multiple layers of Maximum Likelihood Edge Detection, Color Detection and Template Matching Verification. In order to evaluate the detection on each video, while running the algorithm arbitrary frames are selected and an error function is applied to them. In this section we try to detect the object in tracked image patch based on object main inherent features such as color and shape. Thus different template matching techniques will be used to evaluate the similarity of the imaged patch that claims to contain the object.

2.4.1 Circle Hough Gradient

Circle Hough Transform (CHT) is a known algorithm to detect circles. It is a feature extraction technique that produces circle candidates in Hough parameter space and select the pixels with local maximum "votes". Harakannanavar *et al.* (2018) reviews the common methods used for CHT and the experiments for the recognition of individuals based on their IRIS patterns. The purpose of this section is not to develop or modify this well designed method, but to optimize its input parameters. CHT function accepts a number of inputs. The detected circle size and position could change by making a small change on any of the inputs. The input parameters of the function are:

- **image**: One byte (8-bit), single channel image.
- **method**: Currently the only developed and implemented method is "cv2.HOUGH_GRADIENT".[8]
- **dp**: This parameter is the inverse ratio of the Accumulator Resolution Matrix [9] to the search area image resolution. The larger the dp gets, the smaller the accumulator array gets.
- **minDist**: Minimum distance between centers (or edges) of the circles. The smaller this parameter is, the more circles are detected.
- **param1**: Gradient value used in edge detection to define edges.
- **param2**: Accumulator threshold value for parameter method. Small value for this parameter results in the function detecting more circles. The output Circles are sorted by the size of the accumulator value.
- **minRadius**: The minimum radius of the area used by algorithm to search for circles.
- **maxRadius**: Maximum radius that ends the range the algorithm searches for circles in it.

The first two parameters are known and the last two can be estimated by the size of the marker in the previous frame. The optimization algorithm needs to optimize the four other input parameters. Optimization problems are in two categories. If the function under optimization is defined[10] and could be formulated mathematically. One common method to optimize such functions is Gradient Descent[11]. The function Circle Hough Gradient is not a function with constant characteristic[12]. The easiest way to optimize this function is to constantly change its inputs by generating monte carlo random numbers until the output of the function get optimized. A set of training samples and an error function is needed before optimizing the function

[8] OpenCV is Originally developed by Intel

[9] See Yao *et al.* (2010) for more information about ARM matrix

[10] A function that has constant characteristics.

[11] See the Bottou (2012) for more information about optimization problems solved by Gradient Descent

[12] It has nonlinear behavior with many possible local minimum

parameters. Training samples could be captured from any frame and an error function is required.

2.4.1.1 Error Function

The error function measures the difference between the obtained circle from circle hough gradient and the circle that is showing the right place and size of the sphere marker based on the area they overlap. Each training example has coordinates and radius[13] of the marker on a specific frame. Suppose the given coordinates and radius by training set are x_{tr}, y_{tr} and r_{tr} and the correspondence parameters obtained from CHT are x_{CHT}, y_{CHT} and r_{CHT}. A candidate error function should satisfy these two conditions:

- Returns zero if the two circles overlap completely,
- Returns one if the two circles are completely separate.

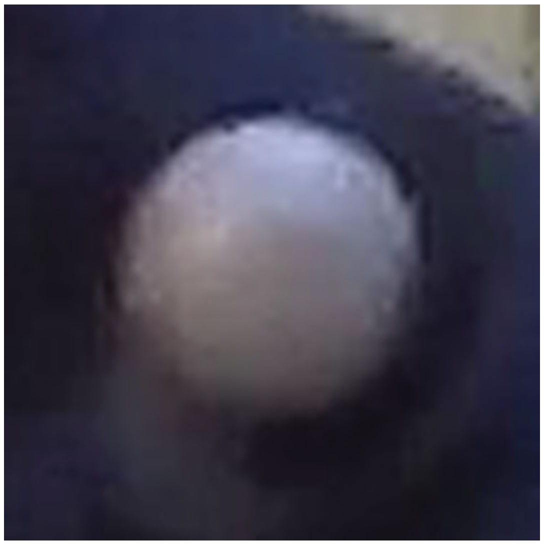
A function named "circlesSharedArea" has been designed and implemented that satisfies the two conditions above. Suppose S_{tr}, S_{CHT} and S_{shared} are accordingly the area occupied by the training sample, the area CHT is claiming to be the target and the area shared between these two. The shared area could be obtained by counting the number of pixels that are in both circles. Thus, the error function could be defined as:

$$2 \times \frac{S_{CHT} + S_{tr} - S_{shared}}{S_{CHT} + S_{tr}} - 1 \qquad (2.19)$$

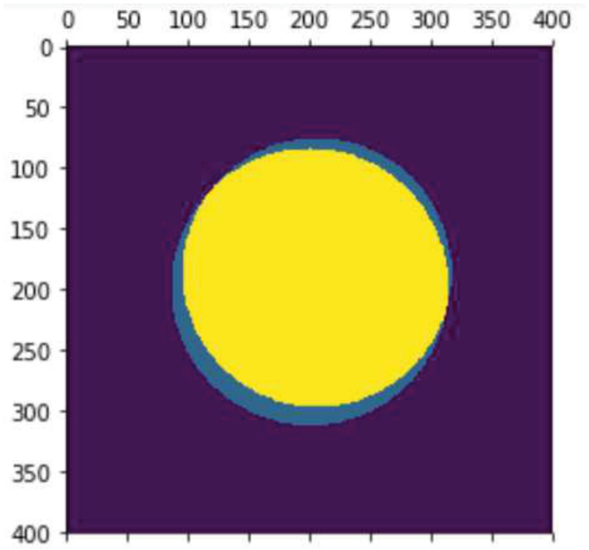
2.4.1.2 Optimizing Function Accuracy

Figure 2.6a is a training example that was captured as a random frame. This image is in fact the template area the tracker follows. Figure 2.6b shows the manually given circle as the true

[13] Size is determined by the number of pixels



a) A training sample



b) Two circles and shared area

Figure 2.6 2D views of g-HSV four dimensional sapce

position and size of the marker, the detected circle using CHT and the shared area colored in yellow. Two different optimization techniques have been developed and tested.

- Starting from a random point[14] and keep changing parameters one by one until the error function stops decreasing[15]. This approach is sensitive to the method parameters change and the result is local optimum but it's comparably fast.

- Check all values for all parameters and pick the set of parameters that makes the error function minimum. This approach finds the global optimum of CHT function and is much slower. All possibilities for the parameters are over 2^{40} and each iteration has its own specific time consumption[16].

Since this algorithm needs to be run only once on a large dataset, second technique is preferred and the results can be used permanently and for all situations. For dataset of 240 training

[14] Each point is a tuple of four parameters: (dp, minDist, param1, param2)

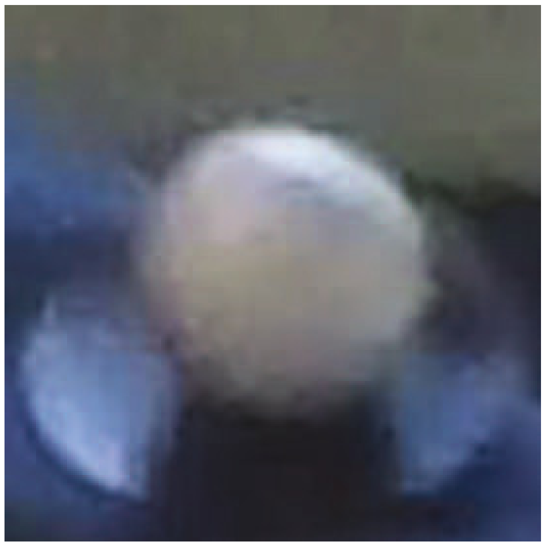
[15] Mont Carlo random sampling method

[16] Since CHT has non-linear behavior

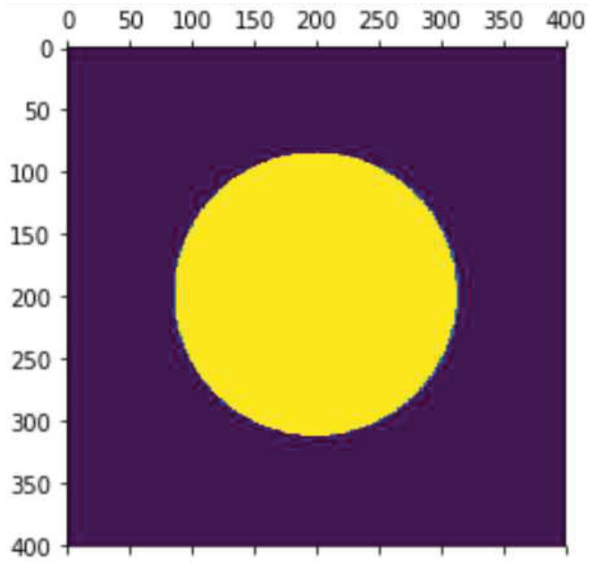
images randomly picked from three different videos, the optimum tuple of input parameters obtained is:

$$P_{opt} = (4, 1, 77, 1) \tag{2.20}$$

By applying parameters showed in equation (3.2) the average tested error measured as: $error_{avg} = 0.10648$.



a) A test sample



b) Two circles and shared area

Figure 2.7 2D views of g-HSV four dimensional sapce

Figure 2.7a shows a test sample used to check the result and and figure 2.7b is showing the two circles perfectly matched on each other and the shared area is almost equal to the CHT and training circle.

2.4.2 Color Detection

We need to develop a technique that can detect and verify the presence of an object on every frame which has common features with previous frame(s). Such method would verify on each frame if we are still following the same object or whether we are going after a wrong object (or an image patch). The method implemented for this section performs two tasks at the same time. At first, it finds the object based on its color features and then, it verifies the presence of the object that should be detected.
It converts the image (or the cropped template image containing the marker) from RGB[17] to HSV and finds a range for each color to search for an object within that range. Finally it puts a rectangle shape boundary[18] around the found object.

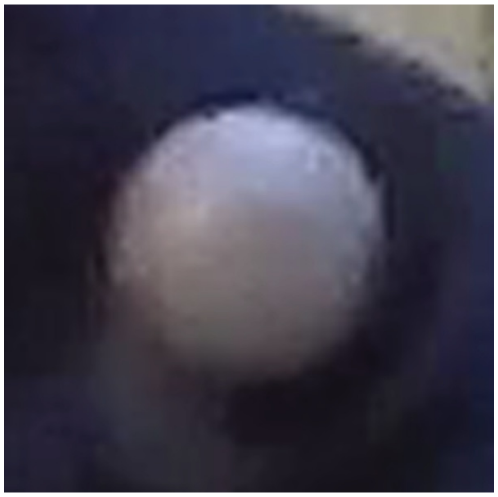
2.4.2.1 HSV vs RGB

HSV stands for "Hue", "Saturation" and "Value". To have a better understanding why we are using HSV color space over RGB lets assume we want to find a specific exclusive color range for the marker. The possible minimum and maximum of each range is 0 and 255 for red, green and blue. An ideal range is narrow and exclusive to the colors containing only the colors of the object. Figure 2.8b shows how vast and indistinguishable the RGB color range is.

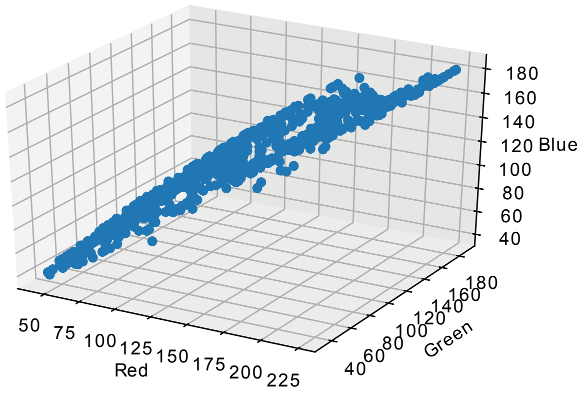
The range of color obtained here for demonstrated object is vast. These ranges for red, green and blue are (44-223), (34-187) and (40-181) respectively. These ranges for RGB contains almost all other colors. In other words if we keep defining specific range of colors in RGB color space, we would not be able to distinguish between different colors.
We need to convert the RGB to a color space that could keep the color of the target in a short narrow range and makes it exclusive to the object only. By trying the same approach for the same image in HSV color space, we find a more useful range.

[17] Red, green and blue color space

[18] A red rectangle box is used for proposed method.

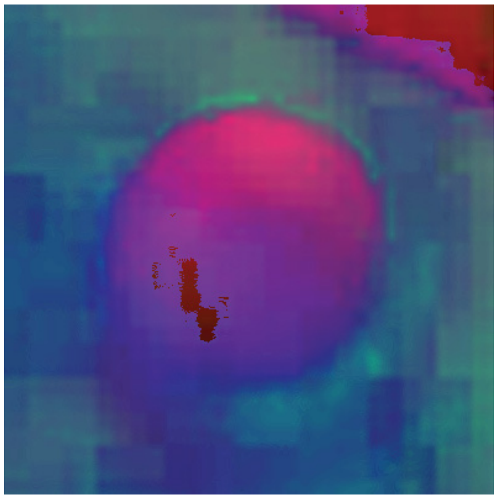


a) Object Showed in RGB Image

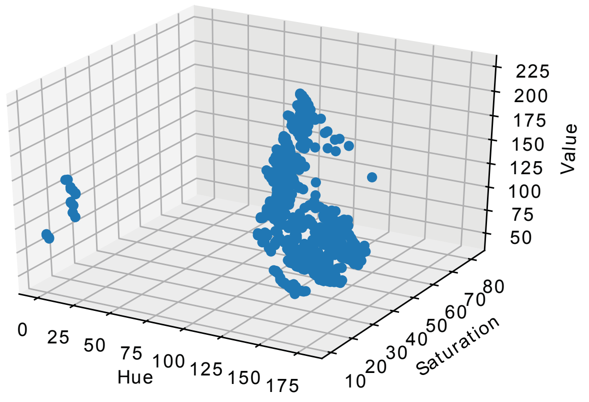


b) Object in RGB Color Space

Figure 2.8 Vast Range of Colors For The Object in RGB



a) Object Showed in HSV Image



b) Object in HSV Color Space

Figure 2.9 Range of Colors For The Object in HSV

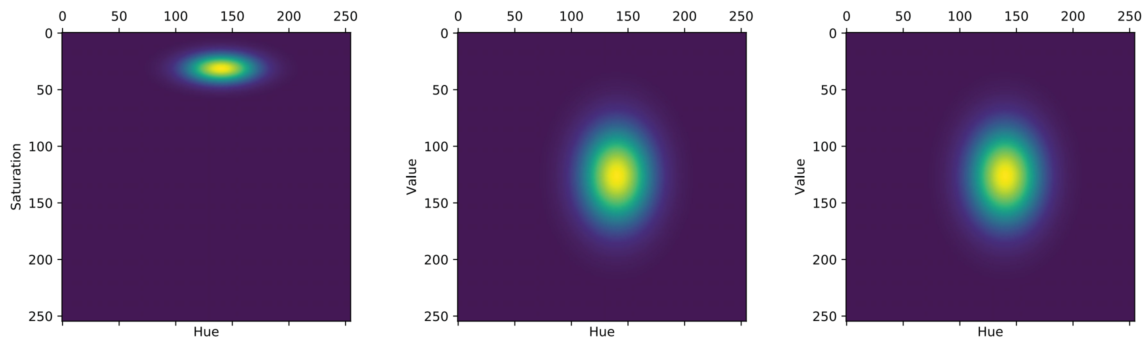
The color range that HSV is giving us is (0,178), (11,82), (44,223) for Hue, Saturaion and Value respectively. Lets take a look at the figure 2.9b again. The small group of pixels at the left side are keeping the range unnecessarily large. These pixels are in fact the small solid red part on the marker shown in figure 2.9a caused by reflection of light on the marker.

2.4.3 Optimizing Selected Part

To develop a method that removes unnecessary pixels from HSV space color we need to see how pixels are distributed around the mean of each range for hue, saturation and value.
By using the Gaussian function we would be able to see how close the colors are to the mean of each range. We define a Gaussian function of “H", “S" and “V" as $g(h;s;v)$.

$$g(h,s,v) = \exp\Big\{ -\Big(\frac{(h-\bar{h})^2}{2\sigma_h^2} + \frac{(s-\bar{s})^2}{2\sigma_s^2} + \frac{(v-\bar{v})^2}{2\sigma_v^2} \Big) \Big\} \tag{2.21}$$

where for $x =$ “h”,“s” or “v”, $\bar{x}$ is mean of x and $\sigma_x^2 = \overline{(x-\bar{x})^2}$ is variance. Function $g(\cdot)$ should be depicted in four-dimensional space. We project it to 2D and 3D pictures to make analyzing it easier. $g(\cdot)$ is a bell shaped function in 4D space. Figure 2.10 demonstrates what the function $g(\cdot)$ is. Each figure is a 3D bell shaped surface while we are looking at it from top. It gets brighter where g has higher value and points are closer to the mean. The reason why they have oval shapes is because the variances are different in each axis.



a) g-H-S b) g-H-V c) g-S-V

Figure 2.10 3D views of g-HSV four dimensional space g-H-S, g-H-V and g-S-V are projections of $g(h,s,v)$ into 3D spaces. They are obtained by taking integrals through the absent dimension:

$$g_{H-S}(h,s) = \int_{-\infty}^{\infty} g(h,s,v)\ \mathrm{d}v \tag{2.22}$$

$$= A\ \exp\Big\{ -\Big(\frac{(h-\bar{h})^2}{2\sigma_h^2} + \frac{(s-\bar{s})^2}{2\sigma_s^2} \Big) \Big\} \tag{2.23}$$

where $A = \sqrt{2\pi\sigma_v^2}$ is a constant value and it only affect the magnitude and not the shape of the projected surface. The two other projections g-H-V and g-S-V can be computed the same way. Figure 2.10 gives us an intuition of the function $g(\cdot)$ and how the values of variances are with respect to each other but it does not show us the aggregation and density of data points. To find out how data is distributed on each axis a new demonstration is needed. Figure 2.11 shows us how data is distributed around its mean on each axis.

Figure 2.11a is showing how the small group of points at the left is keeping the hue range large. What is obvious in figure 2.11 is that most points are gathered around the mean of data. These results are giving us a hint how to filter the data to keep the range for each color narrow.
Before going through the interpretation of figure 2.11, the functions g-H, g-S and g-V need to be discussed. Since we are dealing with discrete points, g-H is obtained by getting summations over all absent dimensions:

$$g_{\mathrm{H}} = \sum_{s=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} g(h,s,v) \tag{2.24}$$

$$= C\ \exp\Big\{ -\Big(\frac{(h-\bar{h})^2}{2\sigma_h^2} \Big) \Big\} \tag{2.25}$$

where $C = 2\pi\sqrt{\sigma_s^2\sigma_v^2}$ is constant and it does not affect the distribution of data along hue axis. The problem is now to find a narrow enough range for each of hue, saturation and value. So

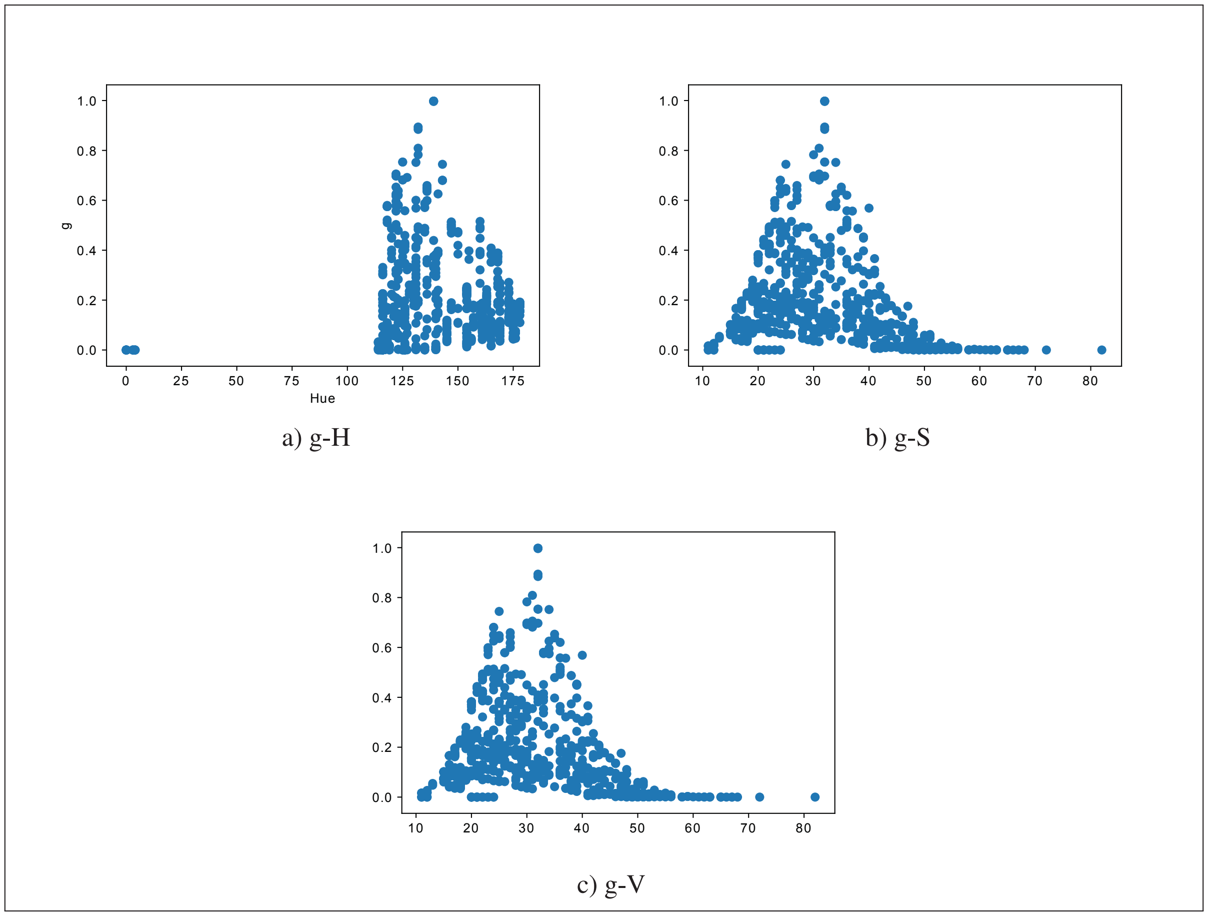


Figure 2.11 2D views of g-HSV four dimensional sapce

far, we know the defined range for each axis should be around mean of data on that dimension. Such a filter is a rectangular cube in figure 2.9b that contains most data points of the object. We call the smallest cube that contain all data points of object, the maximum-size cube. If we put the center of a cube of size zero at the mean point of data in HSV space and start enlarging it's dimensions, its volume would get larger and contains more points of object. The process should be stopped at a threshold where enlarging the volume is not adding much of data into the cube. This threshold is defined where the percentage of enlarged volume is more than the percentage of added points into the cube $\triangle V\% \geq \triangle P_{HSV}\%$ or better yet:

$$\frac{\triangle V}{V_{C_{max}}} \geq \frac{\triangle P_{HSV}}{N_P} = \theta \tag{2.26}$$

where $V_{c_{max}}$ and N_P are volume of maximum-size cube and number of all points respectively. We designed a function named “ColorSmartBoundary" that finds an optimal cube containing an arbitrary percentage of data. Now by designing a filter that could keep 93% of data around the mean and removes others, we can decrease the effect of out of range points and have ranges as (114-175), (17-82) and (69-223) for hue, saturation and value respectively. The rectangular cube that contains this data has almost one-fourth of volume compared to the maximum-size cube. Hence removing unnecessary data helped us to find much smaller ranges for colors that aims to increase the accuracy of detection by object color.

2.4.3.1 Conclusion and results

Color Based Object Detection

Detecting an object based on its color is searching for its signature color once an specific cube of colors is assigned to it. All the experiments are done in HSV color space. Next step is to define a binary mask that searches the whole frame or tracked template area and filters all pixels that are not in desired range of colors. All the pixels with wrong colors are mounted to zero and all right colored-pixels are considered as one. Figures 2.12a, 2.12b and 2.12c are original pictures with the center of the color range considered for it. Figures 2.12d, 2.12e and 2.12f are masks designed for green, red and blue respectively and figures 2.12g, 2.12h and 2.12i are detected part of original image with correspondence masks.

Results The color based algorithm is written and has been tested in Python. Figure 2.13 shows the final detection of the area including the object.

2.4.4 Verifying Lightning Endurance

One of the greatest challenges caused by probabilistic characteristics of real environment is brightness changes. Since the markers are moving during the process, the angular position of the source of the light(s) will change. Hence brightness intensity would change by time.
We developed a measurable method to accurately investigate the effect of lightning on accuracy

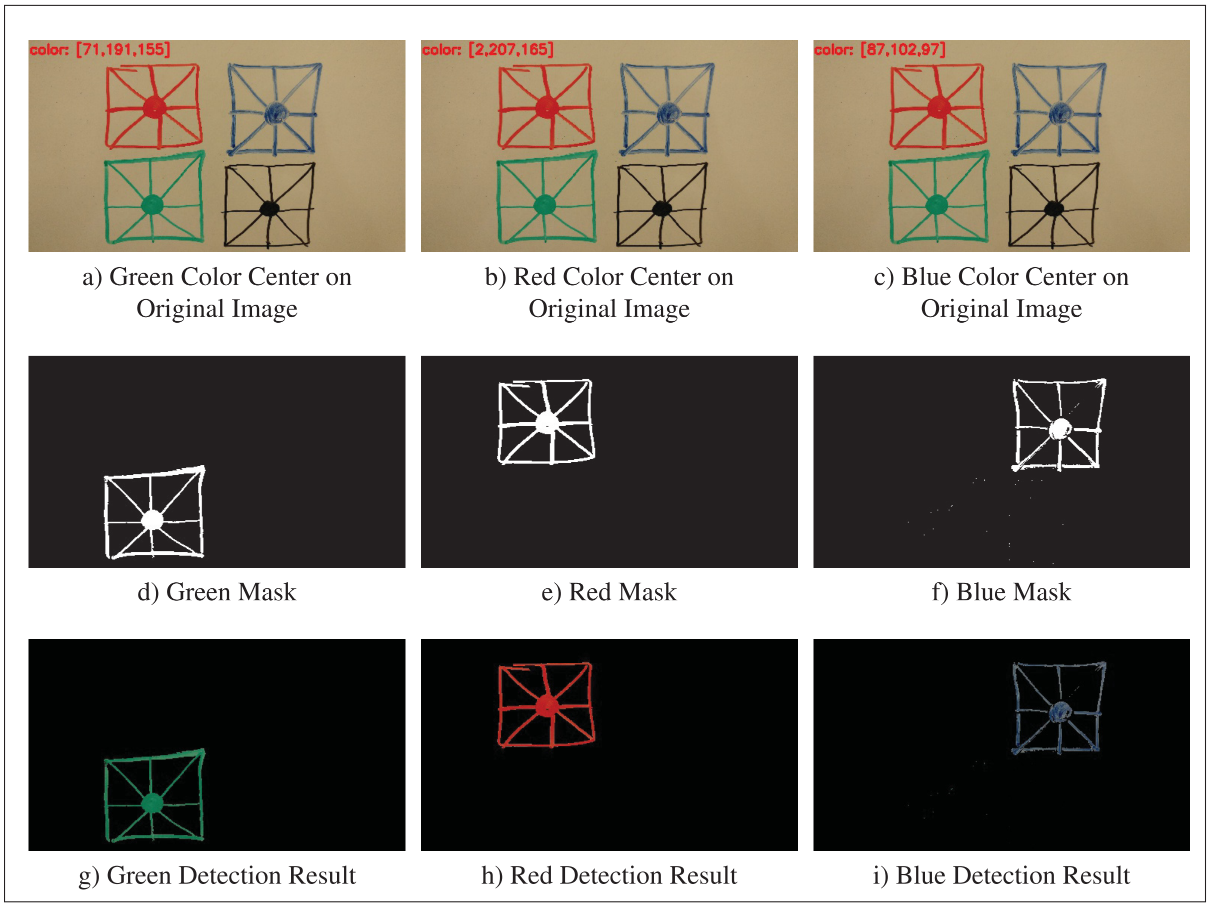

color: [71,191,155]
color: [2,207,165]
color: [87,102,97]
a) Green Color Center on Original Image
b) Red Color Center on Original Image
c) Blue Color Center on Original Image
d) Green Mask
e) Red Mask
f) Blue Mask
g) Green Detection Result
h) Red Detection Result
i) Blue Detection Result

Figure 2.12 An experiment of detecting specific desired colors and filtering others

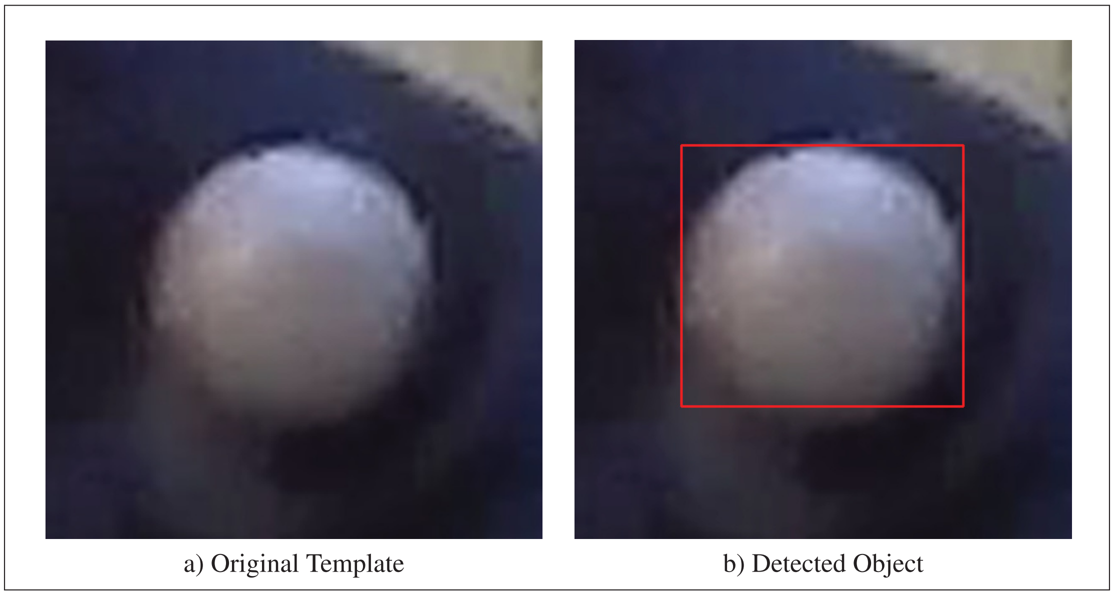


Figure 2.13 Object detection based on color

of detection. For this purpose, we used a captured video as reference and tested the accuracy on different versions of this reference made by processing it and producing exact equivalent videos with different brightnesses from dark measured as zero brightness to the brightest measured as 255 brightness.

In this method, a set of 100 true positions are given by hand as reference pose and the output of the algorithm is compared with the reference pose using an accuracy function to see the changes in accuracy that could be caused by illumination changes. The accuracy function is discussed in more details in the next chapter. Look at the equations (3.13) and (3.13) for more information.

This experiment is repeated for 30 different lightning intensities and six different markers. That means the developed algorithm has been run about 180 times over the reference video and its processed versions to achieve comparable results.

Figure 2.14 shows the results of accuracy alterations based on changes in brightness. What it shows is that if the marker is too dim or too bright, the detection would be more challenging and the accuracy could drop dramatically.

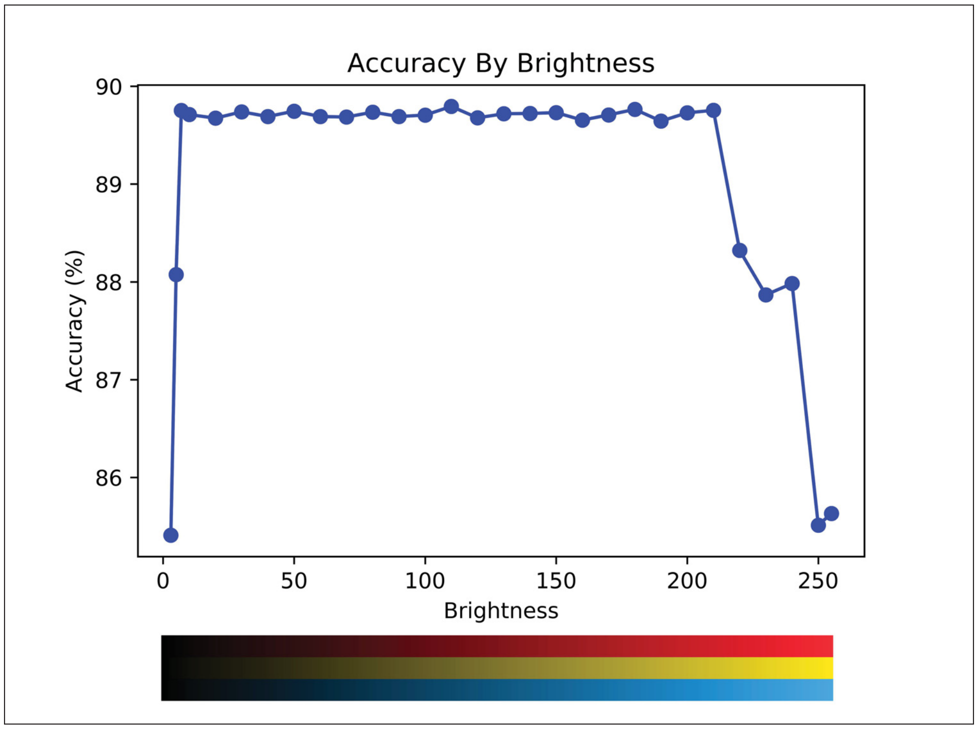


Figure 2.14 Accuracy Under Illumination Changes

CHAPTER 3

VERIFYING AND VALIDATION

3.1 Verifying True Detection

The algorithm performs three tasks before it considers verification. Firstly, it has to track the image patch and find the template image in the new frame. Secondly, it should find the target within the tracker output. Finally, it should find and verify the color of the object. Then, six separate verification units vote if the claimed area is in fact the desired object. The basic of all verification units is template matching. Template matching methods are vary and each is robust against a specific challenge.

3.1.1 Cross Correlation

Briechle & Hanebeck (2001) proposed we measure the similarity based on “Normalized Cross Correlation" and developed a fast method that reduces computational costs. Cross correlation is an accurate method but it is computationally expensive. We reformulate cross correlation to make it compatible with matrix computation and higher performance.

Suppose we have two gray images f and g with the same size of $N \times N$. This is a valid assumption since template could have an arbitrary size of N cropped from search area. The cross correlation coefficient is defined as follow.

$$\rho_{f,g} = \frac{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cdot g(x,y)}{\sqrt{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)^2 \cdot \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g(x,y)^2}} \tag{3.1}$$

where (x,y) is location of a pixel and $f(x,y)$ and $g(x,y)$ are intensities of that pixel in the two images. The numerator of equation (3.1) is the Hadamard product of two matrices f and g. The rows and columns of images are indexed from zero to make it compatible with python compiler. In case of implementing in MATLAB the reader may consider indices starting from one. We assume that f and g are both normalized and have the mean of zero.

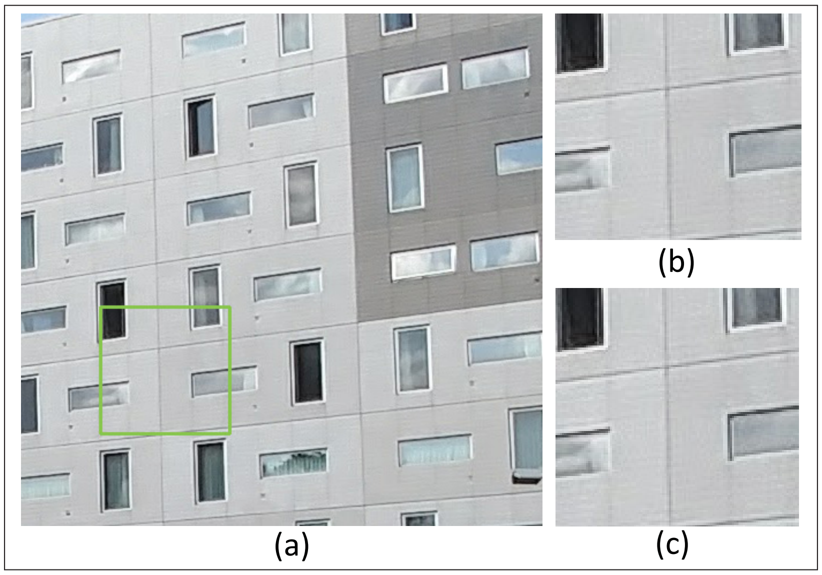


Figure 3.1 a) Search Area, b)Template, c)Found Image.

Now assume the search area and template are $M \times M$ and $N \times N$ matrices respectively where clearly $M > N$. We need to compute ρ for all shifted locations[1].

$$\rho_{f,g}(u,v) = \frac{\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cdot g(u+x,v+y)}{\sqrt{\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)^2 \cdot \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} g(u+x,v+y)^2}} \tag{3.2}$$

u and v are the position of shifted template in search area: $u,v = 0,1,2,\cdots,M-N$

The numerator is convolution of f and g in negative direction and could be replaced with

$$\mathscr{F}^{-1}\{\mathscr{F}(f)\mathscr{F}^*(g)\} \tag{3.3}$$

where $\mathscr{F}\{\cdot\}$ and $\mathscr{F}^{-1}\{\cdot\}$ are the Fourier transform and the Fourier inverse transform operations and $*$ means the complex conjugate. The first element in denominator, $\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)^2$

[1] Which would be $(M-N+1)^2$ different positions

is related to the template and needs to be computed once for all coefficients.

Sarvaiya *et al.* (2009) argues that using “Fast Fourier Transform (FFT)" equation (3.1) has computational complexity of $12M^2 \log_2 M + N^2(M-N+1)^2$ addition of real numbers and $12M^2 \log_2 M + N^2(M-N+1)^2$ multiplication of real numbers. If M is much larger than N the computational complexity of equation (2.2) is M^2N^2. In these cases the direct method would perform faster than FFT.

I implemented the template matching algorithm with normalized cross correlation method. Figure 3.1 shows how this method is accurate and always find the exact position of the template in search area if we assume certain constrains.

The accuracy of cross correlation is high but there are a few important drawbacks. The first victim is performance. By running this algorithm on a system with intel core i7 and 8GB RAM where the search area is 360×360 matrix and template size is 180×180 the consumed time for implementation on Python with direct convolution method is 5.743 seconds. Assuming the same size for reference image and template in this research knowing the fact that our videos are recorded in 240 fps, processing each second of these videos would take 22 minutes and 58 seconds.

Dhome (2002); Wei & Lai (2008) proposed and modified two fast approaches that reduced the running time of the algorithm significantly. Dhome (2002) suggest a two-step template matching. At the first step they chose a few pixels of the template and calculate correlation coefficients by doing normalized cross correlation. Coefficients that are larger than a threshold will be recomputed with the whole template at the second step. Wei & Lai (2008) by making partitions of equation (2.2) into different levels, skipped unnecessary calculation. Their proposed algorithm is efficient under different lighting conditions. This method bring the processing time for each frame bellow one second at its optimal implementation.

The second restriction of this method is that the template must be at the same size of its match on the search area. In tracking applications the size of the object may change due to change of its distance from camera. Even in best tracking algorithms such as KCF proposed by Henriques *et al.* (2015b) changing the template size is not accurate enough. The most important drawback of this method is its incompatibility with color images. By converting frames from RGB or HSV to gray we are ignoring some information that could be useful. However we reformulated equation (3.2) and made it work with three-dimensional matrices (3.4) since the template does not move through third dimension.

$$\rho_{f,g}(u,v,t) = \frac{\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\sum_{z=0}^{2} f(x,y,z)\cdot g(u+x,v+y,z)}{\sqrt{\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\sum_{z=0}^{2} f(x,y,z)^2 \cdot \sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\sum_{z=0}^{2} g(u+x,v+y,z)^2}} \tag{3.4}$$

3.1.2 Sum of Absolute Differences

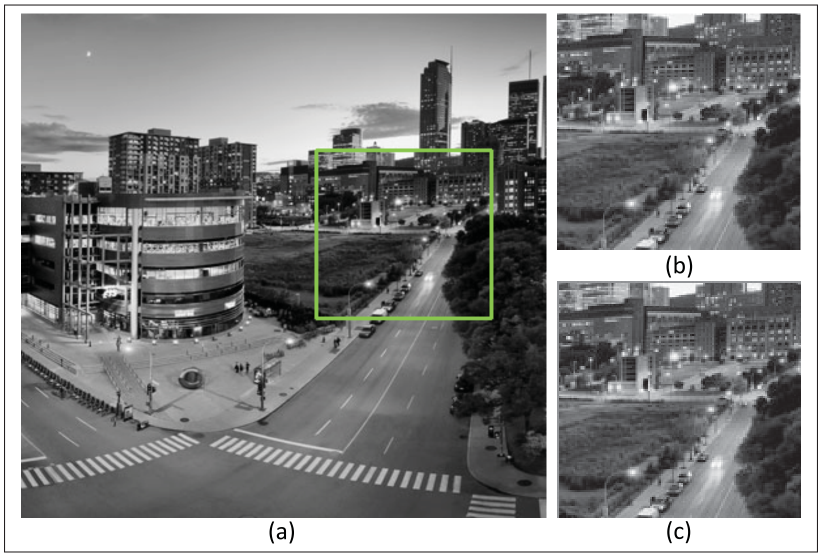


Figure 3.2 a) Search Area, b)Template, c)Found Image.

Sum of absolute differences (SAD) is faster approach that Wei & Lai (2008) claims it finds the true position of the template in the source image with this formula:

$$SAD(u,v) = \sum_{x=0}^{N} \sum_{y=0}^{N} |f(x,y) - g(x+u, y+v)| \tag{3.5}$$

The image patch positioned at $(u,v) = (u_0, v_0)$ is the best match if it's making *SAD* minimum or at its best situation makes it exactly zero.

3.1.3 Correlation Coefficient

Correlation coefficient (CCOEFF) is similar to cross correlation. CCOEFF has always a value between -1 and $+1$ and is less robust to position change but more robust to noise compared to cross correlation. Correlation coefficient between two templates f and g is defined as:

$$\rho_{i,j}(x,y) = \frac{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} (f(x,y) - \mu_f)(g(x+i, y+j) - \mu_g)}{\sqrt{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} (f(x,y) - \mu_f)^2} \sqrt{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} (g(x+i, y+j) - \mu_g)^2}} \tag{3.6}$$

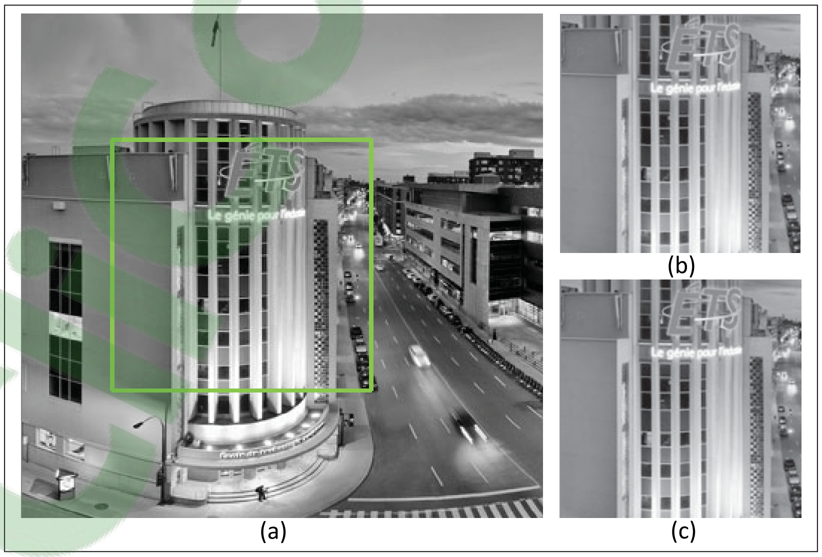


Figure 3.3 a) Search Area, b)Template, c)Found Image.

Suppose δ is a unit magnitude normalized vector with zero mean and σ is proportional to standard deviation, equation (3.6) could be written as:

$$\rho_{i,j}(x,y) = \sum_{x=0}^{N}\sum_{y=0}^{N}\left(\frac{\delta_f(x,y)}{\sigma_f(x,y)} \odot \frac{\delta_g(x+i,y+j)}{\sigma_g(x+i,y+j)}\right) \tag{3.7}$$

σ normalizes the value of the correlation coefficient witch makes it less robust to scalability but more robust normalized Gaussian noise.

3.1.4 Conclusion and results

Putting the three discussed methods together with their normalized versions, there will be a combination of six methods useful for the purpose of template matching. Cross correlation votes for energy similarity, coefficient correlation votes for scalability and sum of absolute differences is robust to a slight position shift. Figure 3.4 shows how these six methods[2] work and verifies the presence of the object as it is recognized in the previous frame.

3.1.4.1 Voting Procedure

Lets assume S_f is the area of the circle found by the proposed algorithm calculated by the obtained coordinates x_f, y_f and r_f. Also, S_t and S_{sh} are the areas of the detected by template matching and shared area between the found one by algorithm and template matching. Equation (3.8) is the portion of the circle detected by template matching that is occupied by the detected circle by the algorithm.

$$V = \frac{2 * S_{sh}}{S_f + S_t} \tag{3.8}$$

[2] including normalized and unnormalized versions

In this equation "V" is called the voting coefficient that is a number between zero and one. One is a 100% agreement of what the proposed algorithm is claiming to be the true marker and zero is a full rejection.

- If the aggregation of votes of the six template matching methods is greater than a threshold TH_1[3], then, the claimed position of the target by the proposed algorithm would be interpreted as rightful.
- If the aggregated votes of the six template matching methods is between TH_1 and a second threshold $TH_2 = 3$, then the claimed result of the proposed method is not considered to be wrong, but it's not accurate enough either. Thus the position in the current frame will be estimated and smoothed by "Kalman Filter".
- If the aggregated votes is less than the TH_2, then the algorithm will stop and ask for the true position manually from the operator.

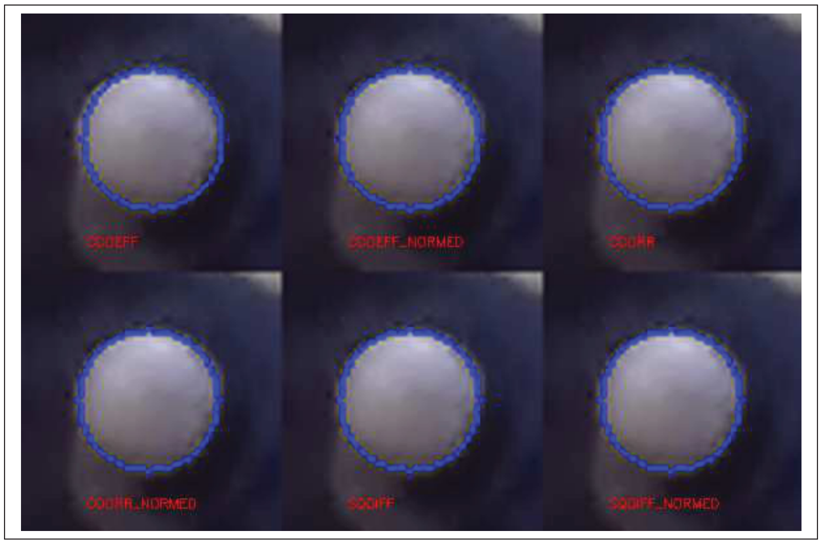


Figure 3.4 a) Search Area, b)Template, c)Found Image.

[3] $TH_1 = 4$ we defined in this case base on running this experiment tens of times in different conditions.

If any of the six methods detect the template, then the presence of the object is verified. Other methods might have error due to scalability, sudden brightness change, rotation and etc. Because each method has it's own robustness, it is expected at least one of them works if not all challenges happen at the same time to one frame.

3.2 3D Positioning

Although the purpose of this study is to find the position of sphere markers in pixels which is firstly, finding the coordinates along x-axis in pixel. Secondly detecting the coordinates in y-axis in pixels. Thirdly finding the size of the markers in pixels, we put one step forward and give a solution to convert pixel-based coordinates to real 3D coordinates in centimeter.
We need to consider the fact that all the achievements so far is not dependent to physical property of the the camera we use. Hence all the results we have obtained are valid with any camera we use. However, physical properties of cameras should be considered if we are getting 3D position in centimeter. That means every time we change the camera or a sort of settings in the same camera, we need to change some parameters and calculations to achieve valid results. Since pixel-based positioning result is enough for most further investigations in other studies as long as we use the same camera to make any two sorts of event compatible, finding 3D position in centimeter is not a necessary task and hence, we just make a quick review of how it works for only one specific camera we used.
Each camera has it's own characteristics such as pixel size, sensor size and focal length. The camera used for this study is “GoPro Hero 4 Silver". Suppose f, l_i, l_p, d and l_o are focal length, size of imaged object, size of pixel on sensor, distance of object from camera and size of the object respectively. Since the GoPro camera pixel size is $l_p = 14.2\mu m$ the size of the object can be calculate as:

$$l_i = (0.0142 \text{ mm}) \times 2r \tag{3.9}$$

where r is the radius of the marker in pixel.

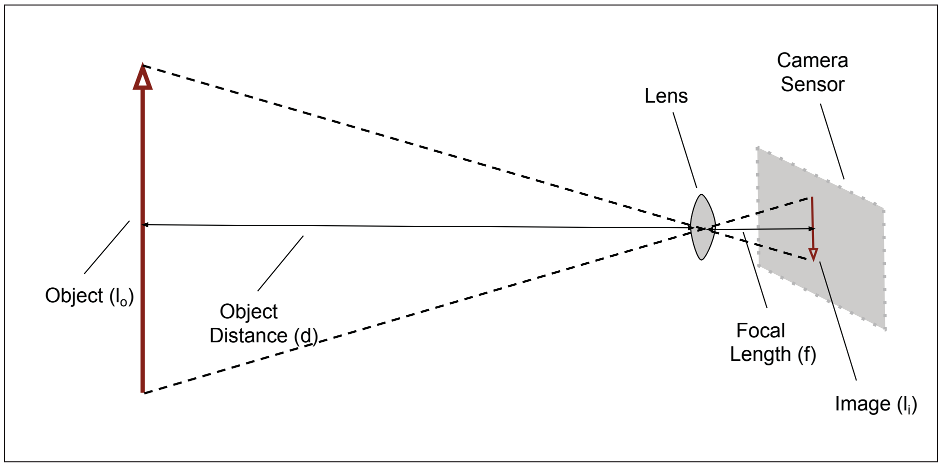


Figure 3.5 How a camera pictures objects

By looking at the figure 3.5 the triangle proportionality theorem states that:

$$\frac{d}{l_o} = \frac{f}{l_i} \tag{3.10}$$

Hence distance from camera could be calculated as:

$$\begin{aligned} d =& f \cdot \frac{l_o}{l_i} \\ =& f \cdot \frac{l_o}{2rl_p} \qquad (3.11) \\ =& 1211.27 \times \frac{l_o}{r} \qquad (3.12) \end{aligned}$$

Equation (3.12) is obtained assuming for the “GoPro Hero 4 Silver" focal length and pixel size are 34.4 mm and 0.0142 mm respectively. In this equation, radius must be put in pixel. Thus. d and l_o will have the same unit. For example, in a case where $r = 20$ pixels and $l_o = 1.7$ cm, the distance from camera is obtained as $d = 102.96$ cm.

3.2.1 Validation

The proposed method returns the coordinates of the marker and verifies its presence so far. The result is a location it's claiming as the true coordinates of the marker. But there is one more question that needs to be answered and that is “How do we know the location is true?". We are simply questioning the 3D positioning method which finds the location of the target in three-dimensional space.
To answer this question a set of experiments called “Validating Marker Location" is arranged. In this experiment a sphere marker should be placed on different and known locations. The results of the algorithm should be compared with the measured coordinates by hand This experiment has been executed over 15 times in where we put the marker in 15 different positions in 3D space with respect to camera. Figure 3.6 is a sample of this experiment and shows the algorithm is accurately detecting the 3D position of the target.

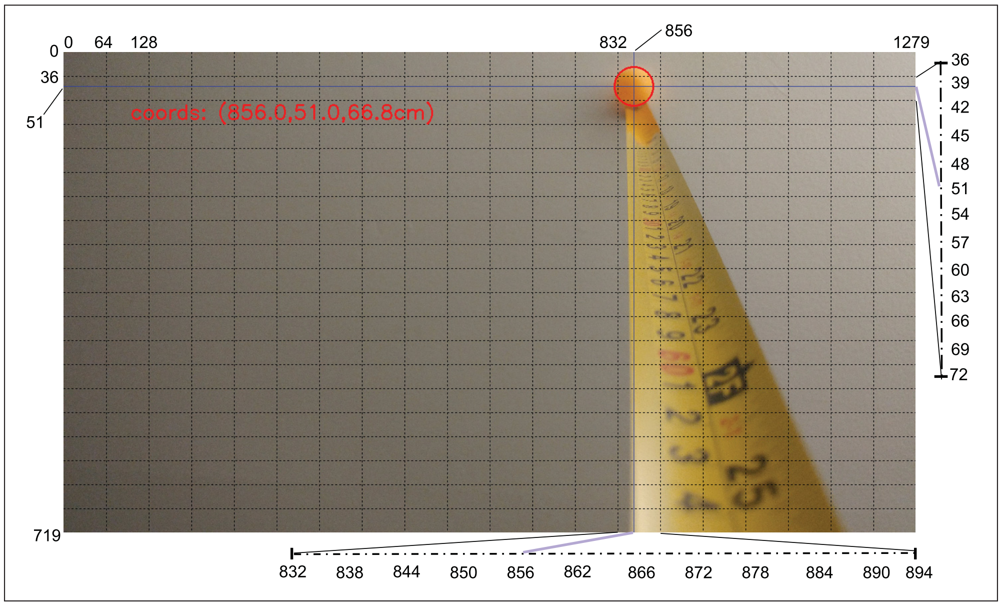


Figure 3.6 Validating Marker Location

In Figure 3.6, the coordinates written in red is in fact what algorithm is proposing. The measured coordinates by hand are (856, 51, 66.7cm) for *x*, *y* and *z* respectively.

3.3 Denoising And Prediction

The output of the algorithm is the position of the marker, so far. By looking at the output in a sequence of many frames, we notice that the output is affected by noise. This noise is caused by measurement, vibration and etc. Thus, the outputs needs to be smoothed for more real and accurate results.

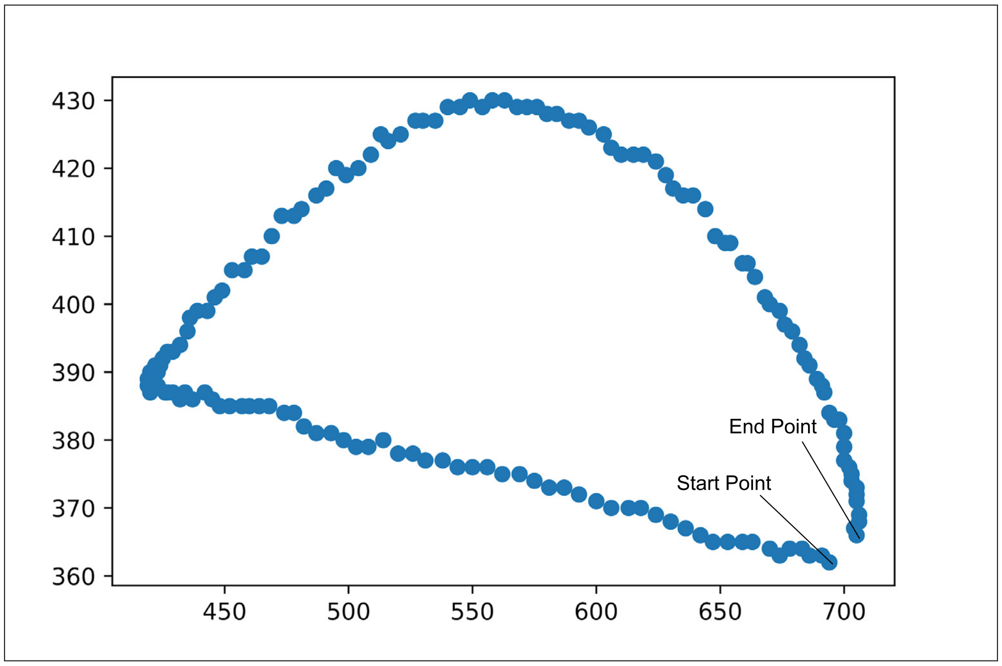


Figure 3.7 A sequence of positioned target

Figure 3.7 shows how output position could be deviated due to noise.

3.3.1 Kalman Filter

Kalman filter [4] is a method that estimates a measured variable containing statistical noise and other inaccuracies over time. This method approximate the joint probability distribution over the output position for each time step. Nada *et al.* (2018) explained the mathematical structure

[4] Also known as Linear Quadratic Estimation (LQE)

of "Kalman Filter", we use the same structure here with modification in the algorithm to 3D positioning estimation.

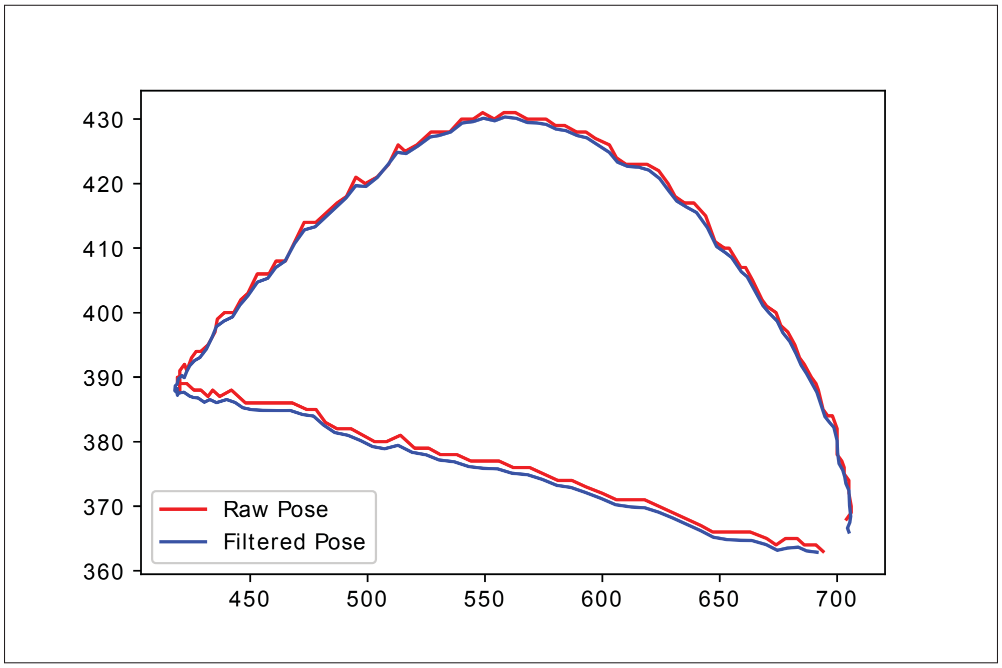


Figure 3.8 Kalman filter in 2D space

3.3.2 Final Results

To make it easier to analyze, figure 3.8 shows a two-dimensional effect of KF. Blue line is the smoothened position and red line is the output position before be given to KF. Kalman filter is shown to be more accurate than positioned obtained from a single measurement.

Figure 3.9 is pictured just to keep the generality of the solution in 3D space. The process is similar to the 2D function discussed.

3.4 Verifying by Comparison

The final results of the proposed method during this study needs to be compared with advanced and recent algorithms from the literature. “Channel and Spatial Reliability (CSRT, 2017)",

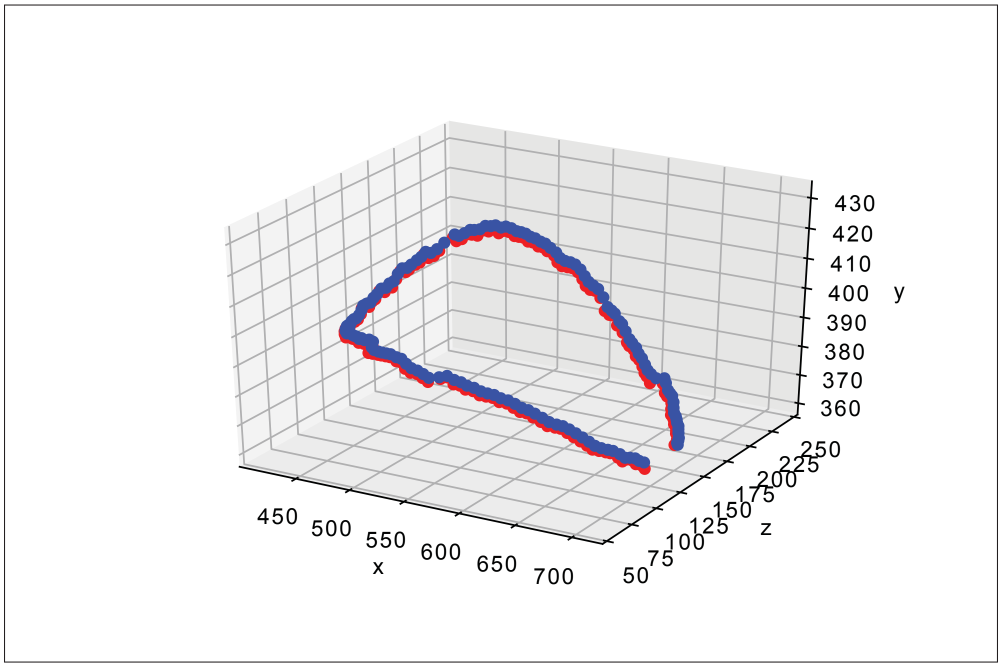


Figure 3.9 Kalman filter in 3D space

“Improved Median Flow (2016)" ,and “Boosting Tracker (2011)" are three most advanced algorithms selected at this stage to compare the results.

3.4.1 Experiment

The main application we use to design our method and algorithm is detection and tracking of markers attached to shoulders, elbows and wrists of a physically impaired athlete person while driving a three-wheel racing wheelchair. A Go-Pro forth generation camera is installed above the front wheel and captures at 240 fps with the resolution of 1280×720 with narrow field of view (FOV) to minimize curved-capturing effect. In this chapter, we compare our method with three other well-known algorithms in the conditions that satisfies our main application. Thus, videos of a person on a wheelchair moving arms are used in this chapter.

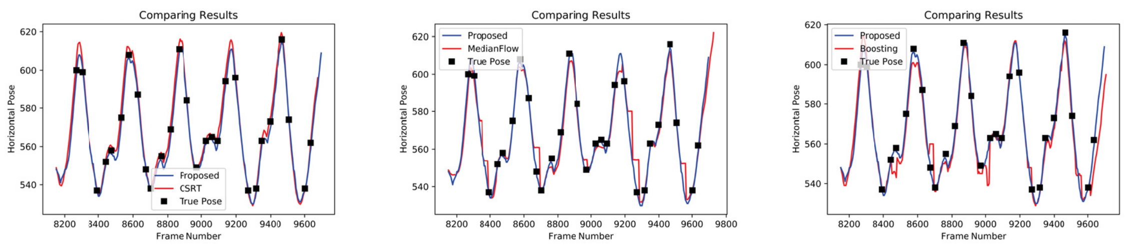
A video of a yellow sphere marker attached to the left wrist is captured using a “Go Pro 4" camera with 240 frames in each second. All algorithms started running from the frame number 8150 and kept running for the next 1600 frames. The true position is given to the algorithms at the very first frame.

The proposed method and the CSRT algorithm ran the video without losing the target, but the other two tested algorithms failed to follow the target many times. In order to have the detected positions in all frames, if an algorithm lost the target, the algorithm was paused manually[5] and the true position is given in the middle of the video again.

The true position in many random areas manually selected by hand is also given to compare the plotted lines as well. The captured marker moves over an almost circular path during the video sequences.

3.4.2 Comparing Results

Since the position of the marker has three dimensions, each dimension should be plotted separately, Figures 3.10, 3.11, and 3.12 compares the results in horizontal axis. The red line shows the position of the marker in pixels on horizontal axis detected by the proposed method and red lines are competitors.



a) Proposed method vs CSRT

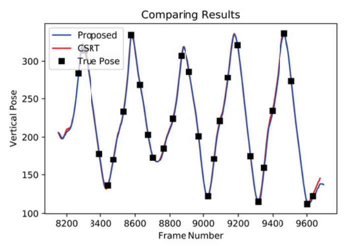
b) Proposed method vs Median Flow

c) Proposed method vs Boosting

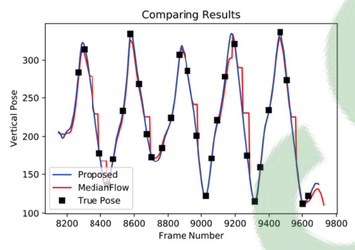
Figure 3.10 Comparing algorithms in horizontal axis

[5] The proposed method does not lose the target during this experiment, however it is developed to pause the algorithm automatically if it senses that there are major differences between the target and detected object.

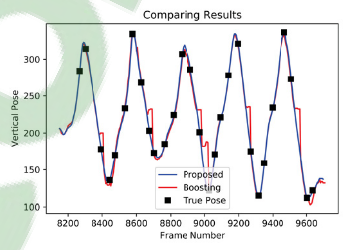
Next three figures compares the positions obtained from proposed and the three testing algorithms in vertical axis in pixels.



a) Proposed method vs CSRT



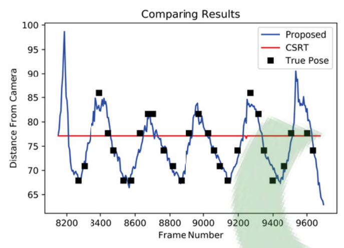
b) Proposed method vs Median Flow



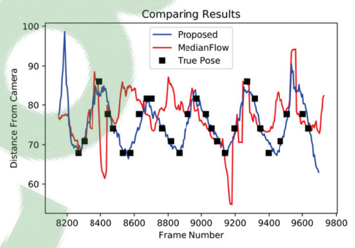
c) Proposed method vs Boosting

Figure 3.11 Comparing algorithms in vertical axis in pixels

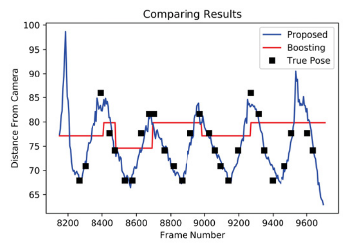
The next three figures compares the proposed algorithm with the other three by distance of the marker from camera in centimeters.



a) Proposed method vs CSRT



b) Proposed method vs Median Flow



c) Proposed method vs Boosting

Figure 3.12 Comparing captured distances from camera in centimeters

3.4.3 Conclusion And Results

The figures 3.10, 3.11, and 3.12 show how different algorithms follow the target comparing to the proposed method. However, a technique to measure the accuracy is needed. Firstly, we need to define a measurement for accuracy or error that is used frequently in the literature. Let's assume the true manually selected area (S_r) and the area found by an algorithm (S_f) share some area called S_{sh} error function is defined as follow:

$$Error = 1 - \frac{2 * S_{sh}}{S_r + S_f} \tag{3.13}$$

This "error" shows how much portion of both areas is not shared by the two areas. This amount is between zero and one. Thus, the accuracy could be defined in percentage as follow:

$$Accuracy = (1 - Error) \times 100 \tag{3.14}$$

Accuracy shows how many percents of the total area occupied by the two areas are shared between them. Zero accuracy is interpreted as false detection or failure and Accuracy = 100 means the detected object and the real target are matched completely.

Table 3.1 Accuracy

		Accuracy (%)			
	Video Samples	**Proposed**	**CSRT**	**Median Flow**	**Boosting**
-	#1	**98.4**	91.6	91.1	89.2
	#2	**97.0**	92.3	87.9	90.6
	#3	**98.1**	93.1	89.9	92.0
	#4	**96.5**	91.5	90.1	88.6
	#5	**96.3**	90.7	88.1	89.9
Average	-	**97.3**	91.4	89.4	90.1

In table 3.1 the highest number in each row is the bold one and the highest number in each column is in red.

Table 3.2 shows how many times we had to pause the algorithm and give the true position manually again. Failure happens in four situations:

1. Occlusion makes the marker disappear for a number of frames or makes the marker undetectable,
2. An object with a similar shape passes by the target,
3. An object with a similar color passes by the target,
4. Accumulated error over time causes the algorithm losing the position of the marker.

Table 3.2 Total Number of Failures (Losing Target)

Failure (#)			
Proposed	**CSRT**	**Median Flow**	**Boosting**
0	8	29	41

CHAPTER 4

CONCLUSION AND FUTURE WORK

Tracking and detection of markers are vastly discussed in the literature and hundreds of different methods are proposed so far. Each method has its own restrictions such as speed of target, shape of the object, illumination influences, partial or total occlusion, and etc. The goal of this study is to find an efficient method (or combination of methods) to determine the position of chosen markers and enhance accuracy and performance.

4.1 Tracking

The developed algorithm starts by receiving the true position from the user (operator) in the first frame. Then the tracker should find a similar image patch in the next frame. If the marker is moving with respect to the background, then the tracker could fail following the target if the size of the image patch is too large with respect to the size of the marker. Since the size of the marker used in this study is small compared to the normal targets most trackers are designed to follow, choosing a small image patch could cause a failure, too. It's a delicate balance to choose a true template size in a range which the algorithm successfully keeps following the right object. On the other hand a small error in the number of pixels in tracking or detection of the target could cause a big error in the percentage of accuracy since the size of the markers or a few pixels only.

Since we can control the shape, size and color of the marker, we can choose markers that gives us better results. Assuming the greatest distance a marker could have with respect to camera, size of the marker should have a minimum amount that tracking system does not fail to follow at its smallest visible size (greatest distance from camera).

Kernelized Correlation Filters (KCF) as discussed in the literature review, is the leading technique that has low failure rate in case of partial occlusion, high performance that can track an object 150 frames per second, and high endurance in illumination variations. Although it normally fails to detect changes in size of the marker, could falsely detect an adjacent similar object and starts following that wrong object and lose accuracy if object is similar to the background. Thus, detection techniques are used to overcome the weaknesses of the tracking system.

4.2 Detection

Edge detection techniques normally have problem to detect object with small number of pixels wide. Thus, we developed a "monte carlo" approach to optimize its result, however edge detection are not able to function accurately once the object move fast, brightness changes dramatically, or object gets too far from the camera. To bring the accuracy back up, we developed a color based detection method that converts the image patch to HSV system. Since we can choose the color of the object, firstly, we better choose a color that does not repeatedly appear frequently in the background and, secondly, it has a clear valid range in the spectrum of HSV system. For instance, blue and yellow are defined in unique ranges of HSV spectrum but red is in two separate ranges.

4.3 Verification

Combination of tracking and detection methods claim a coordinate for position of the marker. Six different template matching methods will vote how truthful this claim is. If verified, the algorithm continues to the next frame. In case of failure, two scenarios are considered.

1. If failures happen in a few frames and verification unit start verifying again after, the unverified frames will be smoothened by "Kalman Filter".
2. If the verification keeps failing for a number of frames in a row, the algorithm would go back to the first unverified frame and asks the user (operator) for the position of the target manually.

4.3.1 Illumination Endurance

Brightness changes could affect the accuracy and could increase the number of failed detection and tracking. If the lightning is too dim or too bright, it would be harder for algorithms to follow or find an object. We tested our algorithm in 30 different lightning intensities for six different markers. Figure 2.14 shows how the average accuracy of those markers is dependent to the changes in brightness.

4.4 Comparison

The result of developed algorithm is compared with three well know and leading recent researches, CSRT, Boosting, and Median Flow. Table 3.1 shows a detailed comparison between these four methods. In this experiment, we can see that we reached a promisingly high accuracy comparing to state-of-the-art methods. Since we could customize the marker's properties to maximize the accuracy related to color detection method we use, a yellow marker with a diameter of four centimeters is chosen.

4.5 Future Work

Detection and tracking objects is a challenging task and many researches have been done so far to overcome with its challenges. Each method is optimized for certain conditions or applications. This research is no exception. We designed a method and an algorithm which tracks and detects sphere colored markers. The true position should be given in the first frame and total occlusion could cause a failure. In case of total occlusion the algorithm would stop running and would ask the operator to manually put the position of the marker again. An all in one algorithm is missing that works for any object of any shape and color and could run with no failure in case of occlusion, illumination changes and all other possible challenges. In this study we could reach higher accuracy with respect to state-of-the-art techniques for our specific problem of sphere markers. The algorithm would not fail if partial occlusion happens and the performance is quite high.

Some problems will be solved automatically in the future since the specifications of new cameras are getting enhanced year by year. For example we expect that frame rate video capturing to be increased by time. Today, cameras can record videos with higher frequencies and higher resolution. Increasing frame per second speed, could help to easier overcome with motion blur and higher frequency makes the edge of objects more visible. Thus the edge base algorithms could work more accurately.

We suggest that a future work could focus on dealing with total occlusion. We need to accept the fact that even human eyes could detect a false similar object in case of total occlusion. Thus, reaching the goal of zero error seems to be unrealistic in real world.

We use markers to find the position of joints on the body that markers are attached to them. A future work could develop a method to somehow detect joints themselves instead of using markers. Of course this method must have greater or equal accuracy to the state-of-the-art methods. Such method would probably includes a neural network that is trained to understand body parts itself. Unfortunately the researches in this area such as tele-immersion presented by Leigh *et al.* (2001) have not reached any acceptable accuracy so far. Although many of those algorithms are not able to understand the difference between joints and other parts of the body. Hence reaching the goal of accurately and directly detecting the joints of body is still a big challenge today.

BIBLIOGRAPHY

Agrawal, S. & Dean, B. K. (2018). Multiple Cartridges Improve Edge Detection Algorithm for Fly Inspired Vision System. *2018 IEEE SENSORS*, pp. 1–4.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade* (pp. 421–436). Springer.

Bouchahma, M., Barhoumi, W., Yan, W. & Al Wardi, H. (2017). Optical-flow-based approach for the detection of shoreline changes using remote sensing data. *Computer Systems and Applications (AICCSA), 2017 IEEE/ACS 14th International Conference on*, pp. 184–189.

Briechle, K. & Hanebeck, U. D. (2001). Template matching using fast normalized cross correlation. *Optical Pattern Recognition XII*, 4387, 95–103.

Chattopadhyay, S., Roros, C. J. & Kak, A. C. (2019). A Collaborative Algorithmic Framework to Track Objects And Events. *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 4000–4004.

Chaudhary, K., Zhao, M., Shi, F., Chen, X., Okada, K. & Inaba, M. (2017). Robust real-time visual tracking using dual-frame deep comparison network integrated with correlation filters. *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 6837–6842.

Chee, K. W. & Teoh, S. S. (2019). Pedestrian Detection in Visual Images Using Combination of HOG and HOM Features. *10th International Conference on Robotics, Vision, Signal Processing and Power Applications*, pp. 591–597.

Chen, L.-C., Barron, J. T., Papandreou, G., Murphy, K. & Yuille, A. L. (2016). Semantic image segmentation with task-specific edge detection using cnns and a discriminatively trained domain transform. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4545–4554.

Chen, Z., Li, M. & Yu, H. (2017). A 3D circular object detection method based on binocular stereo vision. *Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2017 10th International Congress on*, pp. 1–7.

Chrysos, G. G., Antonakos, E. & Zafeiriou, S. (2018). IPST: Incremental Pictorial Structures for Model-Free Tracking of Deformable Objects. *IEEE Transactions on Image Processing*, 27(7), 3529–3540.

Cornelia, A. & Setyawan, I. (2017). Ball detection algorithm for robot soccer based on contour and gradient hough circle transform. *Information Technology, Computer, and Electrical Engineering (ICITACEE), 2017 4th International Conference on*, pp. 136–141.

Dewi, D. A., Sundararajan, E., Prabuwono, A. S. & Cheng, L. M. (2019). Object Detection without Color Feature: Case Study Autonomous Robot. *International Journal of Mechanical Engineering and Robotics Research*, 8(4), 646–650.

Dhome, F. (2002). Real time robust template matching. *Proceedings of BMVC*, pp. 124–132.

Divakaran, A., Yu, Q., Tamrakar, A., Sawhney, H. S., Zhu, J., Javed, O., Liu, J., Cheng, H. & Eledath, J. (2018). Real-time object detection, tracking and occlusion reasoning. Google Patents. US Patent 9,904,852.

Dollár, P. & Zitnick, C. L. (2015). Fast edge detection using structured forests. *IEEE transactions on pattern analysis and machine intelligence*, 37(8), 1558–1570.

Duncan, S., Regenbrecht, H. & Langlotz, T. (2019). A Fast Multi-RGBD-Camera Calibration. *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 906–907.

El-Ghoboushi, M., Ghuniem, A., Gaafar, A.-H. & Abou-Bakr, H. E.-D. (2018). Multiple aircrafts tracking in clutter for multilateration air traffic surveillance system. *Innovative Trends in Computer Engineering (ITCE), 2018 International Conference on*, pp. 225–230.

Gao, J. & Cai, X.-f. (2017). Image matching method based on multi-scale corner detection. *Computational Intelligence and Security (CIS), 2017 13th International Conference on*, pp. 125–129.

Goodsitt, J. E., Walters, A. G., Abad, F. A. T., Taylor, K., Farivar, R., Pham, V. & Truong, A. (2019). Techniques to improve edge detection for images. Google Patents. US Patent App. 10/311,577.

Guo, D., Chen, K., Hu, X., Wei, Y. & Li, J. (2019). A Survey of Prototype Side-channel Attacks Based on Machine Learning Algorithms for Cryptographic Chips. *Journal of Physics: Conference Series*, 1176, 032005. doi: 10.1088/1742-6596/1176/3/032005.

Haggui, O., Tadonki, C., Lacassagne, L., Sayadi, F. & Ouni, B. (2018). Harris corner detection on a NUMA manycore. *Future Generation Computer Systems*, 88, 442–452.

Harakannanavar, S. S., Prabhushetty, K., Hugar, C., Sheravi, A., Badiger, M. & Patil, P. (2018). IREMD: An Efficient Algorithm for Iris Recognition. *International Journal of Advanced Networking and Applications*, 9(5), 3580–3587.

Henriques, J. F., Caseiro, R., Martins, P. & Batista, J. (2015a). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3), 583–596.

Henriques, J. F., Caseiro, R., Martins, P. & Batista, J. (2015b). High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3), 583–596.

Kaware, S. R. (2018). Google Maps with Visual Positioning System.

Kazemi, V., Burenius, M., Azizpour, H. & Sullivan, J. (2013). Multi-view body part recognition with random forests. *2013 24th British Machine Vision Conference, BMVC 2013; Bristol; United Kingdom; 9 September 2013 through 13 September 2013*.

Keuper, M., Tang, S., Andres, B., Brox, T. & Schiele, B. (2018). Motion segmentation & multiple object tracking by correlation co-clustering. *IEEE transactions on pattern analysis and machine intelligence*.

Kiani Galoogahi, H., Sim, T. & Lucey, S. (2013). Multi-channel correlation filters. *Proceedings of the IEEE international conference on computer vision*, pp. 3072–3079.

Kraemer, S., Bouzouraa, M. E. & Stiller, C. (2017). Simultaneous tracking and shape estimation using a multi-layer laserscanner. *Intelligent Transportation Systems (ITSC), 2017 IEEE 20th International Conference on*, pp. 1–7.

Leigh, J., Yu, O., Schonfeld, D., Ansari, R., He, E., Nayak, A., Ge, J., Krishnaprasad, N., Park, K., Cho, Y.-j. et al. (2001). Adaptive networking for tele-immersion. In *Immersive Projection Technology and Virtual Environments 2001* (pp. 199–208). Springer.

Lu, X., Yuan, D., He, Z. & Li, D. (2017). Sparse selective kernelized correlation filter model for visual object tracking. *Security, Pattern Analysis, and Cybernetics (SPAC), 2017 International Conference on*, pp. 100–105.

Luo, J., Chen, X. & Hu, Y. (2017). A fast circle detection method based on threshold segmentation and validity check for FPC images. *Chinese Automation Congress (CAC), 2017*, pp. 3214–3217.

Maciejewski, M., Piszczek, M., Pomianek, M. & Pałka, N. (2019). Objects tracking in virtual reality applications using SteamVR tracking system: selected issues. *14th Conference on Integrated Optics: Sensors, Sensing Structures, and Methods*, 11204, 74 – 80. doi: 10.1117/12.2537294.

Mun, J. & Kim, J. (2017). 2D template matching for automated inspection using generalized hough transform. *Research and Development (SCOReD), 2017 IEEE 15th Student Conference on*, pp. 461–466.

Nada, D., Bousbia-Salah, M. & Bettayeb, M. (2018). Multi-sensor data fusion for wheelchair position estimation with unscented Kalman Filter. *International Journal of Automation and Computing*, 15(2), 207–217.

Pichai, S. & Kumar, A. (2017). Human skin detection in digital images using multi colour scheme system. *Image Information Processing (ICIIP), 2017 Fourth International Conference on*, pp. 1–6.

Piguet, C. (2018). *Low-power electronics design*. Boca Raton: CRC Press.

Pontecorvo, C. & Redding, N. J. (2017). Non-Periodic Translation Symmetry Detection Using Global Self Similarity. *Digital Image Computing: Techniques and Applications (DICTA), 2017 International Conference on*, pp. 1–8.

Putri, D. C., Christie, D. A. & Musa, P. (2017). Robust ball color auto-calibration for tracking. *Informatics and Computing (ICIC), 2017 Second International Conference on*, pp. 1–5.

Raheem, E. A., Ahmad, S. M. S. & Adnan, W. A. W. (2019). Insight on face liveness detection: A systematic literature review. *International Journal of Electrical & Computer Engineering (2088-8708)*, 9.

Revaud, J., Douze, M., Schmid, C. & Jégou, H. (2013). Event retrieval in large video collections with circulant temporal encoding. *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 2459–2466.

Rosten, E., Porter, R. & Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1), 105–119.

Sarkar, A. & Martin, J. (2019). Dot detection, color classification of dots and counting of color classified dots. Google Patents. US Patent App. 10/235,559.

Sarvaiya, J. N., Patnaik, S. & Bombaywala, S. (2009). Image registration by template matching using normalized cross-correlation. *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on*, pp. 819–822.

Shah, S. S. A., Khalil, M. A., Shah, S. I. & Khan, U. S. (2018). Ball Detection and Tracking Through Image Processing Using Embedded Systems. *2018 IEEE 21st International Multi-Topic Conference (INMIC)*, pp. 1–5.

Wei, S.-D. & Lai, S.-H. (2008). Fast template matching based on normalized cross correlation with adaptive multilevel winner update. *IEEE Transactions on Image Processing*, 17(11), 2227–2235.

Wong, A., Shafiee, M. J., Li, F. & Chwyl, B. (2018). Tiny ssd: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. *2018 15th Conference on Computer and Robot Vision (CRV)*, pp. 95–101.

Xian, X., Jiang, Q. & Liu, X. (2017). Target detection in mine based on fusion of color and edge information. *Advanced Mechatronic Systems (ICAMechS), 2017 International Conference on*, pp. 355–360.

Yang, B., Xiang, D., Yu, F. & Chen, X. (2018). Lung tumor segmentation based on the multi-scale template matching and region growing. *Medical Imaging 2018: Biomedical Applications in Molecular, Structural, and Functional Imaging*, 10578, 645 – 651. doi: 10.1117/12.2293065.

Yao, A., Gall, J. & Van Gool, L. (2010). A hough transform-based voting framework for action recognition. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2061–2068.

You, Y., Wang, Y., Chao, W.-L., Garg, D., Pleiss, G., Hariharan, B., Campbell, M. & Weinberger, K. Q. (2019). Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving. *arXiv preprint arXiv:1906.06310*.

Yu, A.-s., Hara, K., Inoue, K. & Urahama, K. (2017). Corner detection in fisheye images by modified Yin-Yang grid. *Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE*, pp. 3292–3297.

Zhang, S., Wakahara, T. & Yamashita, Y. (2017). Image matching using GPT correlation associated with simplified HOG patterns. *Image Processing Theory, Tools and Applications (IPTA), 2017 Seventh International Conference on*, pp. 1–6.