HAKAN TAŞAN

ATILIM UNIVERSITY 2019

A HYBRID METHOD FOR OBJECT TRACKING IN VIDEO

ATILIM UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

HAKAN TAŞAN

MASTER OF SCIENCE THESIS

DEPARTMENT OF SOFTWARE ENGINEERING

JUNE 2019

A HYBRID METHOD FOR OBJECT TRACKING IN VIDEO

ATILIM UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

HAKAN TAŞAN

MASTER OF SCIENCE THESIS

DEPARTMENT OF SOFTWARE ENGINEERING

JUNE 2019

Approval of the Graduate School of Natural and Applied Sciences, Atılım University.

_____

Prof. Dr. Ali KARA
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of **Master of Science in Software Engineering, Atılım University.**

_____

Prof. Dr. Ali YAZICI
Head of Department

This is to certify that we have read the thesis A HYBRID METHOD FOR OBJECT TRACKING IN VIDEO submitted by HAKAN TAŞAN and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Asst. Prof. Dr. Erhan GÖKÇAY
Supervisor

Examining Committee Members:

Asst. Prof. Dr. Hakan TORA
Aircraft Electrics & Electronic Department, Atılım University _____

Asst. Prof. Dr. Erhan GÖKÇAY
Software Engineering Department, Atılım University _____

Asst. Prof. Dr. Kasım ÖZTOPRAK
Computer Engineering Department, KTO Karatay University _____

Date: 25 June 2019

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Hakan TAŞAN

Signature :

ABSTRACT

A HYBRID METHOD FOR OBJECT TRACKING IN VIDEO

Taşan, Hakan

M. S., Software Engineering Department

Supervisor : Asst. Prof. Dr. Erhan GÖKÇAY

June 2019, 85 pages

Detecting the object in the video and tracking it has been emerging as an important research field in computer vision and image processing. Many algorithms have been developed for object tracking and there are some conditions in which each algorithm is successful or unsuccessful. In this thesis, a robust hybrid system that consisting of three object detection and tracking algorithms is proposed for the purpose of tracking object in video. These algorithms are template matching, color-based histogram and SURF based on feature point. OpenCV library have been used to implement these algorithms in hybrid system. While implementing algorithms, different techniques have been applied such as gaussian blur, color space conversions, Otsu thresholding, sliding window approach, feature extraction and description, and distance measurements. Any object from the video can be selected and the selected object can be traced in the rest of the video. To prevent occlusion of the object and to minimize the effects of sudden movement of scene, refreshing selected object approach is used each fifth frame of the video. Aim of the hybrid system is to improve the detection rate of the object to be tracked in sequence of video frames. All performance tests have been performed on NTU-VOI 2018, Visual Tracker Benchmark 2013, NfS 2017 and Davis 2017 datasets. The test results of the proposed hybrid system have been compared with the results of the three individual detecting and

tracking algorithms. The results show that hybrid system gives the best performance except for processing time for tracking object in video.

Keywords: Object Detection, Object Tracking, Template Matching, Color Histogram, SURF.

GCRIS

ÖZ

VİDEODA NESNE TAKİBİ İÇİN HİBRİT METOT GELİŞTİRMESİ

Taşan, Hakan
Yüksek Lisans, Yazılım Mühendisliği Bölümü
Tez Yöneticisi : Dr. Öğr. Üyesi Erhan GÖKÇAY

Haziran 2019, 85 sayfa

Videodaki nesnenin algılanması ve takibi, bilgisayarla görü ve görüntü işlemede önemli bir araştırma alanı olarak ortaya çıkmıştır. Nesne takibi için birçok algoritma geliştirilmiştir ve her algoritmanın başarılı veya başarısız olduğu bazı koşullar vardır. Bu tezde, videoda nesne takibi amacıyla üç nesne tespiti ve takibi algoritmasından oluşan güçlü bir karma sistem önerilmiştir. Bunlar şablon eşleştirme, renk histogramı ve özellik çıkarımına dayalı SURF algoritmalarıdır. Bu algoritmaları hibrit sistemde uygulamak için OpenCV kütüphanesi kullanılmıştır. Algoritmalar uygulanırken; gaussian blur, renk uzayı dönüşümleri, Otsu eşiklemesi, kayan pencere yaklaşımı, özellik çıkarımı ve betimlemesi, ve uzaklık hesaplamaları gibi farklı teknikler uygulanmıştır. Videodaki herhangi bir nesne seçilebilir ve seçilen nesne videonun geri kalanında takip edilebilir. Nesnenin tıkanmasını önlemek ve sahnenin ani hareketinin etkilerini en aza indirmek için, videonun her beşinci karesinde seçilen nesnenin yenilenmesi yaklaşımı kullanılır. Hibrit sistemin amacı, video karelerindeki takip edilecek nesnenin tespit oranını iyileştirmektir. Tüm performans testleri NTU-VOI 2018, Visual Tracker Benchmark 2013, NfS 2017 ve Davis 2017 veri setleri üzerinde gerçekleştirilmiştir. Önerilen hibrit sistemin test sonuçları, üç ayrı tespit ve takip algoritmasının sonuçlarıyla karşılaştırılmıştır. Sonuçlar, hibrit sistemin video nesne takibi için işlem süresi dışında en iyi performansı verdiğini göstermektedir.

Anahtar Kelimeler: Nesne Tespiti, Nesne Takibi, Şablon Eşleştirme, Renk Histogramı, SURF.

GCRIS

DEDICATION

To My Family

ACKNOWLEDGEMENTS

I wish to Express my thanks and gratitude to my supervisor Asst. Prof. Dr. Erhan Gökçay for his guidance, support, friendship, encouragements and toleration throughout the preparation of this thesis.

I would like to thank my friends and colleagues for giving me new ideas to write my thesis.

I am also thankful to my dear wife Gözde Tarcan for her kind friendship, encouragement and help tremendously throughout my studies in all cases.

Last but not least, I am grateful to my family and beloved ones for being always with me and their support of me no matter what happens.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

SURF	Speed up Robust Features
SIFT	Scale Invariant Features
ORB	Oriented FAST and Rotated BRIEF
FAST	Features from Accelerated Segment Test
BRIEF	Binary Robust Independent Elementary Features
LOG	Laplacian of Gaussian
HSV	Hue Saturation Value
RGB	Red-Green-Blue
BGR	Blue-Green-Red
3D	Three-Dimensional
2D	Two-Dimensional
kNN	K-Nearest Neighbor
NNDR	Nearest Neighbor Distance Ratio
OpenCV	Open Source Computer Vision Library

CHAPTER 1

INTRODUCTION

Computer vision is an area that studies how to understand, interpret and reconstruct a three-dimensional scene from its two-dimensional images presented in the scene. One of the important purpose of computer vision is to make able digital systems to replicate some functions of human being such as vision and perception. Plenty of effort has been consumed on visual object tracking to reach this objective [1-5,7-9,11,15], which is significant and difficult study areas in computer vision. Because it has a broad range of real-world systems including robot vision, video surveillance, authentication systems, traffic monitoring, human-computer interaction and other some applications, it has attracted the attention of researchers. Some applications of them about visual object tracking are listed below.

a) For video surveillance, it may be successfully used for monitoring human being acts in car park, residential areas, and banks [16] or to detect suspicious activities or unlikely events.

b) For traffic flow monitoring [17], highways are monitored via street cameras in some countries. In this way, it may be used to detect traffic accident [18], or pedestrian counting [19].

c) For vehicle navigation, it is used for video-based path planning and obstacle avoidance capabilities.

d) For human-computer interaction, it may be applied to some applications such as gesture and hand recognition [20], eye gaze tracking and also mobile video conferencing [21].

e) For video tracking, it is utilized to automatically detect and track objects in video frames

Object tracking in a video is a challenging process. Despite the fact that object tracking has been researched for years [3-7,10,12-14], it still suffers from difficulties including low-quality frame rate, low resolution, color distortion, small and no detail objects, real-time processing requirements, scene illumination changes, rapid scene or object motions, complex object shapes, full or partial object occlusion (non-overlapping), noise disturbance, pose variation and shape deformation. These challenges may cause object tracking failures.

Basically, video object tracking is the process of checking existence of object and precisely locating of a target object in video sequence using image processing. It is needed an automated system to obtain understandable information while processing videos due to quite impossible of manually handling videos. In this study, a new robust object tracking system has been proposed to track selected object in videos using image processing methods.

If we briefly explain new hybrid system, it consists of a combination of three algorithms used for object tracking from image processing algorithms. That is, system uses the best of the three algorithms that try to find the object effectively in each frame of an input video. These algorithms are Template Matching, Histogram based on the color, and Speeded Up Robust Features (SURF) based on feature points. While implementing each algorithm, some basic steps have been applied to find object. For example, in template matching; sliding window approach, smoothing with gaussian blur, color space conversion and Otsu thresholding have been used. In color-based histogram; still sliding window approach, color space conversion, histogram calculation and normalizing operations have been used. In SURF; feature extraction, feature description and feature matching operations have been used. Finally; in template matching, color-based histogram and SURF, distance measurements have been performed between the selected object and correspondence video frame using Normalized Cross Correlation Coefficient, Correlation and K-Nearest Neighbor and Nearest Neighbor Distance Ratio, respectively. The flowchart of the new system developed can be seen in Figure 5-1. Detail information about algorithms and used methods will be explained in Chapter 4 and 5.

The novelty of hybrid system is that the user determines the object that she/he wants to follow in the first frame of the video, so nobody does not know the tracking object before object is selected by user and there is no need to train objects. The object to be tracked may be stationary or moving object such as car, bird, human, tree, or bag. Moreover, in case of sudden movement of the object and the possibility of obstruction, the selected object is assumed that as if it were selected for the first time in every 5 framed. Then, object tracking process continues in the same way by renewing the selected object every 5 frames.

A number of efforts have been made toward performance evaluation of three algorithms and hybrid system separately to compare with each other. The individual performance of each algorithm and the performance of hybrid system have been tested using random sample videos from different datasets published in [22,24-26] and only one test video from YouTube.

This paper is organized as follows; we have done a literature survey in next chapter. In Chapter 3, basic information has been given for image processing operations used in algorithms in order to better understand the working principle of them. In Chapter 4, three algorithms used in hybrid system has been explained briefly. In Chapter 5, we propose a new hybrid system for object tracking in video. Experimental results obtained from object tracking algorithms and hybrid system separately has been given and then discussed in Chapter 6 and 7, respectively.

CHAPTER 2

LITERATURE SURVEY

Fazli et. al [27] present a new system for efficient object tracking that combines Scale Invariant Feature Transform (SIFT) and color features using color histogram. SIFT algorithm is used for target representation and localization. In this study, an image is transformed into local feature vectors. They also extract image feature from the target region to create a color histogram.

Wang, Yiwei et al. [14] introduce a new method for tracking moving objects in video frames. In each frame in the video, it does this by separating moving objects from the background. For this, they have developed a rule based method to track the objects based on the four variables which are the grayscale distributions, sizes, positions, and presence of textures of the objects. Their results indicate that their method gets success in some conditions like possible collisions or object stops.

Kim et al. [28] introduce a combined shape and feature-based non-rigid object tracking algorithm. It has block matching, feature extraction, and motion detection which robust to object's rapid movement and their features. In their work, a set of features called shape control points are generated by detecting edges in the neighboring four directions.

Lipton et al. [29] describes an end-to-end algorithm to extract object from a video. Firstly, according to image-based properties, objects are classified into predefined categories. After that, tracking process is performed. Once objects are classified, objects are tracked by a combination of template matching and temporal differencing algorithms. Thus, objects are robustly identified in spite of partial occlusions and ambiguous poses, and background clutter is effectively rejected.

In [30], a general framework using template matching for object tracking in video images is presented. Their algorithm allows to track in real time systems efficiently. The position of the target template in the first image should be known. Actual position of the template in subsequent images are obtained by comparing the grey level values of the target template with the grey level values of the predicted region.

In study [31], character recognition system on the ID card is explained. Their recognition process included into four stages; pre-processing, text-area extraction, segmentation and recognition. Template matching algorithm is used to recognize the text on ID card and the experiment shows a good result which can achieve 92.74% of accuracy.

In paper [32] proposes a novel template matching algorithm called algebraic template matching. Algorithm efficiently calculates similarities between the partial images of the input image of widths and heights and template. First, template image is approximated by proposed algorithm. After that, this polynomial algorithm is used to match the input image instead of the template image. According to the authors, there is no algorithm that calculates similarities incrementally like the proposed algorithm by changing the widths and heights of the partial image.

In [33], a new feature selection method using template matching is presented. The introduced method selects features based on the upper bound of the template matching error and then tracking process is continued with that features using template matching. Their experimental results show that feature selection method gives better performance than that of other feature detectors.

Prabhakar et al. [34], a combination of frame differencing and template matching algorithm is presented. Object is detected by frame differencing, then detected object is tracked by efficient template matching algorithm. In this study, the templates used for matching purpose are generated dynamically which ensures that the change in orientation and position of object does not hinder the tracking system.

To detect objects by comparing their color histograms, color indexing technique is introduced by Swain and Ballard [35,36]. Histogram intersection (HI) method has been used to calculate the rate of matching between histograms.

In [12], a new method is presented for extracting objects moving independently of the background from a video sequence taken by a moving camera. First, they extract and track feature points through the video sequence using Kanade-Lucas-Tomasi algorithm, and select the trajectories of background points by exploiting geometric constraints. Then, they generate a panoramic image of the background and compare it with the individual frames.

In some researches [37-43], overview and methodologies are discussed about template matching algorithm.

Wenjing Jia et al. [44] introduces and compares six histogram based image matching methods which has the conventional histogram intersection (HI) method, Wong and Cheung's merged palette histogram matching (MPHM) method, Jia et al.'s Gaussian weighted histogram intersection (GWHI) method, the refined GWHI, Gevers' color ratio gradient (CRG) method, and Jia et al.'s color edge co-occurrence histogram (CECH) method.

In [45], both color and edge histograms are proposed as ways to detect and track objects in video. Also, two histogram-based methods are evaluated according to their speed, robustness and accuracy. Objects are tracked by dividing each video frame into cells and comparing the histograms computed for the cells of the current frame with corresponding cells of the scene.

In [46], an adaptive algorithm is proposed based on color histogram back projection. It facilitates the tracking process when objects are multiple in video. Also, for demonstrating multi object tracking, CLICK-IT which is an interactive multiple objects tracking system has been implemented.

Six histogram-based search methods have been researched and compared with each other in both HSV and RGB color spaces by Sangog Jeong [47]. Then, six histogram-based image retrieval methods are compared by providing recall and precision graphs for each test images. In his study, he stated that HSV color space gives better performance than RGB color space.

P. M. Panchal et al. [48] presents and evaluates the SIFT and SURF algorithms. Their experimental results show that SIFT detects more features than SURF. However, speed of the SIFT is slower than SURF. Moreover, in another paper [49], an evaluation between the three feature detector and descriptor methods which are SIFT, PCA-SIFT and SURF is performed. They point out that SURF is fast and has good performance as the same as SIFT.

Geng Du et al. [50] propose to use SURF method for face recognition. They also have done detail comparisons of SURF and SIFT.

In some researches [51-53], detailed explanation is given about SURF algorithm. Furthermore, in [54,55] papers, object tracking applications based on SURF features method are presented.

CHAPTER 3

PRELIMINARY FOR PROPOSED SYSTEM

With image processing functions and libraries, various operations can be performed on images or video sequences including color space conversion, blurring, resizing, thresholding, cropping, arithmetic and other some matrix operations. The basic information about these functions and libraries used in our study is given to better understand the working principle of algorithms and hybrid system.

3.1 Image Representation in Digital

As mentioned above, it is difficult task to transfer human motion perception to computer environment as a whole. Because in machine vision, a computer receives the image or video frame as a grid of numbers. In Figure 3-1, human being sees an apple but a computer sees just a grid of numbers.

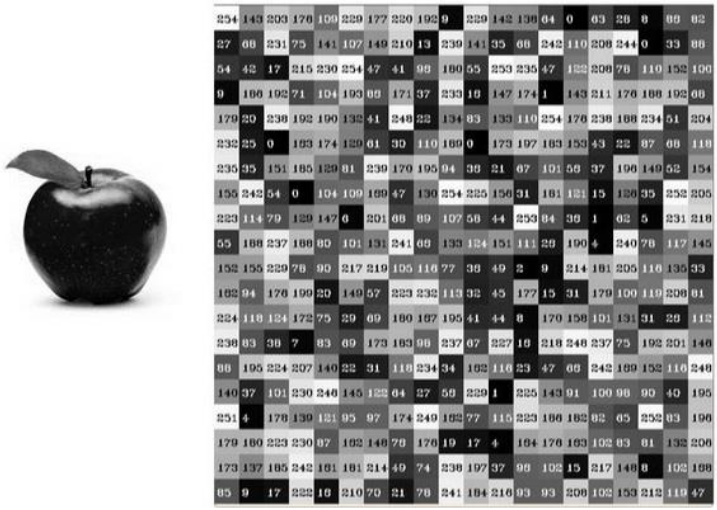


Figure 3-1 Human (left) vs. Computer (right) Perception

An image is a two-dimensional numerical arrays called matrix. Each matrix contains image elements which are called pixel, and each pixel has a specific value and location in the matrix. A sample image matrix can be seen in Equation 3-1. M refers to the number of rows and the image's height, as well as N refers to the number of columns and the image's width. This also represents the resolution of the image.

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \dots & a_{M-1,N-1} \end{bmatrix} \tag{3-1}$$

Moreover, an image is a function as a form of $f(x, y)$, where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x, y) is called the intensity of the image at that point [23]. The coordinate values at the origin are represented as $(x, y) = (0,0)$. Equation 3-2 denotes a matrix form of an image.

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0, N-1 \\ f(1,0) & f(1,1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1, N-1) \end{bmatrix} \tag{3-2}$$

If we talk about the video, it consists of images, each of which is called a video frame, and image processing methods can be applied to the video frames separately.

3.1.1 Pixel Relationships

3.1.1.1 Pixel Neighbors

A pixel at point $f(x, y)$ has a total of 8 neighbors. The neighbors' coordinates, two of which are vertical, two of which are horizontal and four of which are diagonal, are given by:

$$f(x,y)=\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & f(x-1,y-1) & f(x-1,y) & f(x-1,y+1) & \cdots \\ \cdots & f(x,y-1) & f(x,y) & f(x,y+1) & \cdots \\ \cdots & f(x+1,y-1) & f(x+1,y) & f(x+1,y+1) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \quad (3\text{-}3)$$

3.1.1.2 Arithmetic Operations on Images

Images are represented in the form of matrices as in Equation 3-1 and 3-2. Assume that $f(x,y)$ and $h(x,y)$ are two images to be processed. Arithmetic operations on these images are defined between all pairs of corresponding pixels in $f(x,y)$ and $h(x,y)$. For example, image subtraction and image addition between $f(x,y)$ and $h(x,y)$ are expressed as in Equation 3-4 and Equation 3-5, respectively.

$$g(x,y) = f(x,y) - h(x,y) \quad (3\text{-}4)$$

$$g(x,y) = f(x,y) + h(x,y) \quad (3\text{-}5)$$

where in Equation 3-4, $g(x,y)$ is computed by subtracting pixel in f from the pixel in h. And also in Equation 3-5, $g(x,y)$ is computed by adding pixel in f to the pixel in h.

One of the most benefit of image subtraction in the area is the detection of differences between two images. For example, moving objects between the first and second frames can be detected with image subtraction.

3.2 OpenCV

In this thesis, OpenCV (version 2.4.13.6-vc14), which is an Open Source library for computer vision, is used. It includes over 500 functions implementing computer vision methods such as detection, image pyramids, segmentation, transforms, feature matching, camera calibration, machine learning, matrix math and a variety of artificial intelligence. It also has BSD license that is free for academic and commercial use [56].

3.3 Color Spaces

A color space is a way of describing colors in digital systems [57] and in the literature some color spaces have been commonly used for tasks such as object detection and tracking, image segmentation [27,35,36,44-46]. Although a computer defines a color using the amounts of red, green, and blue phosphor emission required to match a color, human being define a color by its attributes of hue, saturation (brightness), and value (colorfulness) [36]. In general, three color spaces are widely used in modern day computer vision: RGB or BGR, HSV and Grayscale. In hybrid system, BGR, Grayscale and HSV color spaces are used so now in next three sections we will briefly explain these color spaces to understand tracking algorithms.

3.3.1 RGB or BGR

RGB is the acronym for Red (R), Green (G), and Blue (B) colors. It is the most widely used color space until now. The idea is that red, green and blue are the three basic colors that can be mixed with each other in various proportions to create all other colors. It may be illustrated by a cube with the three axes that are perpendicular to one another. This color space is showed in Figure 3-2. Any color can be easily defined by giving its red, green and blue values between 0 and 255 in the color cube.

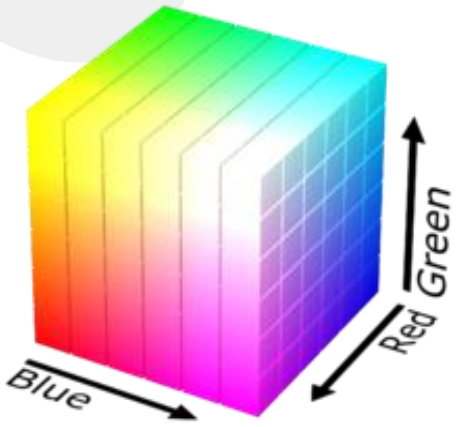


Figure 3-2 RGB or BGR Color Cube

Some common RGB color triplets are listed below:

- White for (255, 255, 255)
- Black for (0, 0, 0)
- Red for (255, 0, 0)
- Green for (0, 255, 0)
- Blue for (0, 0, 255)

3.3.2 Grayscale

Grayscale images are represented by a single intensity value per pixel. The pixel value can be between 0 and 255, and it indicates different tones of gray, that is actually brightness.

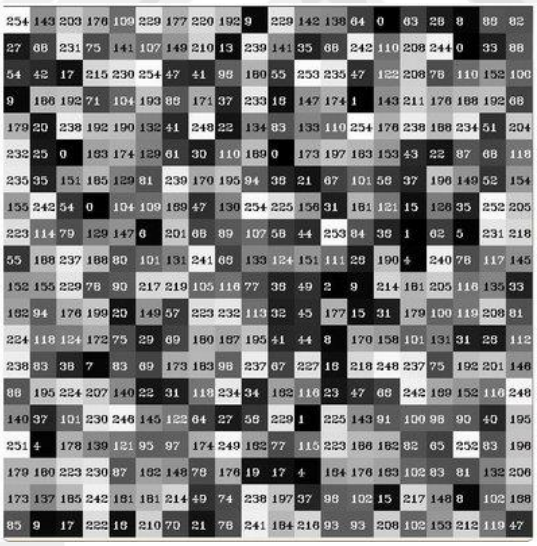


Figure 3-3 An Example of Grayscale Image and Pixel Values

3.3.3 HSV

The three dimensional HSV color space has been described by Alvy Ray Smith in 1978 [58], and it refers to Hue (H), Saturation (S) and Value (V). It is one of the most suitable color spaces for color recognition and also related to the way of human eyes perceive color [59]. Because, unlike RGB, the brightness value in HSV color space may be expressed

separately, so it is possible to reach the real color from H and S values, no matter how much brightness or shade in the picture. This model may be illustrated by a cone shape as Figure 3-4.

Hue represents the color by the circular angle of the cone and is valued between 0° and 360° (red, green, blue) in OpenCV. Red, green and blue primary starts with at 0°-360°, 120° and 240°, respectively. So there are red values both at 0° and 360°.

Saturation represents the richness of the color by the radius of the cone and varies from 0 to 1 in OpenCV. As the saturation grows, the color appears in more vivid color tones, but as the saturation falls, the color approaches gray tones.

Value represents the brightness of the color by the height of the cone. That is ratio of white in image and varies from 0 to 1 in OpenCV. From 0 to 1, the brightness increases.

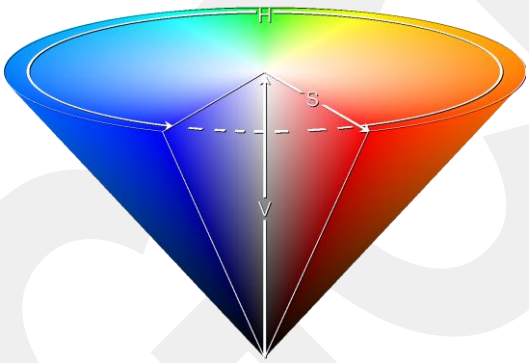


Figure 3-4 HSV Cone

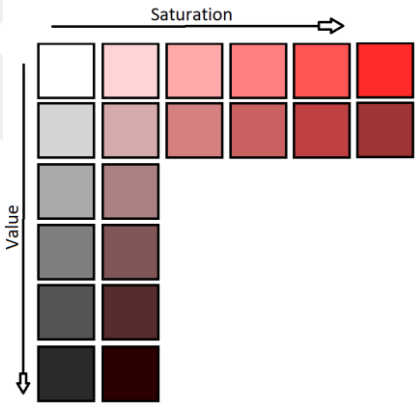


Figure 3-5 Saturation and Value

3.4 OpenCV and Color Spaces

OpenCV provides more than 150 color space conversion types available. Because two of the conversion types are used in hybrid system, only two of them, which are most widely used ones, will be looked into, BGR ↔ Grayscale and BGR ↔ HSV (details in Chapter 4).

Although RGB often refers to default color format, OpenCV stores color internally as a 24-bit 3D BGR image (the bytes are reversed) [60], in which each pixel in the matrix is represented by a three-element array as given below. Each three-element describes a color.

$$\begin{bmatrix} [240, 194, 146] & [238, 192, 144] & [238, 189, 143] \\ [238, 189, 143] & [241, 189, 143] & [241, 189, 143] \\ [240, 191, 143] & [241, 193, 145] & [242, 194, 146] \end{bmatrix} \quad (3\text{-}6)$$

3.5 Sliding Window Technique

In image processing, the sliding window technique is a rectangular region of fixed width and height that slides over the given source image. The window image $W[x, y]$ and the source image $S[x, y]$ are shown to have pixel location on the Figure 3-6. The window image slides from the upper left corner to the right in x-direction as shown in Figure 3-7 over each pixel in the source image. Window image slides down in y-direction when the right edge of window image reaches the right edge of source image. Then, an operation like comparison may be performed when window slides, between the window image and patch of under the source image and window image.

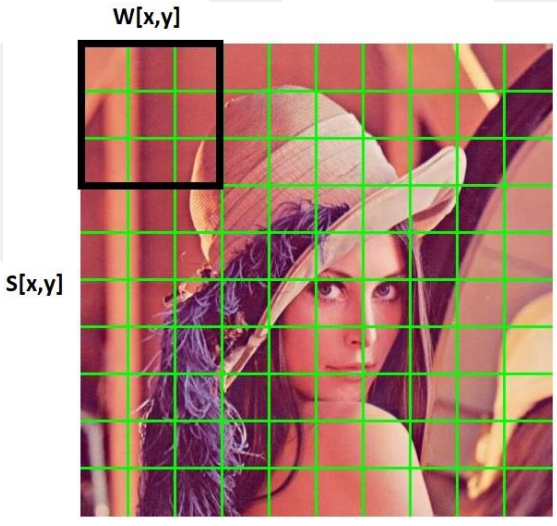


Figure 3-6 Sliding Window Technique

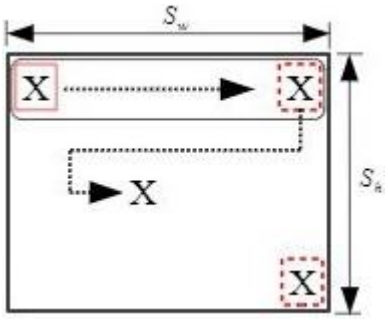


Figure 3-7 Sliding Window Moving Direction

CHAPTER 4

ALGORITHMS

In this chapter, detail information about algorithms and methods used in hybrid system are given. They are Template Matching, Color-based histogram and SURF.

4.1 Template Matching

One of the classical and widely used approach in digital image processing for object detection is template matching. Basically, it is a method used to find the location of a template image in a source image. It has been provided by OpenCV as *matchTemplate()* method and can be implemented to hybrid system easily.

A sliding window technique is used to search maximum possible match in template matching. As can be seen in Figure 4-7 and according to the OpenCV standards [66], basic algorithm steps have been developed for tracking of object in video using template matching. The basic algorithm steps can be seen in Figure 5-3 (b).

This algorithm suite well for single object tracking and widely used when single object is to be tracked [5]. Also, tracking process is much easier for static object than dynamic object. Sensitivity to noise, make it not good for real-time demanding conditions. This method consumes less memory. It is usually not strong against changes on the image such as light, rotation and fast motion of object [29].

4.1.1 Smoothing by GaussianBlur

In image processing, smoothing operation which is called blurring is one of the simplest and frequently used pre-processing operation prior to processing of image. It is generally used to improve the quality of the image by reducing the noise of the image. In here, noise means that is represented as an unwanted information (pixels) or unneeded small details within the video image frames. Noise from the image can be eliminated by GaussianBlur method among smoothing operations.

In hybrid system, applying *GaussianBlur()* method provided by OpenCV is the first step of the template matching algorithm. It does smoothing operation by sliding a Gaussian kernel, which is basically a matrix that represents how should be combined a window of n-by-n pixels to get a single pixel value throughout the source image array. That is, in similar to convolution process in Figure 4-2, each pixel value in the source image array is computed based on the value of the Gaussian kernel. According to the OpenCV standards [61,62], *GaussianBlur()* method and parameters have been applied in hybrid system in template matching algorithm.

Table 4-1 GaussianBlur Method in OpenCV

Method	GaussianBlur(Mat sourceImage, Mat outputImage, Size kernelSize, double sigmaX, double sigmaY)
Parameter	**Description**
sourceImage	source image in Mat array
outputImage	output image in Mat array
kernelSize	Gaussian kernel size with width and height
sigmaX	Gaussian kernel standard deviation in x-direction
sigmaY	Gaussian kernel standard deviation in y-direction

In this method, the size of the Gaussian kernel, which must be both positive and odd, are specified. It is need to very careful in choosing the size of the Gaussian kernel because

when it is too small, noise of the image may not able be to removed. However, when it is too large, small details of the image may be removed. Moreover, the value of Gaussian kernel standard deviation in x- and y-direction should be chosen carefully. sigmaY is defined as the same as sigmaX when only sigmaX is defined. They are computed from the Gaussian kernel size when both are given as zeros [62]. $3x3$, $5x5$, and $7x7$ Gaussian kernels with the "standard" σ (i.e., sigmaX = 0) gives higher performance and speed than other Gaussian kernels and σ in OpenCV implementation of GaussianBlur [61]. In this case, in hybrid system, a $7x7$ Gaussian kernel size and sigmaX = 0 have been preferred. That is, σ value will be computed by Gaussian kernel. It can be said that the results of this method are inaccurate when the image is not smoothed with optimum values. According to the above references, method steps must be:

1) Find σ values
2) Create Gaussian kernel matrix with values
3) Apply convolving to image array

In first step, σ value can be found with the following formula:

$$\sigma = 0.3\left(\frac{n}{2} - 1\right) + 0.8 \; where \; n = \begin{matrix} ksize.width \; for \; horizontal \; kernel \\ ksize.height \; for \; vertical \; kernel \end{matrix} \quad (4\text{-}1)$$

In hybrid system, Gaussian kernel size width and height are for 7, σ value are:

$$\sigma = 0.3\,(7/2 - 1) + 0.8 \quad (4\text{-}2)$$
$$\sigma = 0.3\,(2.5) + 0.8 \quad (4\text{-}3)$$
$$\sigma = 0.75 + 0.8 \quad (4\text{-}4)$$
$$\sigma = 1.55 \quad (4\text{-}5)$$

Then, a Gaussian kernel matrix can be created and each value in the matrix is calculated with the given formula:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (4\text{-}6)$$

where x and y are the coordinates of the pixel of the Gaussian kernel, and σ is the Gaussian kernel standard deviation. In here, we assume that the indices of this array span the range -3:3 along each axis, and the center of the Gaussian kernel is at position $[0,0]$. An example of a Gaussian kernel matrix with $\sigma = 1.55$ yielded by the above expression can be seen in Figure 4-1.

[i,j]	-3	-2	-1	0	1	2	3
-3	.011	.039	.082	.105	.082	.039	.011
-2	.039	.135	.287	.368	.287	.135	.039
-1	.082	.287	.606	.779	.606	.287	.082
0	.105	.368	.779	1.000	.779	.368	.105
1	.082	.287	.606	.779	.606	.287	.082
2	.039	.135	.287	.368	.287	.135	.039
3	.011	.039	.082	.105	.082	.039	.011

Figure 4-1 7x7 Kernel Matrix

After a Gaussian kernel has been calculated, convolution operation is performed by sliding window technique across the source image with Gaussian kernel. The general expression of a convolution is:

$$h[i,j] = f[i,j] * g[i,j] = \sum_{k=1}^{n} \sum_{l=1}^{m} f[k,l] g[i-k, j-l] \quad (4\text{-}7)$$

where $g[i,j]$ is referred to as source image, $f[i,j]$ is the Gaussian kernel, $h[i,j]$ is the output image, and the kernel has m rows and n columns.

As an example of convolution, in Figure 4-2, the right matrix represents the Gaussian kernel with size $3x3$, and the left matrix represents the source image array. $h[2,2]$ or I_{35} of the output image, which is the central element, is a weighted combination of all the

entries of the $3x3$ image matrix in source image array, with weights given by the Gaussian kernel.

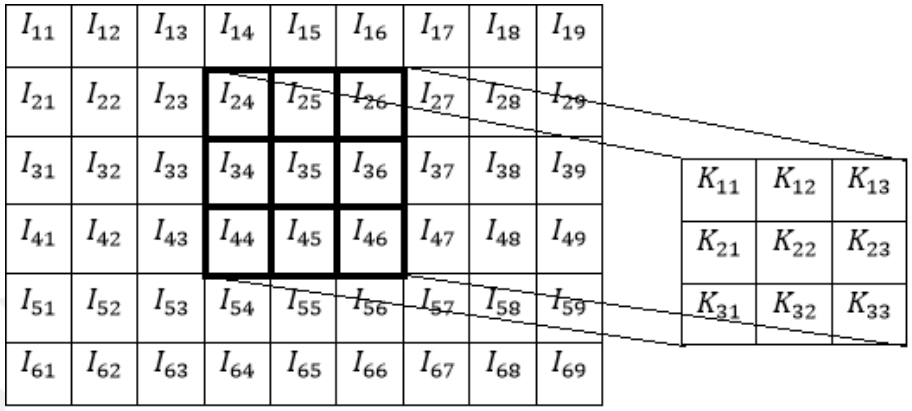


Figure 4-2 Convolution Process

$$h[2,2] = K_{11} * I_{24} + K_{12} * I_{25} + K_{13} * I_{26} + K_{21} * I_{34} + K_{22} * I_{35} + K_{23} * I_{36} + K_{31} * I_{44} + K_{32} * I_{45} + K_{33} * I_{46} \quad (4\text{-}8)$$

Figure 4-3 GaussianBlur example: input image on the left, output image on the right

4.1.2 Converting BGR to Grayscale Color Space

Thresholding operation is generally performed on grayscale images. Before the apply thresholding operation, it is needed to convert the template and the source to grayscale.

After that, thresholding operation can be used [63]. The source image can be converted from BGR to Grayscale color space using *cvtColor()* method.

Table 4-2 cvtColor Method for Grayscale in OpenCV

Method	cvtColor(Mat sourceImage, Mat outputImage, int codeType)
Parameter	**Description**
sourceImage	input image in Mat array
outputImage	output image in Mat array
codeType	color space conversion type (COLOR_BGR2GRAY)

The grayscale image is obtained by the weighted average of the pixel values of the red, green and blue color planes with approximately %30, %59 and %11, respectively. Color planes are between 0 and 1, and they are determined by sensitivity of the human eye to color. Higher weightage is given to green plane because of the most sensitive color to human eyes [76]. The grayscale image is calculated using the following formula:

$$RGB\ or\ BGR\ to\ Gray: Y = (0.299) * R + (0.587) * G + (0.114) * B \quad (4\text{-}9)$$

4.1.3 Thresholding

Thresholding is the most useful image segmentation method that creates a binary image from a grayscale image [64].

Table 4-3 Thresholding Method in OpenCV

Method	threshold(Mat sourceImage, Mat outputImage, double thresholdValue, double maxValue, int thresholdType)
Parameter	**Description**
sourceImage	input image in Mat array
outputImage	output image in Mat array
thresholdValue	threshold value
maxValue	maximum value to use with the THRESH_BINARY thresholding types
thresholdType	threshold types

The *threshold()* function handles only grayscale source images and in above section, all pixels in input image were converted to grayscale color space. This function applies identified threshold between the 0-255 to each array pixel. It is typically used to filter pixels with very large or very small values. A threshold value is defined manually. Then, a comparison of each pixel value is performed with respect to a threshold. Pixels greater than this are set to one value of 255 (white), pixels less than to another value of 0 (black).

There are several types of threshold provided by the OpenCV function including THRESH_TOZERO_INV, THRESH_TOZERO, THRESH_TRUNC, THRESH_BINARY_INV, and THRESH_BINARY. Also, THRESH_OTSU as a special threshold type can be combined with one of the above types. In hybrid system, it has been used THRESH_BINARY and THRESH_OTSU types. THRESH_BINARY is determined by type as:

$$dst(x,y) = \begin{cases} maxval & if\ src(x,y) > thresh \\ 0 & otherwise \end{cases} \quad (4\text{-}10)$$

where $src(x,y)$ is the source pixel, the threshold value is $thresh$, the destination pixel $dst(x,y)$ may be set to 0 or $maxval$ according to the thresh value.

While a threshold can be decided manually, it is better to use the image data to compute one. In hybrid system, since the selected object is unknown, it cannot be known whether the selected threshold value is good or not. To solve this problem, Otsu [65] suggested a method which determines the optimal threshold value and uses it instead of the specified threshold value. For threshold value, we simply pass zero in *threshold()* method for *thresholdValue* parameter. That is, a threshold value (t) automatically can be computed by Otsu's algorithm given by the relation below:

$$\sigma_\omega^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \tag{4-11}$$

where

$$q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_2(t) = \sum_{i=t+1}^{I} P(i) \tag{4-12}$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{i\,P(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{i\,P(i)}{q_2(t)} \tag{4-13}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{q_2(t)} \tag{4-14}$$

An example of RGB, grayscale and Otsu threshold image can be seen in Figure 4-4, Figure 4-5 and Figure 4-6. The RGB image given in Figure 4-4 is converted to a grayscale image. Otsu thresholding is applied on grayscale image given in Figure 4-5. In Figure 4-6, the Otsu thresholded result image is obtained.

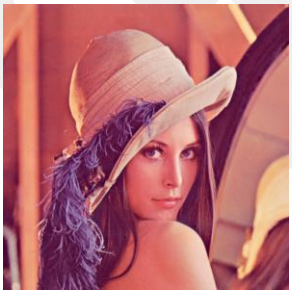


Figure 4-4 BGR Lenna

Figure 4-5 Grayscale Lenna

Figure 4-6 Otsu Thresholded Lenna

In Figure 4-7, it can be seen how noise filtering makes better the image. For a noisy image; in first, the value of 127 has been applied as a threshold. In second, Otsu thresholding has been directly applied. In third, Gaussian kernel has been applied first, then Otsu thresholding has been applied.

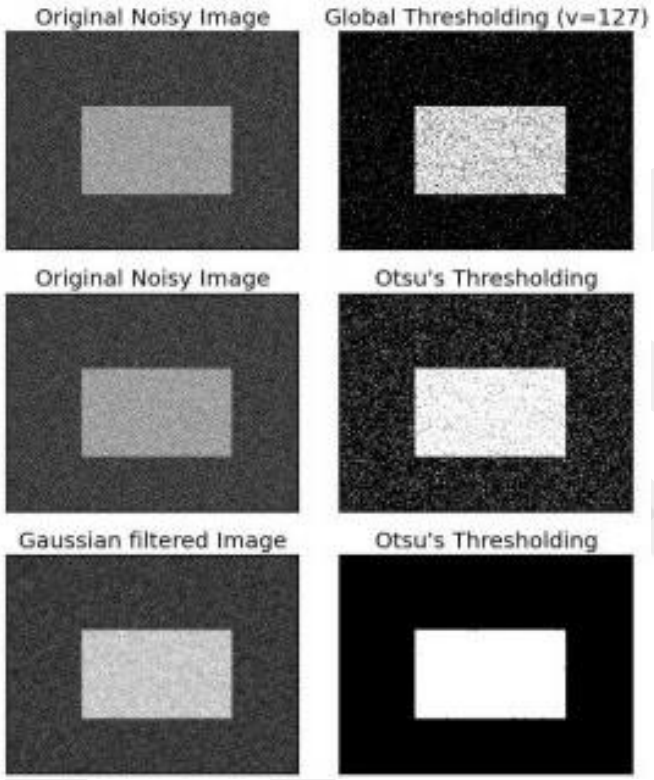


Figure 4-7 Comparison of Global Thresholding and Otsu Thresholding [66]

4.1.4 Matching Process

Our frame has been smoothed and grayscaled, now that we can match the frame and selected object. In this classic template matching method, the distance metric between the target and the template is measured. In image processing, Cross Correlation Coefficient is a measure of the distance of two images where the images are of different sizes. By sliding the template image over the source image, distance between the two images is measured.

Table 4-4 matchTemplate Method in OpenCV

Method	matchTemplate(Mat sourceImage, Mat templateImage, Mat result, int method)
Parameter	**Description**
sourceImage	input image in Mat array
templateImage	template image in Mat array
result	comparison result image in Mat array
method	matching comparison types such as TM_SQDIFF, TM_SQDIFF_NORMED, TM_CCORR, TM_CCORR_NORMED, TM_CCOEFF, TM_CCOEFF_NORMED

matchTemplate() function slides throughout the source image to compare with the template image. In hybrid system, it has been used *TM_CCOEFF_NORMED* method and the Normalized Cross Correlation Coefficient can be calculated in this way:

$$R(x,y) = \frac{\sum_{x',y'} (T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \quad (4\text{-}15)$$

The matching gives a measure of the degree of distance between the source and the template image. After *matchTemplate()* function finishes the comparison, the best match and its location can be found as global maximum value using the *minMaxLoc()* function. The computed maximum value is placed in *max_val* and addresses/points also given by *max_loc*. max_*val* takes value between 0 and 1 and value close to 1 indicates the highest match [67].

4.2 Color-Based Histogram

A color-based histogram is an illustration of the color distribution (e.g. red, green, blue) of an image as a graph or plot. Firstly, color-based histograms have been introduced by Swain and Ballard [68] and then generalized by Schiele and Crowley [69].

Despite the fact that color has been used in earlier work as an identification feature, it has been neglected recently [70, 71]. Lack of good algorithms for color tracking may be the one reason for this. However, it can be efficiently used for tracking object due to largely independent of view and resolution, and being an identifying feature that is local [73]. For this reason, a color-based histogram method has been used in hybrid system for object tracking in video. Our goal is to determine the location of a selected object in the video. As with template matching, the color-based histogram also uses the sliding window technique for detecting object. Firstly, color histogram of selected object has been calculated. Then, in each video frame, color histogram has been calculated at each iteration of sliding. The histograms have been matched to detect selected object. Figure 4-8 shows a sample of color histogram in HSV color space from our application. The Hue histogram in the upper left corner, the Saturation histogram in the upper right corner, and the Value histogram in the lower left corner are shown.

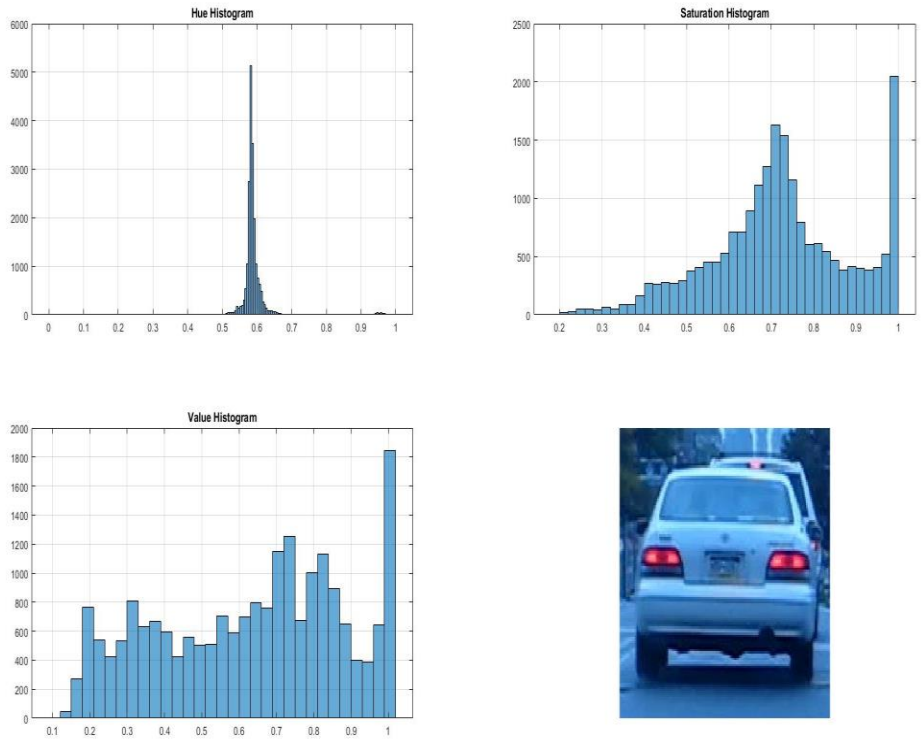


Figure 4-8 Sample Color Histogram

According to the OpenCV standards [66], basic algorithm steps we have just developed for tracking of object in video using color-based histogram may be summarized as in Figure 5-3 (c).

This method is not widely used because of its huge calculation time and consuming large memory. it is very cumbersome for real time tracking. However, because it does not need the exact separation of the object from its background, this method produces accurate movement of the objects.

4.2.1 Converting BGR to HSV Color Space

HSV color space means that observing the color change by separating the color into a hue, saturation and value, and they can be distinguished separately [73, 74] as illustrated in Figure 3-4. Since color is defined in more detail in HSV color space, it has become the another widely used color space in object tracking. So this process has been done in HSV color space.

Firstly, the source image need to be converted from BGR to HSV color space using the *cvtColor()* method in OpenCV. Mathematical calculations are denoted below.

Table 4-5 cvtColor Method for HSV in OpenCV

Method	cvtColor(Mat sourceImage, Mat outputImage, int code)
Parameter	**Description**
sourceImage	input image in Mat array
outputImage	output image in Mat array
code	color space conversion type (COLOR_BGR2HSV)

$$V \leftarrow max(R, G, B) \tag{4-16}$$

$$S \leftarrow \begin{cases} \frac{V - min(R,G,B)}{V} & if(V \neq 0) \\ 0 & otherwise \end{cases} \tag{4-17}$$

$$H \leftarrow \begin{cases} 60(G-B)/(V-min(R,G,B)) & if\ (V=R) \\ 120+60(B-R)/(V-min(R,G,B)) & if(V=G) \\ 240+60(R-G)/(V-min(R,G,B)) & if(V=B) \end{cases} \quad (4\text{-}18)$$

where $0 \leq H \leq 360$, $0 \leq S \leq 1$, and $0 \leq V \leq 1$.

4.2.2 Histogram Calculation

In this thesis, two-dimensional histograms as Hue and Saturation have been calculated. Because as it can be seen that in Figure 3-4, the color may only be defined with H and S channels. V channel is related with brightness of the color, not the color itself. Also, Histogram Comparison examples in OpenCV documentation [75], the V channel is not considered.

Before method explaining, let's identify some parts of the color histogram to understand it easily.

- dims, channels: indicates dimensions of the histogram such as Hue, Saturation
- bins: number of subdivisions in each dim.
- range: the range for the values to be measured.

In OpenCV, calculation of histogram is performed by *calcHist()* method. It enables you to compute a color histogram from images. However, while explaining the application in Chapter 5, since the HSV color space representation would be more understandable rather than the histogram representation, the histogram representation of the selected object and video frames have not been used.

Table 4-6 calcHist Method in OpenCV

Method	calcHist(List<Mat> sourceImage, MatOfInt dimensions, Mat mask, Mat outputHistogram, MatOfInt bins, MatOfFloat ranges, boolean accumulate)
Parameter	**Description**
sourceImage	input image in Mat array
dimensions	dimensions, channels. In our application, we need to process both Hue and Saturation, so the channels would be [0,1]
mask	optional mask
outputHistogram	output histogram
bins	number of bins in each channel. In our application, it has been used 50 bins and 60 bins for Hue and Saturation, respectively
ranges	ranges of each pixel values. In our application, we know that Hue will have a range between 0-180 and Saturation will have a range between 0-256
accumulate	accumulation flag

4.2.3 Normalize Color Histogram

Before comparing the two image histograms, we are going to normalize the two color histogram matrices. For normalization, the range of values of an array is normalized using the *normalize()* method.

Table 4-7 normalize Method in OpenCV

Method	normalize(Mat sourceHistogram, Mat outputHistogram, double alpha, double beta, int normType)
Parameter	**Description**
sourceHistogram	color histogram input
outputHistogram	color histogram output
alpha	minimum pixel value
beta	maximum pixel value
normType	normalization type

In hybrid system, NORM_MINMAX normalization type has been used. Because of Saturation's range is 0-255, we set alpha (minimum) as 0 and beta (maximum) as 255. The minimum pixel value of histogram output will be mapped to the minimum output value (alpha), and the maximum pixel value of histogram output will be mapped to the maximum output value (beta). That is, all values of histogram are between 0 and 255.

$$mindst(I) = alpha, maxdst(l) = beta \quad (4\text{-}19)$$

4.2.4 Histogram Comparison

In color histogram comparison, the most important point is how to measure the similarity between histograms. *compareHist()* method is provided by OpenCV to perform a comparison and there are many different distance metrics for measuring similarity between color histograms such as Correlation, Chi-Square, Intersection and Bhattacharyya. In this thesis, to compare histograms between the current video frame and selected object, “Correlation” distance metric has been implemented to the application. For the Correlation method, the score is between 0 and 1, and the higher the score, better the match [75].

Table 4-8 compareHist Method in OpenCV

Method	compareHist(Mat histogram1, Mat histogram2, int method)
Parameter	**Description**
histogram1	first histogram
histogram2	second histogram
method	comparison method

The following equation shows the *compareHist()* method with Correlation.

$$d(H_1, H_2) = \frac{\sum_I (H_1(I) - \overline{H_1})(H_2(I) - \overline{H_2})}{\sqrt{\sum_I (H_1(I) - \overline{H_1})^2 \sum_I (H_2(I) - \overline{H_2})^2}} \quad (4\text{-}20)$$

where

$$\overline{H_k} = \frac{1}{N} \sum_j H_k(J) \quad (4\text{-}21)$$

and N is a total number of histogram bins, $H1$ as a histogram1 and $H2$ as a histogram2.

4.3 Speeded-Up Robust Features

In this chapter, Speeded Up Robust Features which is called as SURF algorithm used for video tracking based on feature points has been simply explained. Object tracking based on feature points is the increasingly used tracking technique because of their robustness in identifying the objects. A feature is a unique, characteristic or easily recognizable point in the image that can be detected in a robust manner. Features are special because it is no matter how the image scales whether the image rotates, shrinks/expands or subject to distortion. Corners, high-density areas, edges and blobs (an area of an image that greatly differs from its surrounding areas) are good features that can be easily tracked and compared.

There are several algorithms that can be used to detect and extract features such as BRIEF, FAST, Shi-Tomasi Corner Detection, SIFT, SURF, ORB, and Harris Corner Detection. Each of them has its strength sides. For example, Harris and Shi-Tomasi are useful to detect corners, SIFT, SURF and BRIEF are useful to detect blobs.

SURF algorithm has been implemented to hybrid system to track objects in video. SURF was first presented by H. Bay et al. [53] at the 2006 European Conference on Computer Vision. SURF is a rotation and scale invariant technique that means able to detect objects at any angle and scale.

According to the OpenCV standards [23], in general, SURF algorithm can be divided into three main steps as feature detection, feature description and feature matching. OpenCV provides a rich set of functions for all sorts of feature detection, descriptor extraction and descriptor matching. The advantage of this is that features and descriptors around those features can be extracted using only one method without having extra knowledge. Nevertheless, it will be useful to know the background of the algorithm briefly.

4.3.1 Feature Detection

Features are detected from distinguishing location of the object such as corners, blobs, and edges. Choosing the right features plays an important role because the selection result has a significant impact on the performance of the object tracking. SURF uses a very basic Hessian Matrix approximation for feature detection due to its good performance in accuracy and computation time. When a coordinate in an image S is given as $P = [x, y]$, the Hessian matrix $H(x, \sigma)$ in P at scale σ can be defined as:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{yx}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \quad (4\text{-}22)$$

where $L_{xx}(x,\sigma)$, $L_{xy}(x,\sigma)$ and $L_{yy}(x,\sigma)$ are the convolution result of the second order Gaussian derivative $\frac{\partial^2}{\partial x^2}g(\sigma)$, $\frac{\partial^2}{\partial x \partial y}g(\sigma)$, $\frac{\partial^2}{\partial y^2}g(\sigma)$, respectively. These derivatives are also known as LoG (Laplacian of Gaussian). That is, $L_{xx}(x,\sigma)$ represents the second order Gaussian derivative with respect to image coordinate at P [77]. Where the approximated determinant of the Hessian Matrix is maximum represents the features in the image coordinate at P [53]. Determinant of the Hessian Matrix which can be approximated with the following equation:

$$\det(H_{approx}) = D_{xx}D_{yy} - \omega(wD_{xy})^2 \quad (4\text{-}23)$$

where w usually kept as 0.9 and σ kept as 1.2 (for lowest scale) [53]. Using different smoothing values of σ, features can be detected in all scales.

4.3.2 Feature Descriptor

SURF uses Haar-wavelet approach to build feature descriptor. It defines a distribution of the pixel intensities within the feature neighborhood. In order to be rotation invariant, Haar-wavelet responses shown in Figure 4-9 are calculated in x- and y-direction within a circular neighborhood of radius 6s around the feature, with s the scale at which the feature was detected [53]. Then, the dominant orientation is calculated by summing of all responses within a sliding orientation window of angle 60 degrees [53]. The area in which the sum of the horizontal and vertical responses is maximum is chosen as the dominant orientation.

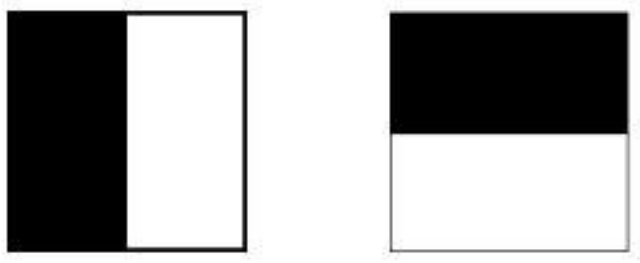


Figure 4-9 Haar-wavelet filters to compute the responses in x-direction (left) and y-direction (right)

After that, a square region of size 20s, where s is the scale factor at which the feature was identified, is built aligned to the chosen orientation. The region is divided into smaller $4x4$ square sub-regions. A few simple features are computed for each sub-region (shown in Figure 4-10). That is, the surf descriptor for every key point is 64 dimensional. It can be said that feature descriptors are coded forms of features. Descriptors also have to be robust to noise.

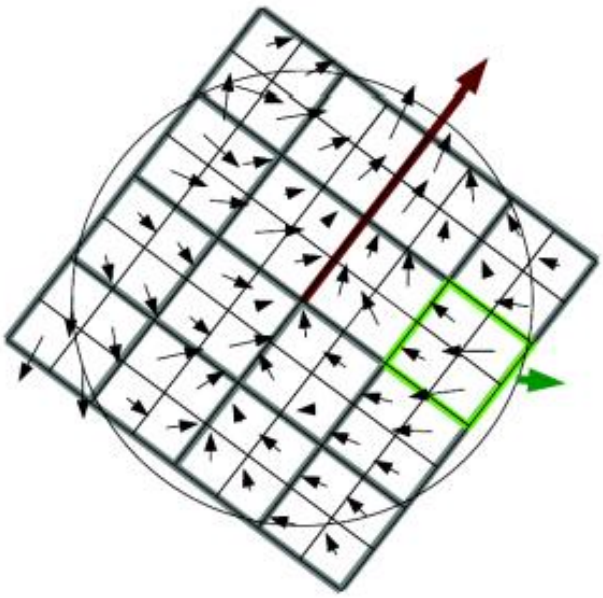


Figure 4-10 64 dimensional key point. The wavelet responses are computed for each square.

4.3.3 Descriptor Matching

Finally, the feature descriptors are matched between images. In our application, matching process is based on the kNN (K-Nearest Neighbor) distance method and the distance is calculated between all potential matching pairs (selected object descriptors and video frame descriptors). The basic idea of kNN is to search for closest match between descriptors. Since we assume that k = 2, the kNN search gives us 2 best matches for each query descriptor and each match has a distance associated with it. What is important is how we decide which of all these matches are good matches after all. For this, a Nearest Neighbor Distance Ratio, which is a simple statistical method proposed by Lowe [78], is used to reduce mismatches within all matching pairs as like Equation 4-24.

For every query descriptor, if the following equation is provided

$$match1.distance \leq match2.distance * NNDR \quad (4\text{-}24)$$

match1 is a good match and add to good match list, otherwise discard both match1 and match2 as false matches. Lowe proposed that matches in which the distance ratio is greater than 0.8, which eliminates 90% of the false matches while discarding less than 5% of the correct matches [78] and Snavely et al. [79] lower the ratio to 0.6. So, it has been used as 0.6 in hybrid system. As mentioned above, each match has a distance and the range of values it takes is between 0 and 1. The average of the good matches is calculated to find the matching score.

$$Matching\ Score = \frac{Sum\ of\ Distance\ in\ Good\ Matches}{\#\ of\ Good\ Matches} \quad (4\text{-}25)$$

In OpenCV, it is very easy to implement the feature point technique to hybrid system. For this, all of the above-described processing steps are performed automatically when using the following methods. *FeatureDetector.detect()* method takes an image and detects features with its location, orientation etc. *DescriptorExtractor.compute()* method takes features as input and gives descriptor for the corresponding feature. Then, to find matches features, k-nearest neighbors as *DescriptorMatcher.knnMatch()* methods can be used.

CHAPTER 5

PROPOSED SYSTEM

At this point we have all of the basics to develop and understand hybrid system. In this chapter, object tracking system in video has been proposed to resolve some of the deficiencies of each algorithm that arise when tracking objects. Our object tracking application is based on three algorithms that Template Matching, Color-Based Histogram and SURF, mentioned above in details. The main strength of our application consists of combining and using these algorithms. As it can be seen the test results in Chapter 6, hybrid system improves the tracking accuracy and decreases the tracking failures while object tracking in video. Now, we will explain how these algorithms have been implemented to our application. Figure 5-1 demonstrates the base flowchart of our application.

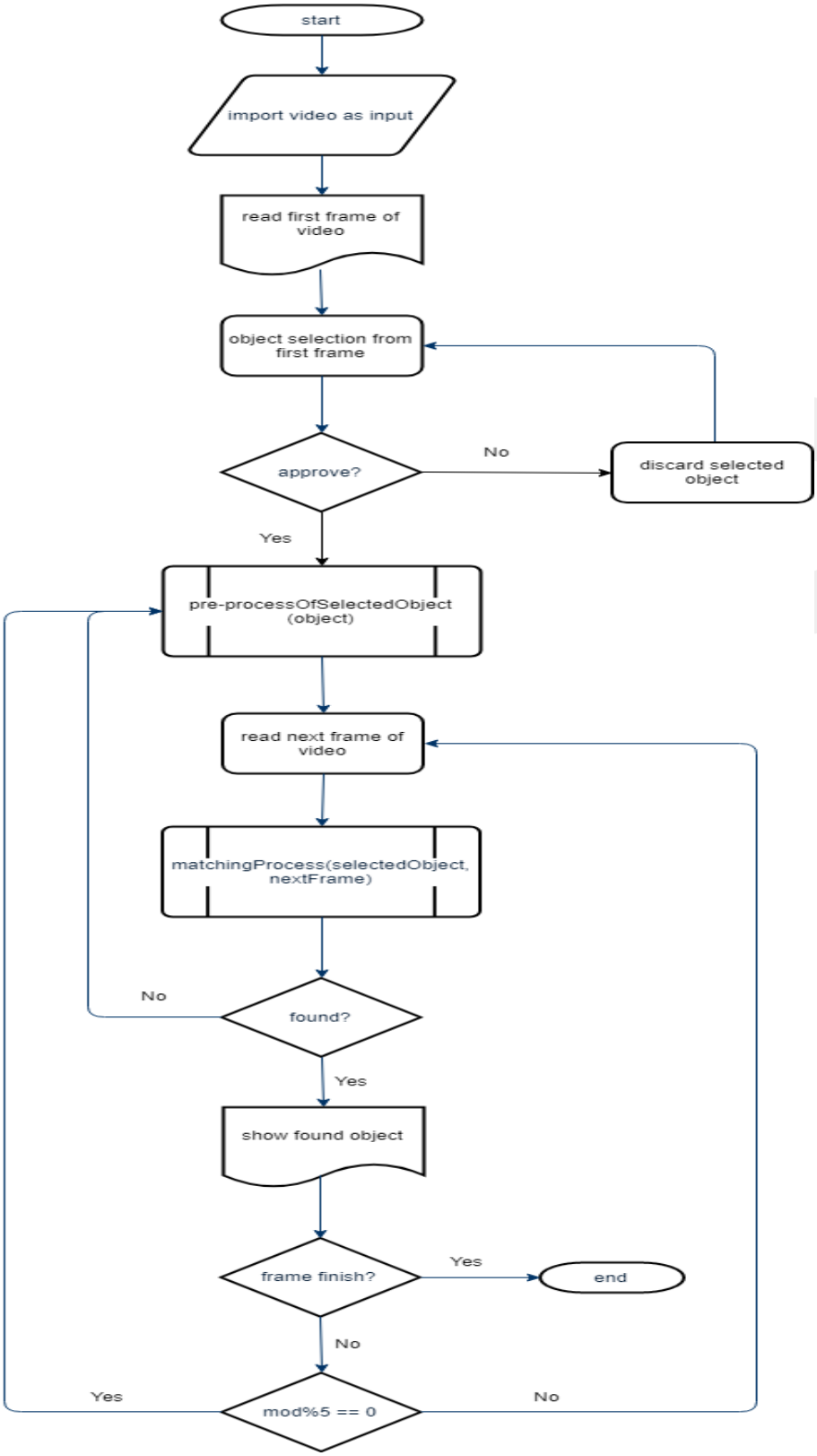


Figure 5-1 Base Flowchart of Our System

According to the base flowchart, firstly, the video with the object to be tracked is imported. The application accepts only mpeg, mpg, mp4, avi, wmv and mkv video extensions. Then, user selects the object to track. Figure 5-2 illustrate the object selection. The target objects to be tracked can be any object. For changeable scenes with various types of tracking objects, and since the object to be selected is unknown, we do not use ground-truth image data or data training.

(a) *(b)*

(c)

Figure 5-2 Object selection steps. (a) First frame of video. (b) Object selection with rectangle. (c) Selected object.

Next, as shown in the Figure 5-3, the preprocesses are performed for each algorithm related to the selected object. For template matching; Gaussian blurring, color space conversion from BGR to Grayscale and thresholding using Otsu operations are applied, respectively. For color-based histogram; color space conversion is applied from BGR to HSV. H-S histogram is calculated and then normalize process is performed to calculated

histogram. For SURF; features of selected object are extracted and descripted. All operations are explained in detail in Chapter 4.

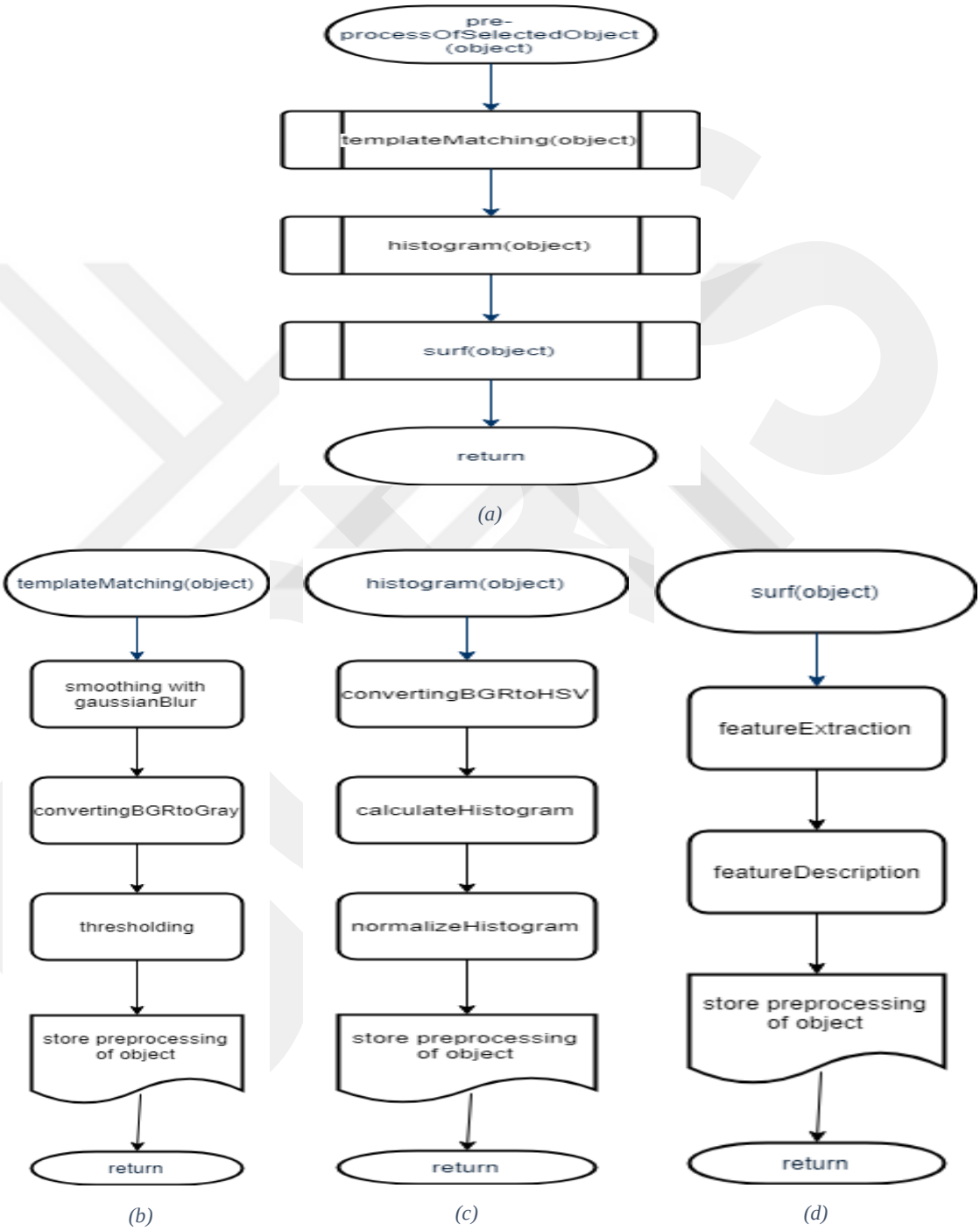
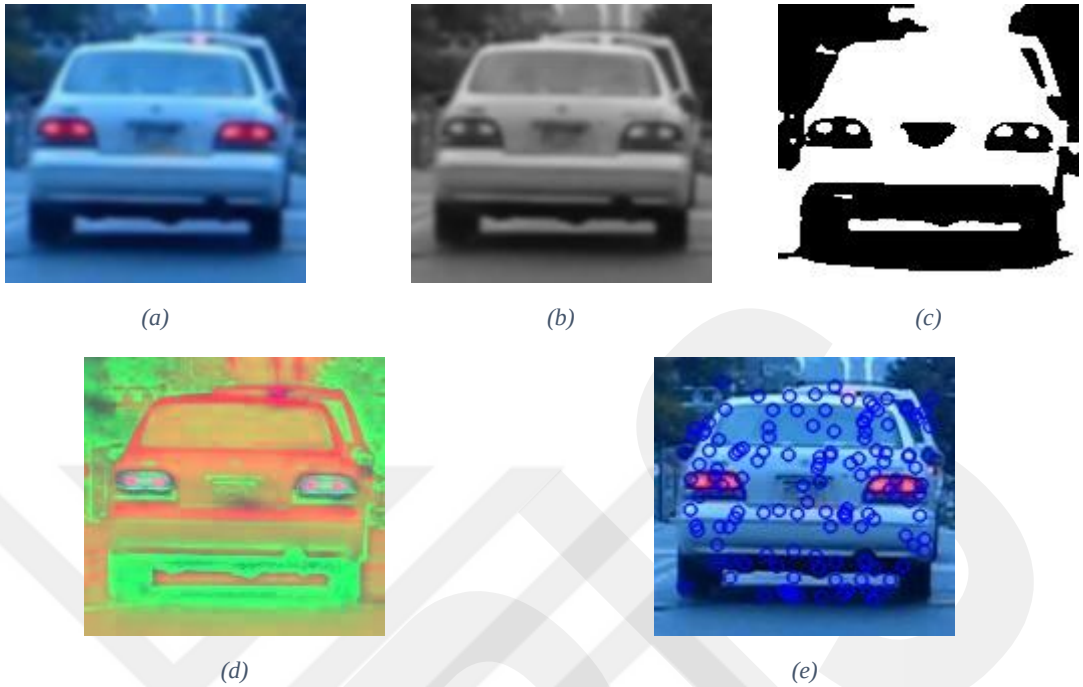


Figure 5-3 (a) Base flowchart of preprocesses operations for selected object. (b) Template matching operations for selected object. (c) Histogram operations for selected object. (d) SURF operation for selected object.



(a) *(b)* *(c)*

(d) *(e)*

Figure 5-4 Obtained preprocessing object for each algorithm. (a) Gaussian blurring of selected object with ksize 7. (b) Grayscale of selected object. (c) Thresholding using Otsu of selected object to be matched. (d) HSV of selected object to be matched with the current frame. (e) Feature points of selected object to be matched with the current frame.

Secondly, video frames begin to play sequentially. Our goal is to find the best similarity ratio among the three algorithms between the selected object and each video frame. That is, system uses the best of the three algorithms that try to find the object effectively in each frame of an input video. To find the similarity, all three algorithms are executed according to the explained in Chapter 4 in order to find the selected object on the current video frame.

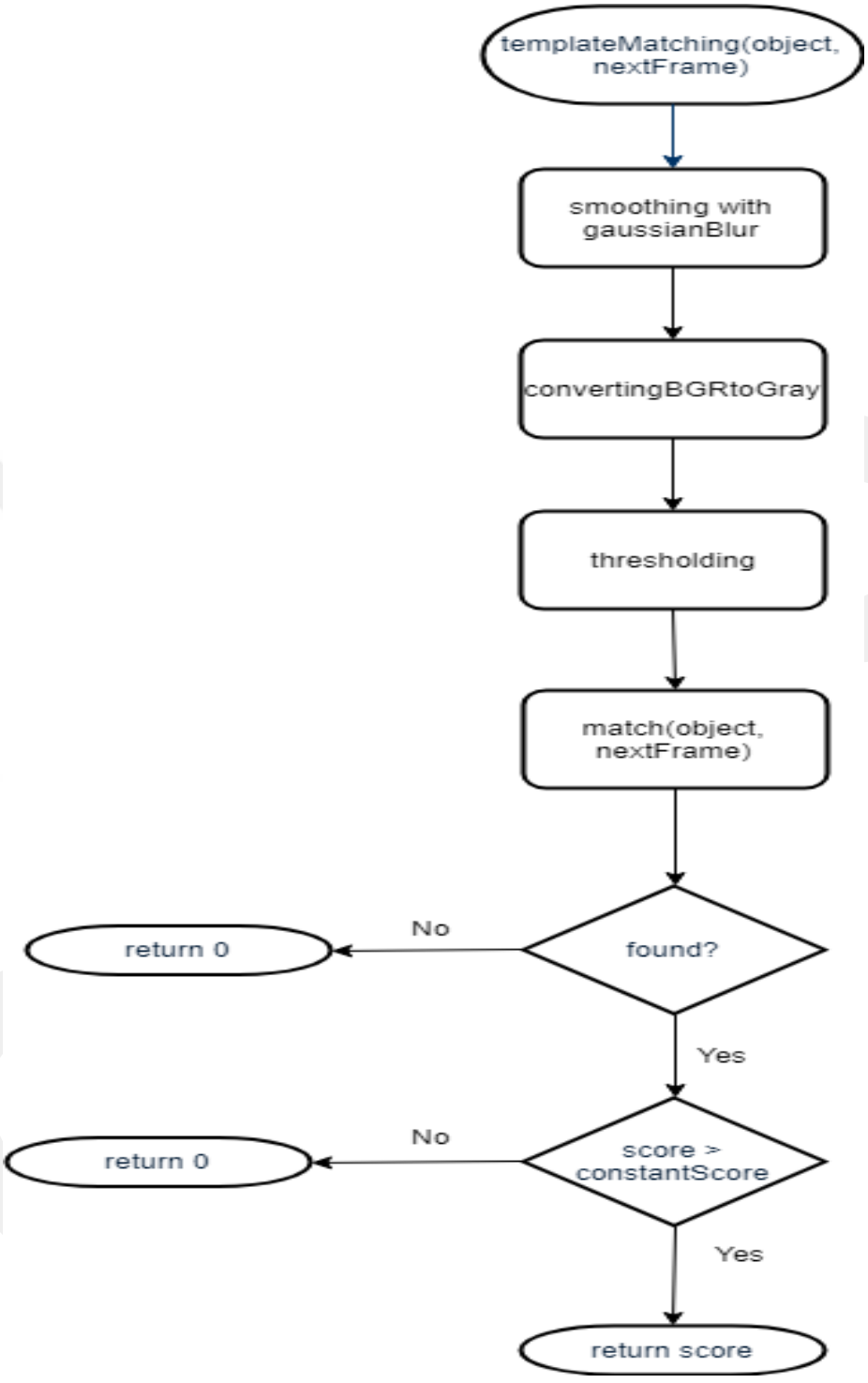


Figure 5-5 Find Similarity with Template Matching

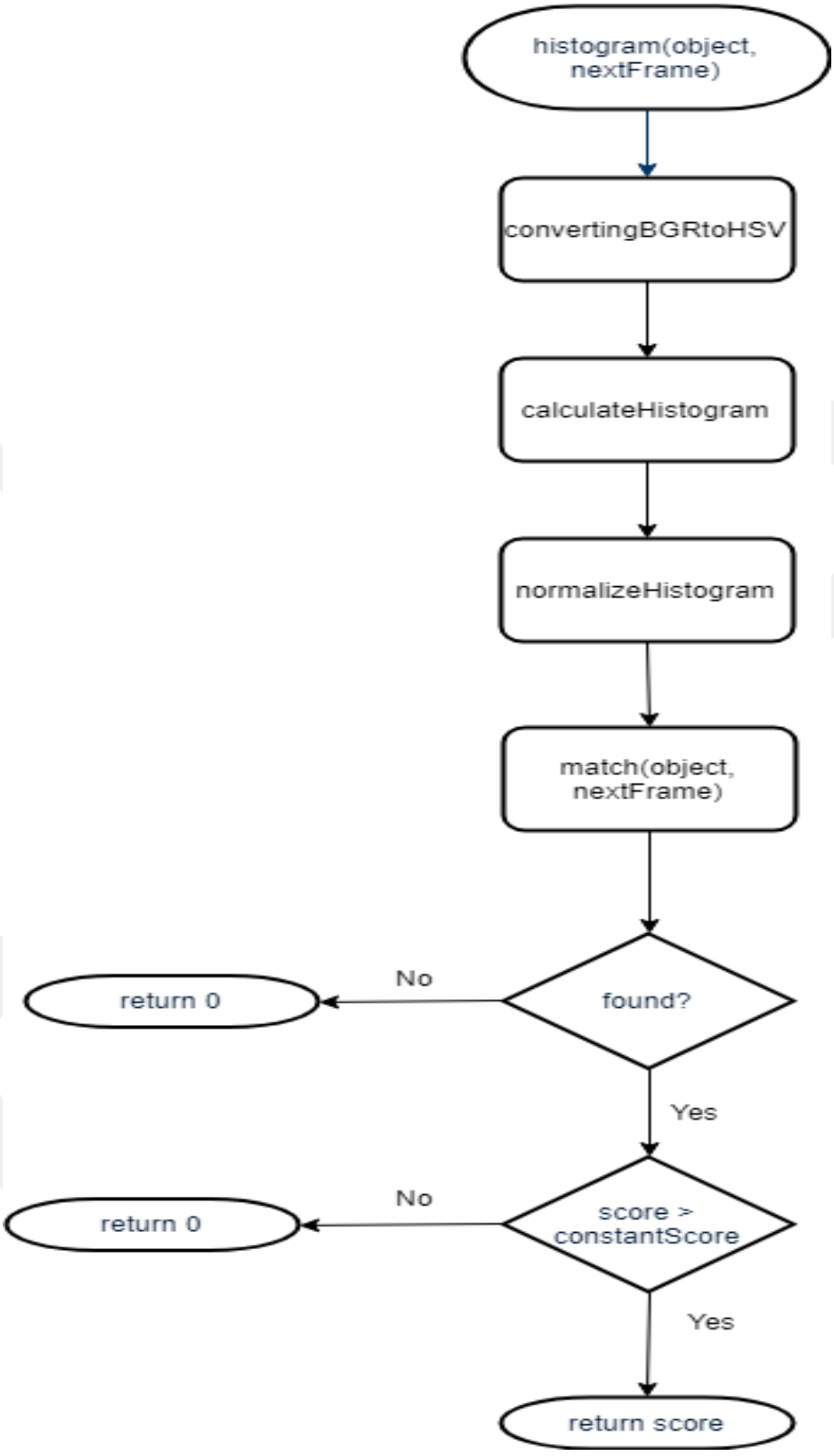


Figure 5-6 Find Similarity with Histogram

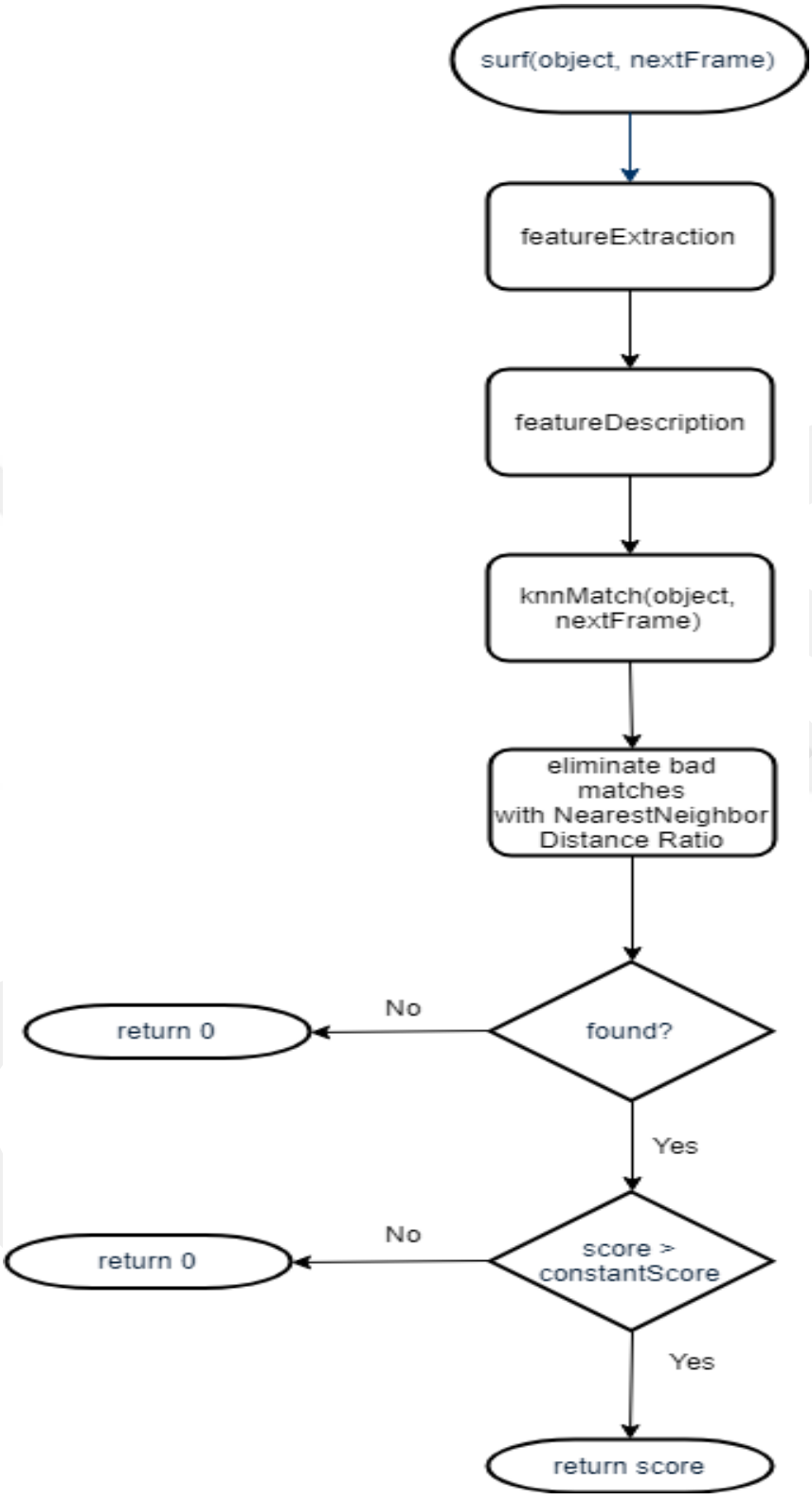


Figure 5-7 Find Similarity with SURF

Figure 5-8 Test results of the BlueCar2 video frames #21, #106 and #186, respectively. (a) Hybrid system results. (b) Template matching results. (c) Color histogram results. (d) SURF results.

According to the flowcharts in Figure 5-5, 5-6 and 5-7, for each algorithm that is executed, the percentage of similarities is calculated if the object is detected. We also explained in details in Chapter 4 how to find distance for each algorithm. The algorithm with the highest similarity percentage indicates that the selected object is more similar to the object

on the current frame. In Figure 5-9, comparing process can be seen between the three algorithm's score. In order to obtain more accurate results, we assume that we ignore the matches with a similarity rate below 70 percent. Figure 5-10 shows the all operations and results on the current frame.

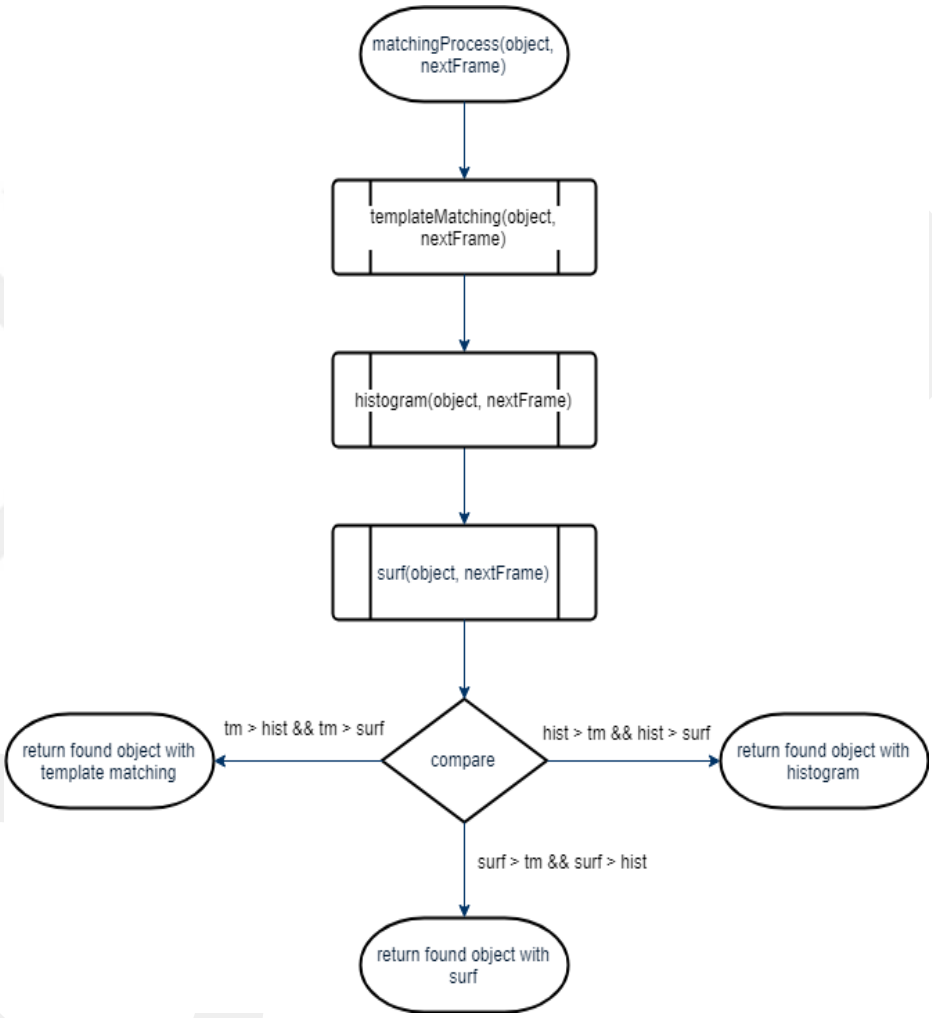


Figure 5-9 Comparing algorithm's scores

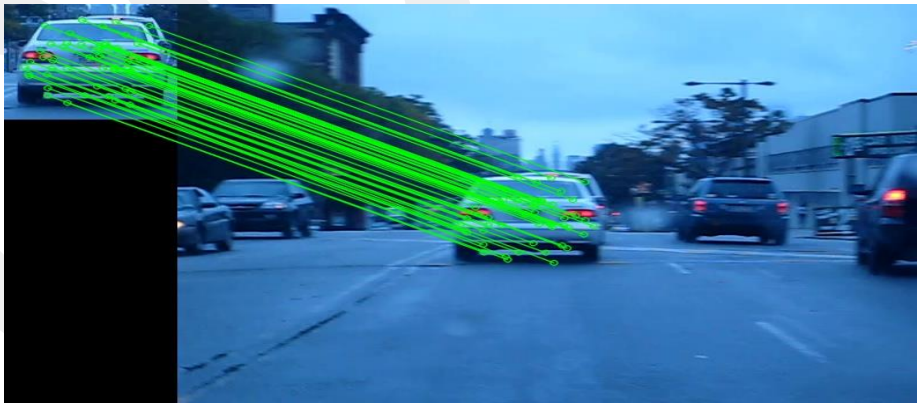
(a)

(b)

(c)

(d)

(e)



(f)

Figure 5-10 (a) Gaussian blurring of current frame with ksize 7. (b) Grayscale of current frame. (c) Thresholding using Otsu of current frame to be matched with the selected object. (d) Template Matching Similarity Ratio 0.4972170889377594 and Matched Object (Since the similarity rate is below 70 percent, we assume that there is no match). (e) Color-Based Histogram Similarity Ratio 0.9827497348219524 and Matched Object. (f) SURF Similarity Ratio 0.8941978530994817 and Matched Object.

If the object is found, it is represented by a green rectangular box using the algorithm with the highest similarity ratio.

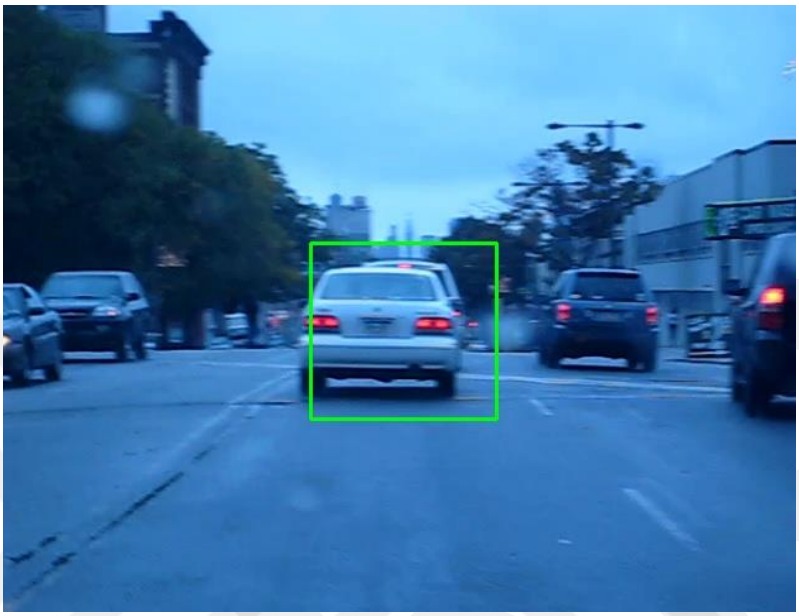
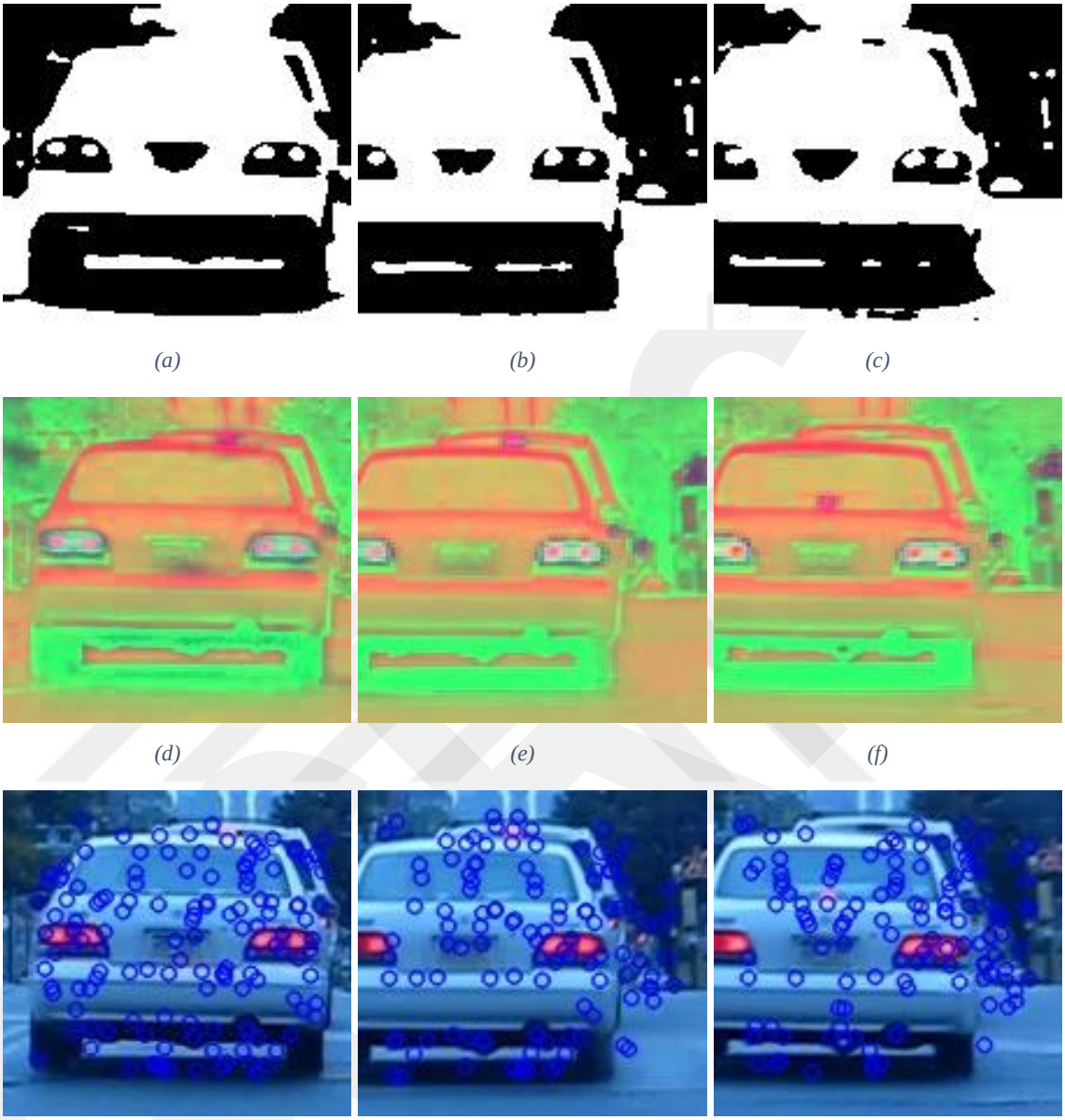


Figure 5-11 Object representation of best matched (Color-Based Histogram for sample)

Finally, in case of sudden movement of the object and the possibility of obstruction, we accept the selected object as if it were selected for the first time in every 5 framed. Then, object tracking process continues in the same way by renewing the selected object every 5 frames. For example, in Figure 5-12; the first, 5th and 10th frames are shown, and object found by algorithm in those frames, the area shown as a rectangle in that frame, will be considered as the new object.



(a) *(b)* *(c)*

(d) *(e)* *(f)*

(g) *(h)* *(i)*

Figure 5-12 (a) Selected object from first frame in Template Matching. (b)(c) Detected object from 5. and 10. frame in Template Matching, respectively. (d) Selected object from first frame in Color-Based Histogram. (e)(f) Detected object from 5. and 10. frame in Color-Based Histogram, respectively. (g) Selected object from first frame in SURF. (h)(i) Detected object from 5. and 10. frame in SURF.

CHAPTER 6

EXPERIMENTAL RESULTS

Tracking accuracy and tracking failures have been tested toward performance evaluation of three algorithms and our system separately on the 12 sample videos chosen randomly from different datasets published in [22-26] and only one test video from YouTube. Avidemux 2.7.2 as a free software program has been used to edit the videos. Experiments have been performed including various objects such as moving and stationary toys, logo over the car, moving car, speed skater and book. OpenCV libraries have been used for implementation of image processing algorithms. Also, the application has been coded in Java language in Eclipse IDE on the AMD Phenom™ II X4 955 Processor 3.20 GHz, 8,00 GB x64 System.

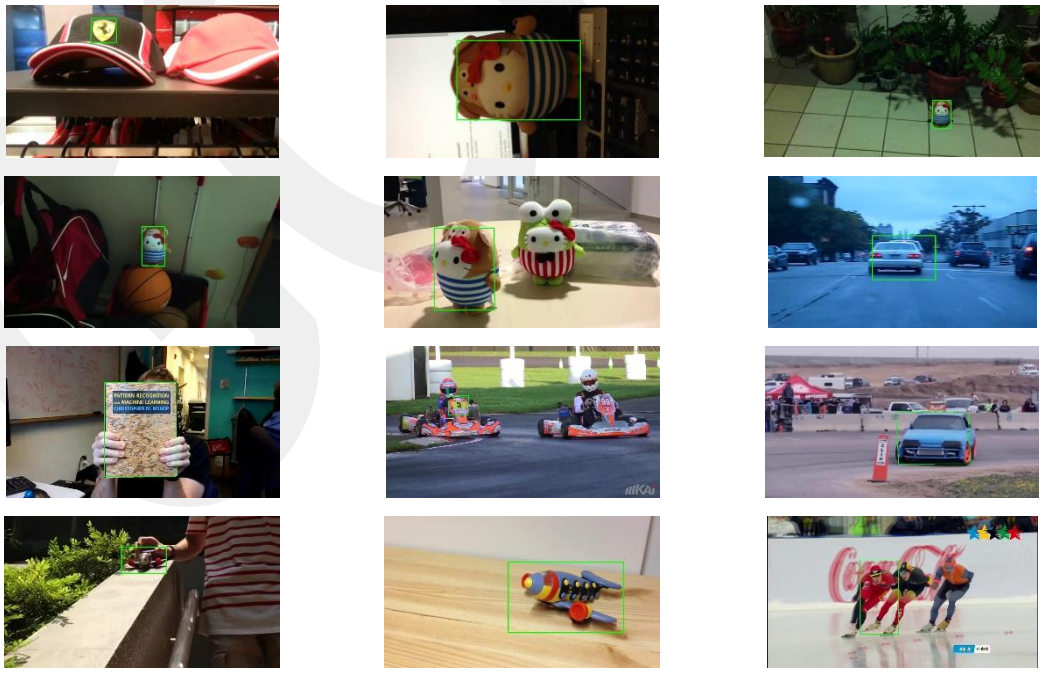


Figure 6-1 Test Videos from Datasets

We have defined a metric for success characterizing the Detection Rate of the system. Starting with the first frame of the video, Detection Rate is computed for each frame of the video. True and false detection are computed for each frame of the video.

- *TD (True Detection):* detected object that correspond to selected object. The region showing at least 50 percent of the detected object is assumed as true.
- *FD (False Detection)*: detected object that do not correspond to selected object. The region showing less than 50 percent of the detected object or any region that do not detected object are assumed as false.

$$Detection\ Rate = \frac{TD}{TD+FD\ (\#\ of\ Frames)} \quad (5\text{-}1)$$

The individual performance of each algorithm and the performance of hybrid system have been given below for each test video. As it can be seen from the experimental results, hybrid system gives the highest success rate. For example, in Table 6-3, the template matching algorithm in the hybrid application has found the correct object in 19 frames, despite finding the wrong object in 31 frames. Similarly, the histogram algorithm in the hybrid application has found the correct object in 13 frames, despite finding the wrong object in 48 frames. In the same way, SURF algorithm in the hybrid application has found the correct object in 98 frames, despite finding the wrong object in 2 frames. In total 7 frames, no objects matched. When individual algorithms are examined, template matching algorithm could not find object matches in any frame. Histogram algorithm has found the correct object in 98 frames. SURF algorithm has found the correct object in 89, despite finding the wrong object in 42 frames.

Moreover, the graphs of test results have been shared for easier understanding of the test results. For example, in Figure 6-2, match of rates is shown in y-axis, video frames are denoted in x-axis. The matching ratios, which between the selected object and the current video frame, obtained using the hybrid system and three algorithms are graphically displayed separately.

Another graphical example in Figure 6-3, by only running the hybrid system, it can be seen which of the 3 algorithms gives the highest match in each video frames. Thus, switches between the algorithms can be seen easily. Each node on the graph shows the algorithm that gives the best match on that frame.

As a result, it can be seen that the test data in the tables and graphs, hybrid system gives higher matching rates than the other algorithms to track object in the videos.

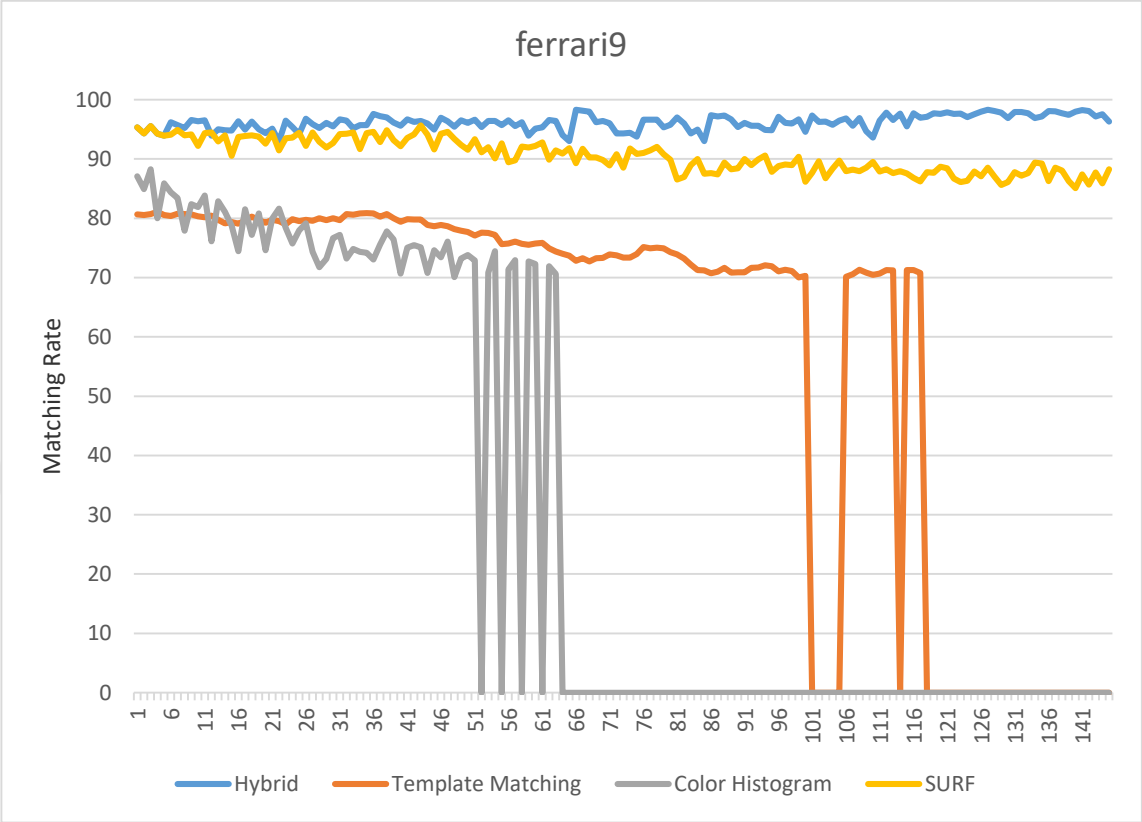


Figure 6-2 Test result of ferrari9

Table 6-1 Object tracking test result of ferrari9 video

#1 - ferrari9	**Frame Count**	**True Detection**		**False Detection**	**Not Found**	**DR %**	**Time Process**
Hybrid	145	***TM***	0	0	0	**100**	9 Min 9 Sec
		Histogram	134	0			
		SURF	11	0			
Template Matching		111		0	34	76,55	19 Sec
Histogram		59		0	86	40,68	7 Min 43 Sec
SURF		142		3	0	97,93	1 Min 4 Sec

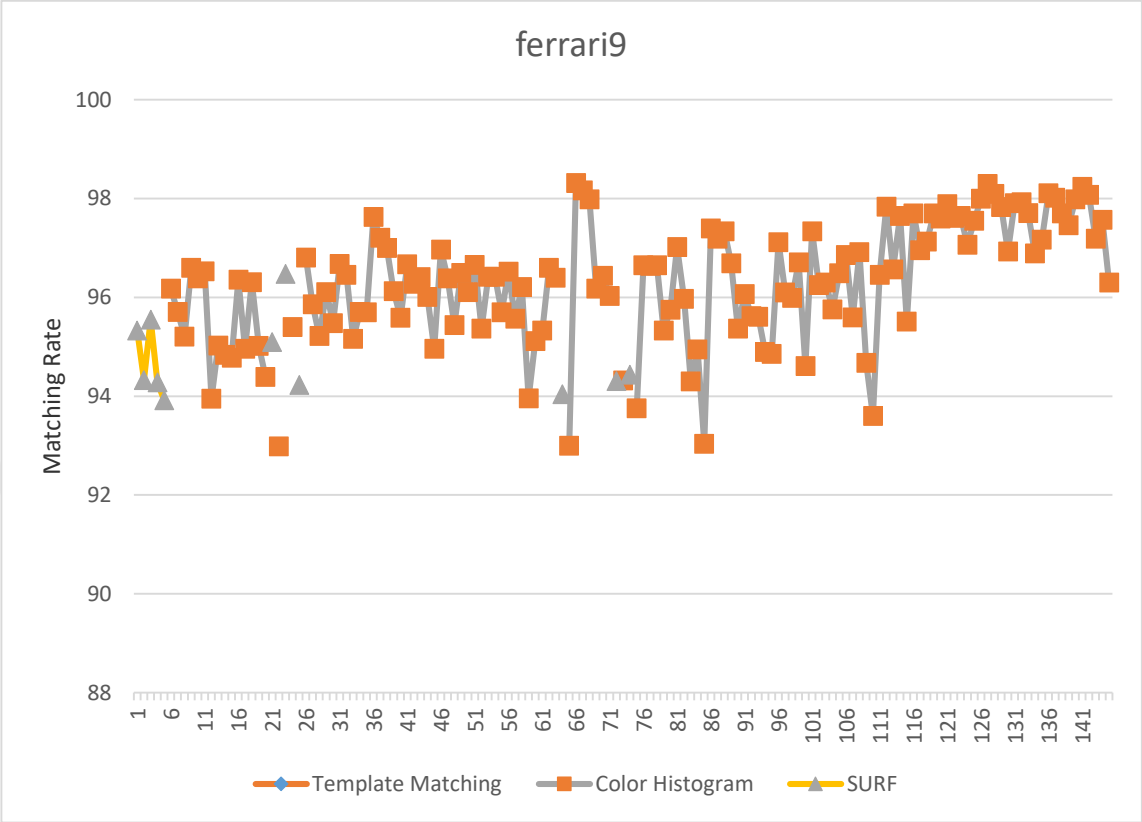


Figure 6-3 Test results of ferrari9 for Hybrid System including three algorithms

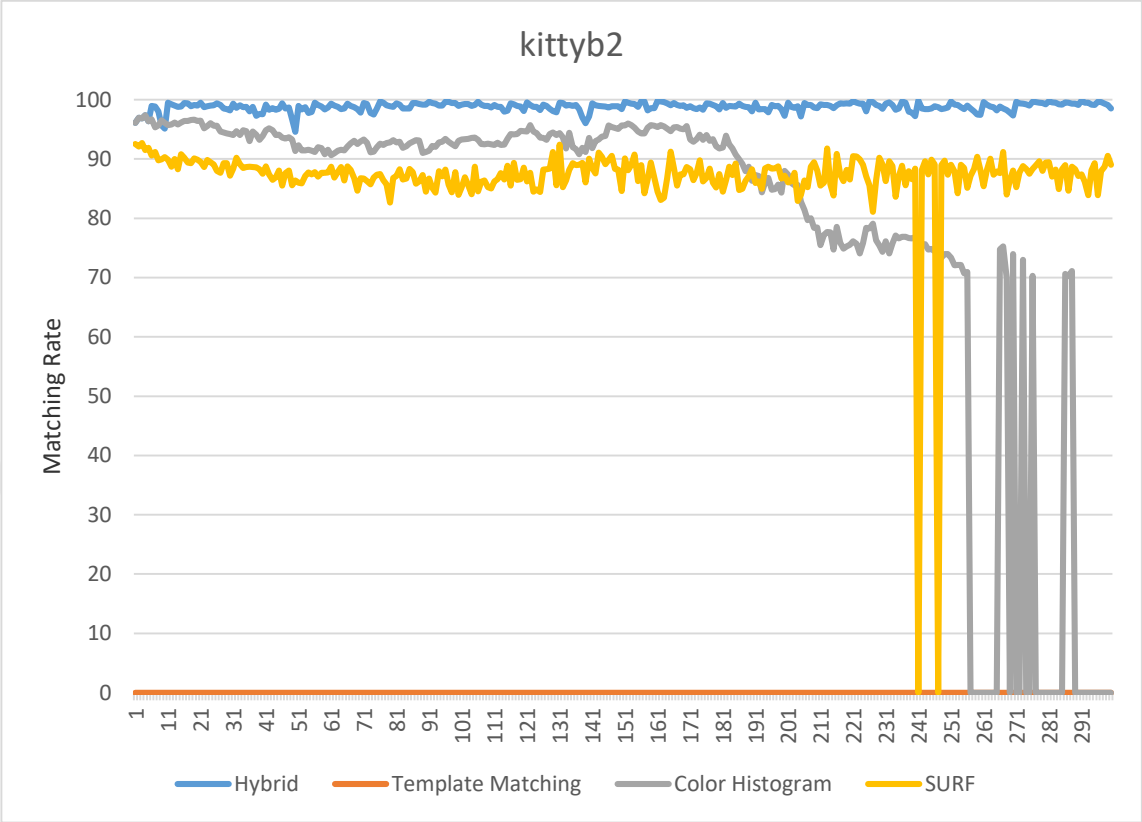


Figure 6-4 Test result of kittyb2

Table 6-2 Object tracking test result of kittyb2 video

#2 - kittyb2	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	300	***TM***	0	0	0	**100**	1 Hour 2 Min 19 Sec
		Histogram	300	0			
		SURF	0	0			
Template Matching		0		0	300	0	47 Sec
Histogram		205		9	86	68,33	1 Hour 38 Sec
SURF		174		124	2	58	1 Min 33 Sec

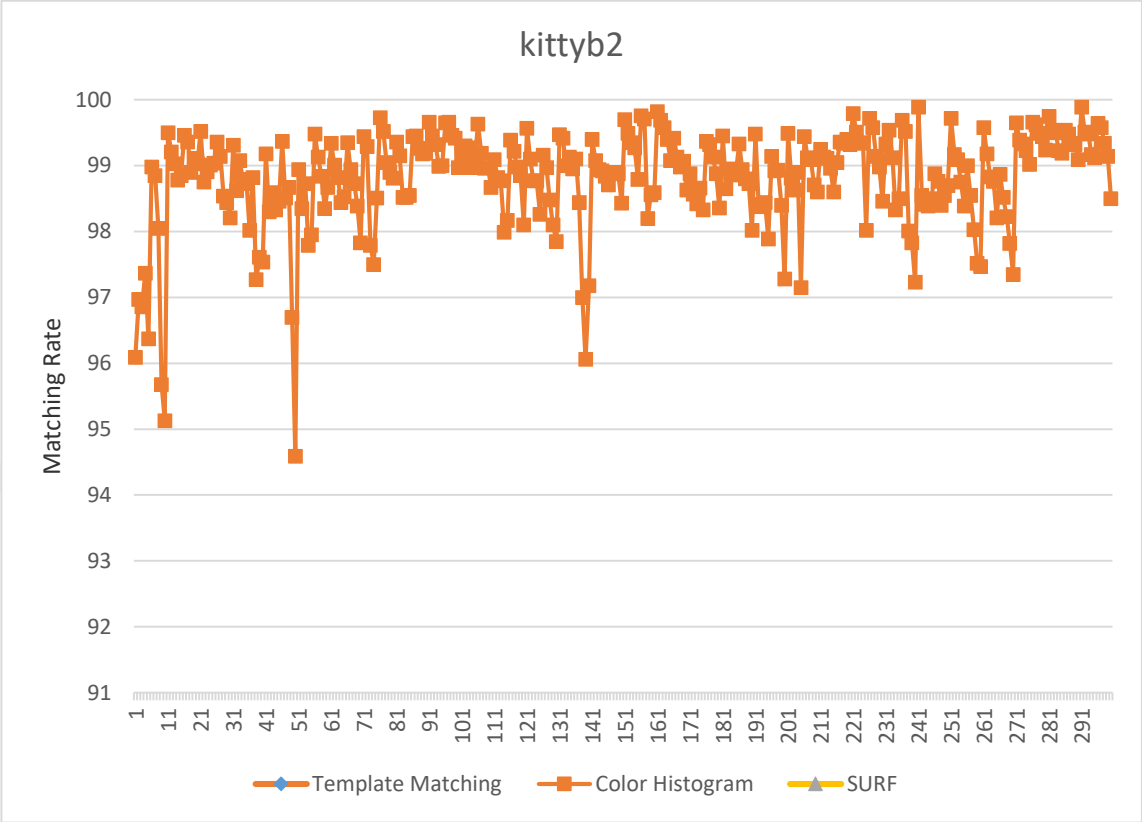


Figure 6-5 Test results of kittyb2 for Hybrid System including three algorithms

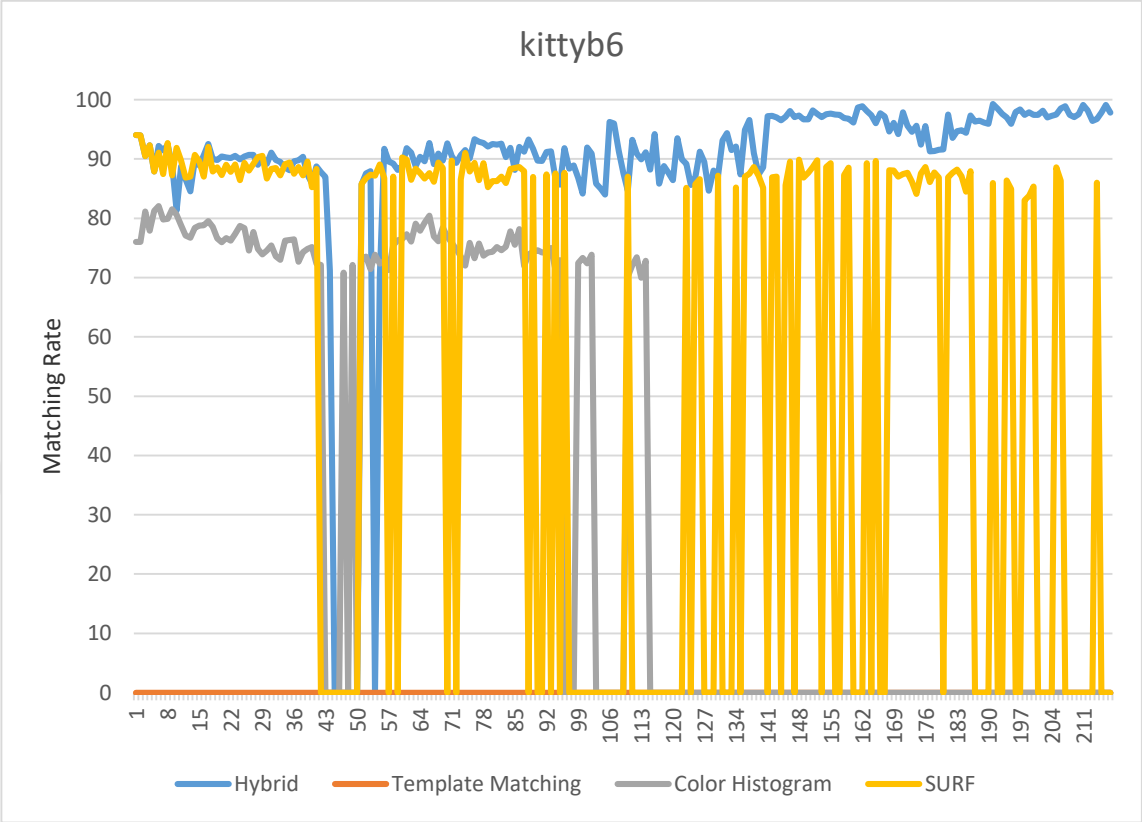


Figure 6-6 Test result of kittyb6

Table 6-3 Object tracking test result of kittyb6 video

#3 - kittyb6	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	217	***TM***	19	31	7	**59,90**	18 Min 17 Sec
		Histogram	13	48			
		SURF	98	2			
Template Matching		0		0	217	0	27 Sec
Histogram		98		0	119	45,16	10 Min 47 Sec
SURF		89		42	86	41,01	1 Min 5 Sec

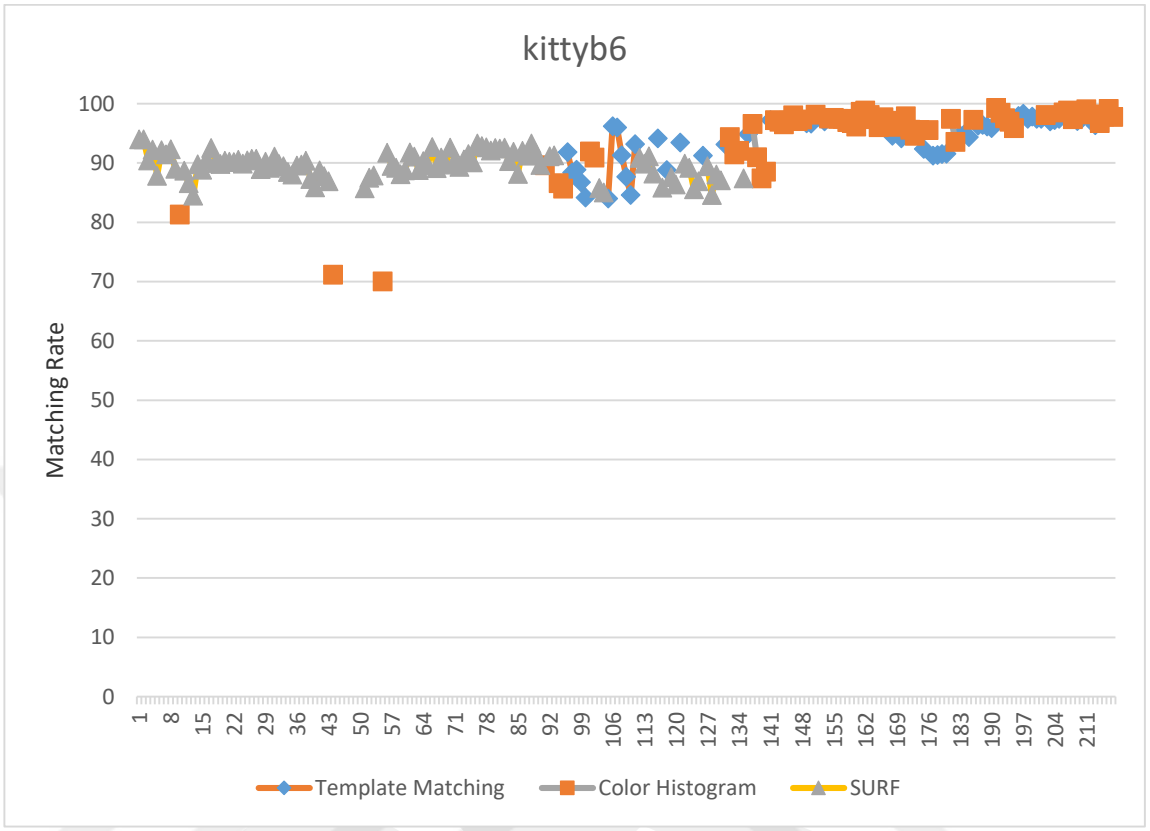


Figure 6-7 Test results of kittyb6 for Hybrid System including three algorithms

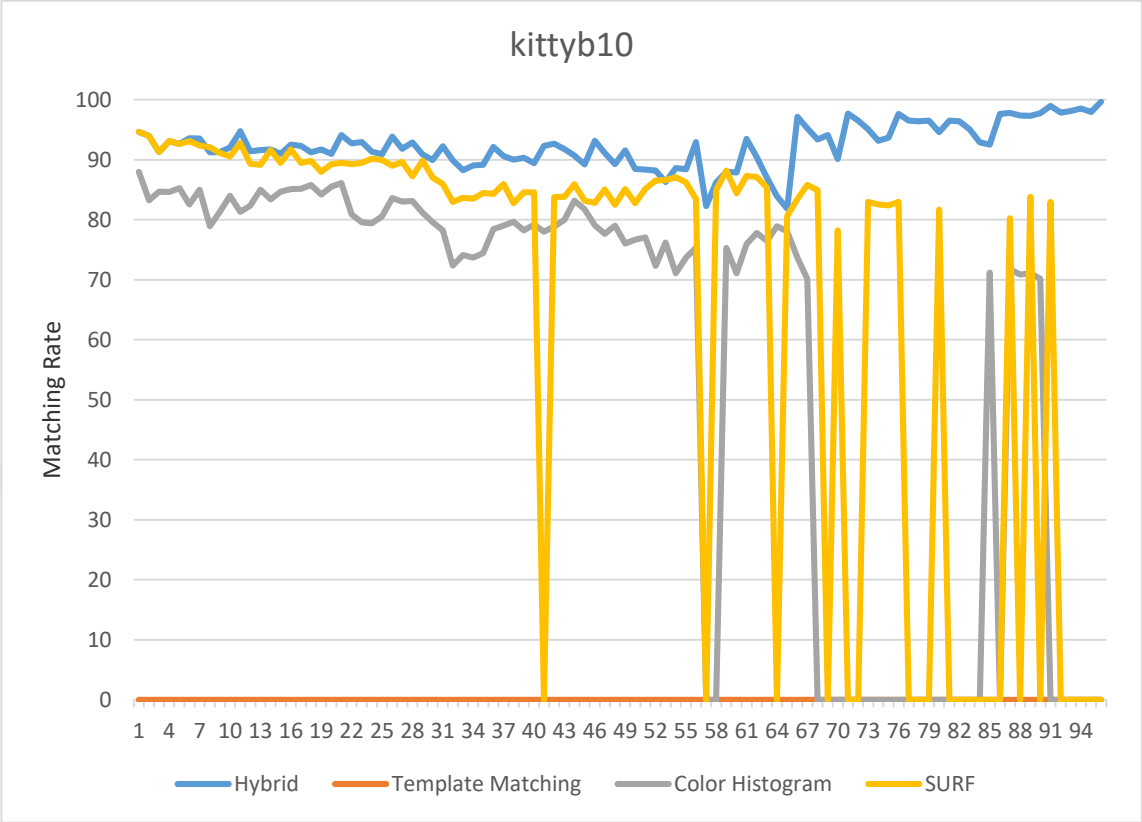


Figure 6-8 Test result of kittyb10

Table 6-4 Object tracking test result of kittyb10 video

#4 - kittyb10	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	97	***TM***	0	0	0	**100**	11 Min 55 Sec
		Histogram	42	0			
		SURF	55	0			
Template Matching		0		0	97	0	13 Sec
Histogram		70		0	27	72,16	6 Min 46 Sec
SURF		67		8	22	69,07	31 Sec

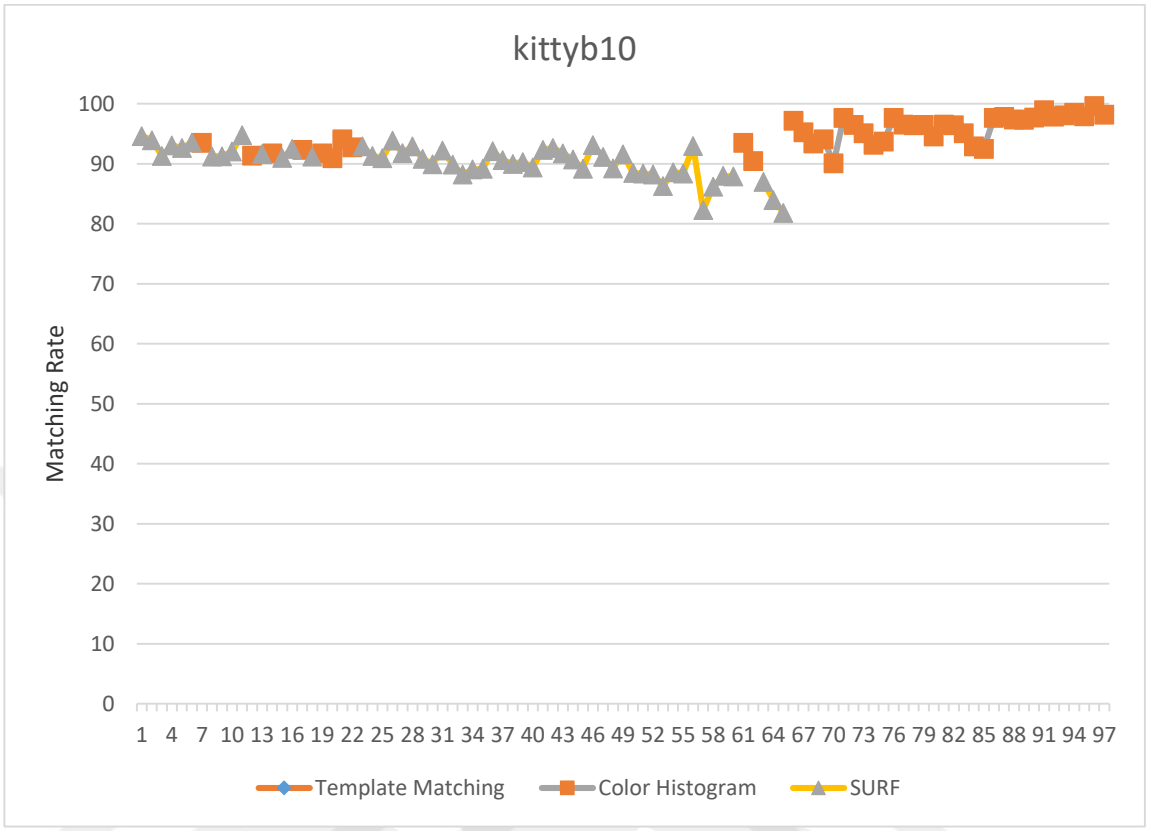


Figure 6-9 Test results of kittyb10 for Hybrid System including three algorithms

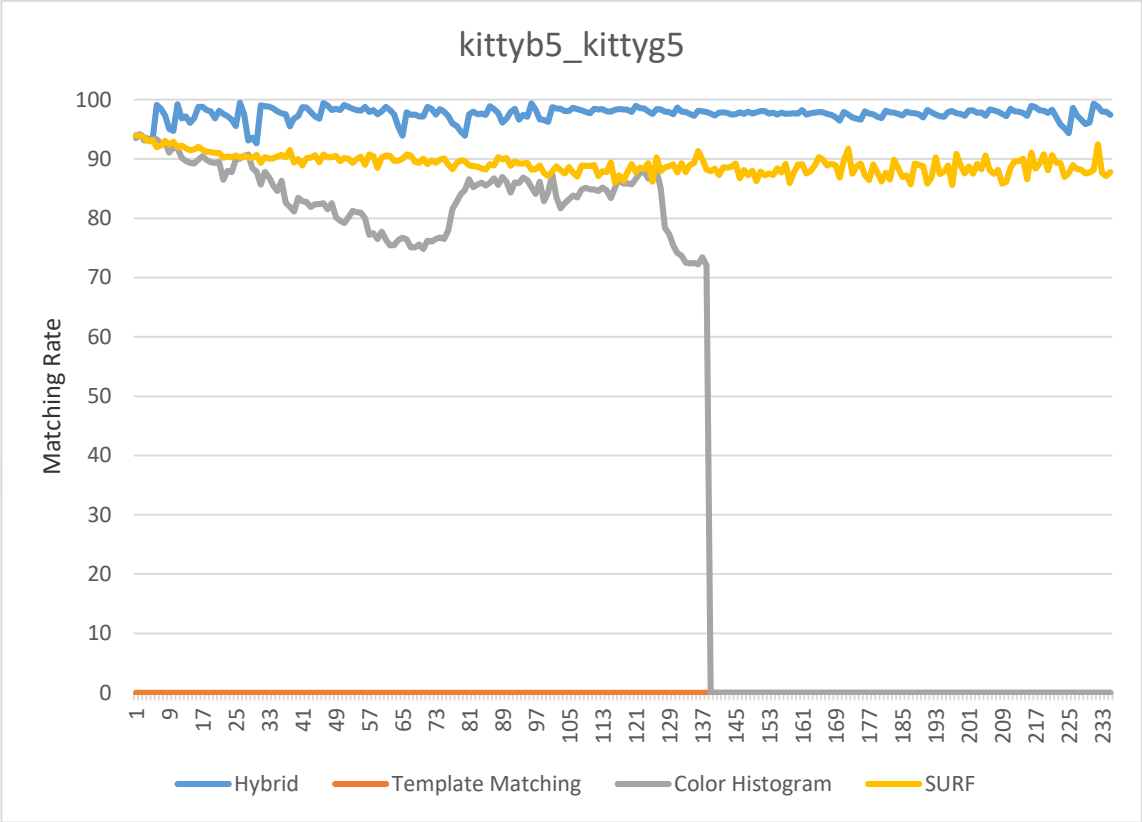


Figure 6-10 Test result of kittyb5_kittyg5

Table 6-5 Object tracking test result of kittyb5_kittyg5 video

#5 - kittyb5_kittyg5	**Frame Count**	**True Detection**		**False Detection**	**Not Found**	**DR %**	**Time Process**
Hybrid	235	***TM***	130	0	0	**100**	42 Min 16 Sec
		Histogram	100	0			
		SURF	5	0			
Template Matching		0		0	235	0	26 Sec
Histogram		138		0	97	58,72	39 Min 14 Sec
SURF		166		69	0	70,63	1 Min 32 Sec

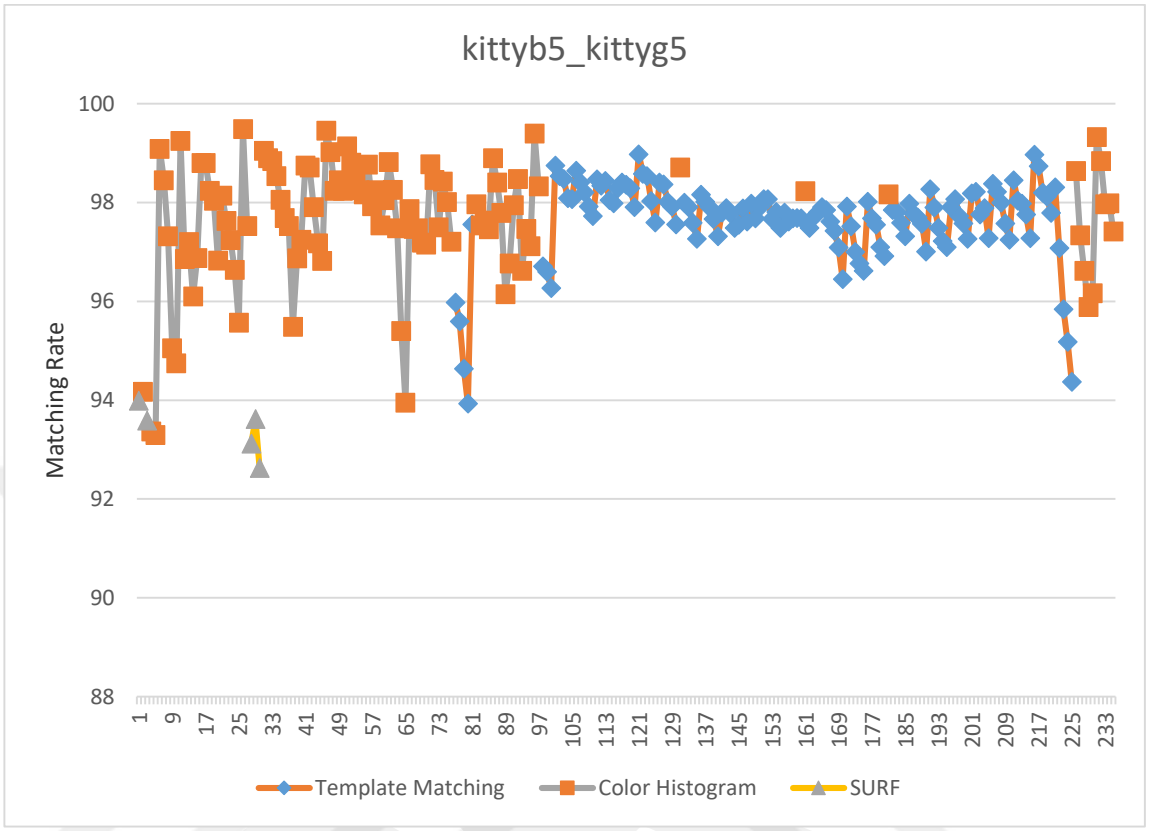


Figure 6-11 Test results of kittyb5_kittyg5 for Hybrid System including three algorithms

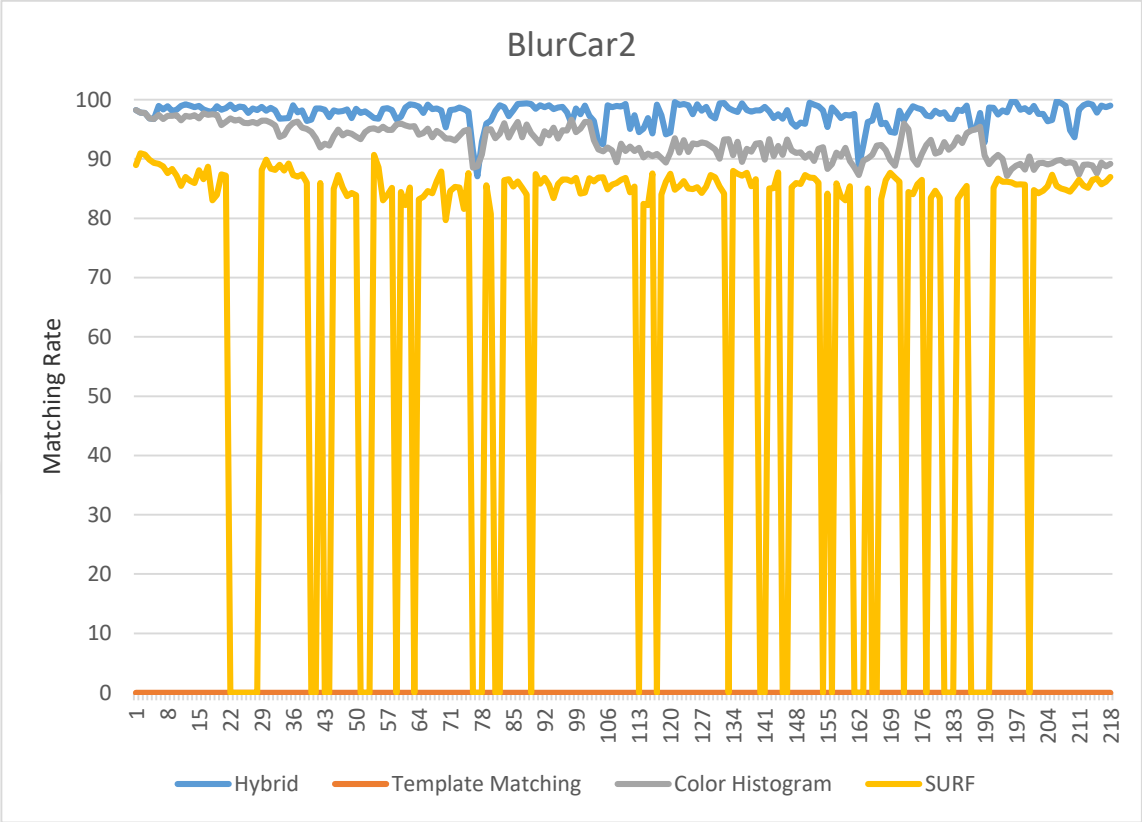


Figure 6-12 Test result of BlurCar2

Table 6-6 Object tracking test result of BlueCar2 video

#6 - BlurCar2	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	219	***TM***	0	0	0	**85,84**	26 Min 29 Sec
		Histogram	188	31			
		SURF	0	0			
Template Matching		0		0	219	0	22 Sec
Histogram		114		105	0	52,05	24 Min 28 Sec
SURF		141		31	47	64,38	1 Min 5 Sec

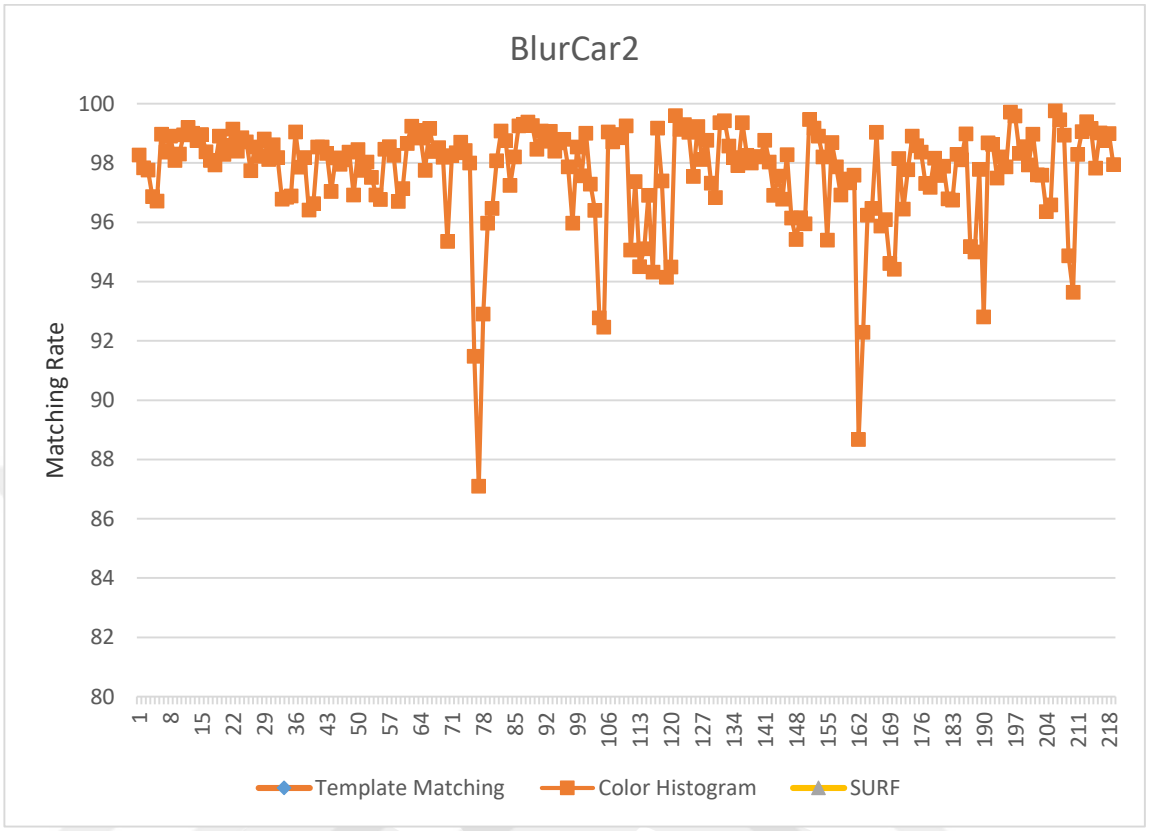


Figure 6-13 Test results of BlurCar2 for Hybrid System including three algorithms

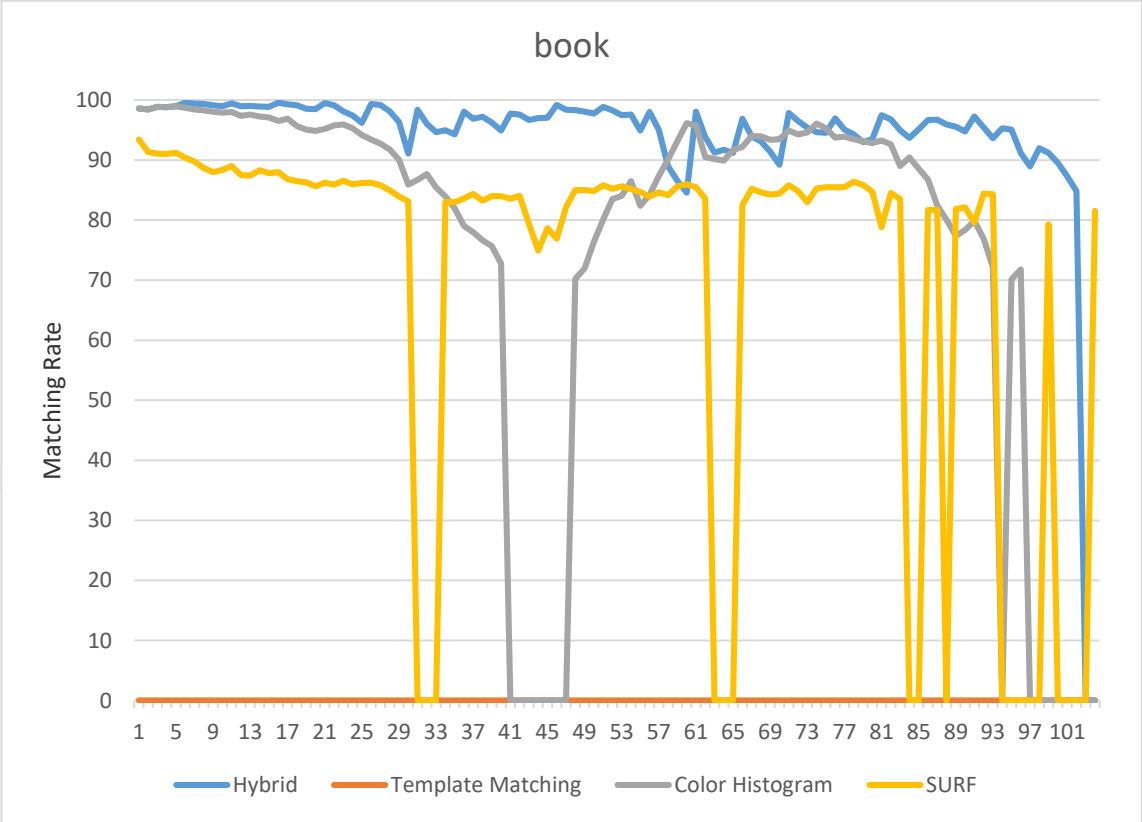


Figure 6-14 Test result of book

Table 6-7 Object tracking test result of book video

#7 - book	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	105	***TM***	0	0	3	**90,47**	1 Hour 23 Min 25 Sec
		Histogram	95	2			
		SURF	4	1			
Template Matching		0		0	105	0	25 Sec
Histogram		88		0	17	83,80	1 Hour 13 Min 12 Sec
SURF		83		3	19	79,04	1 Min 23 Sec

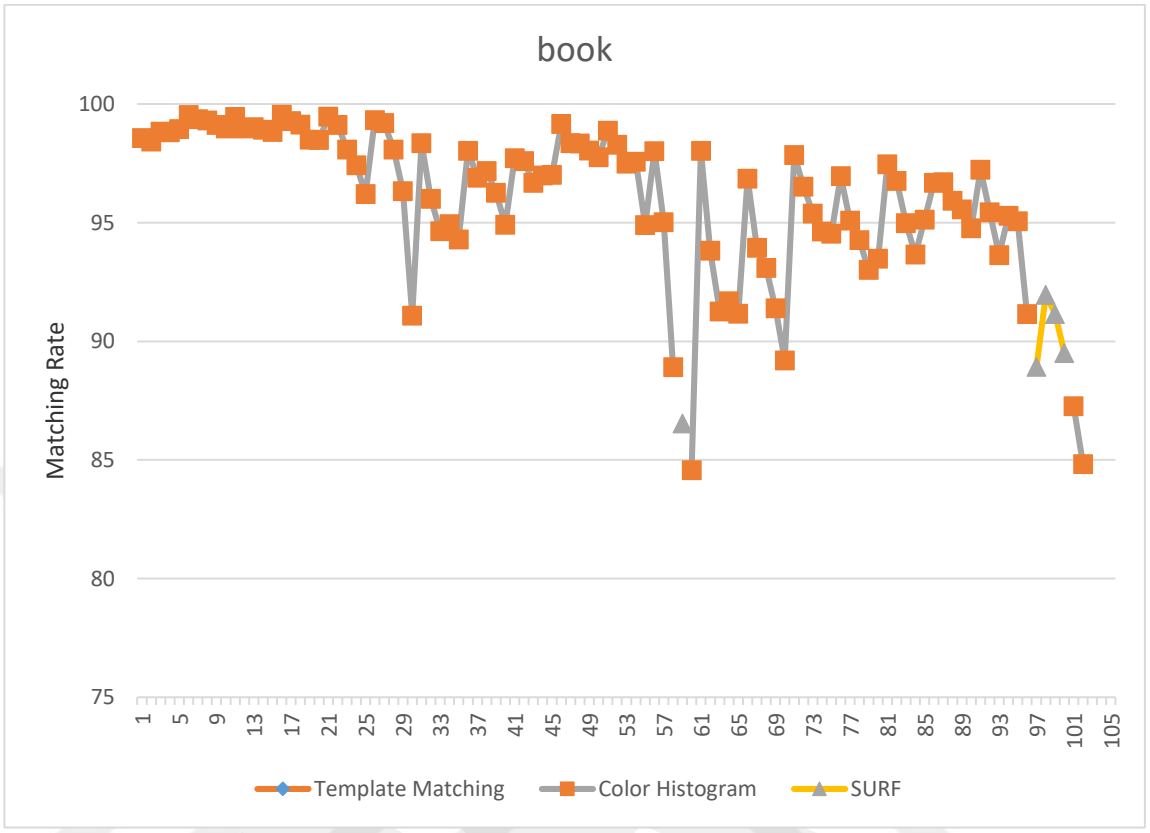


Figure 6-15 Test results of book for Hybrid System including three algorithms

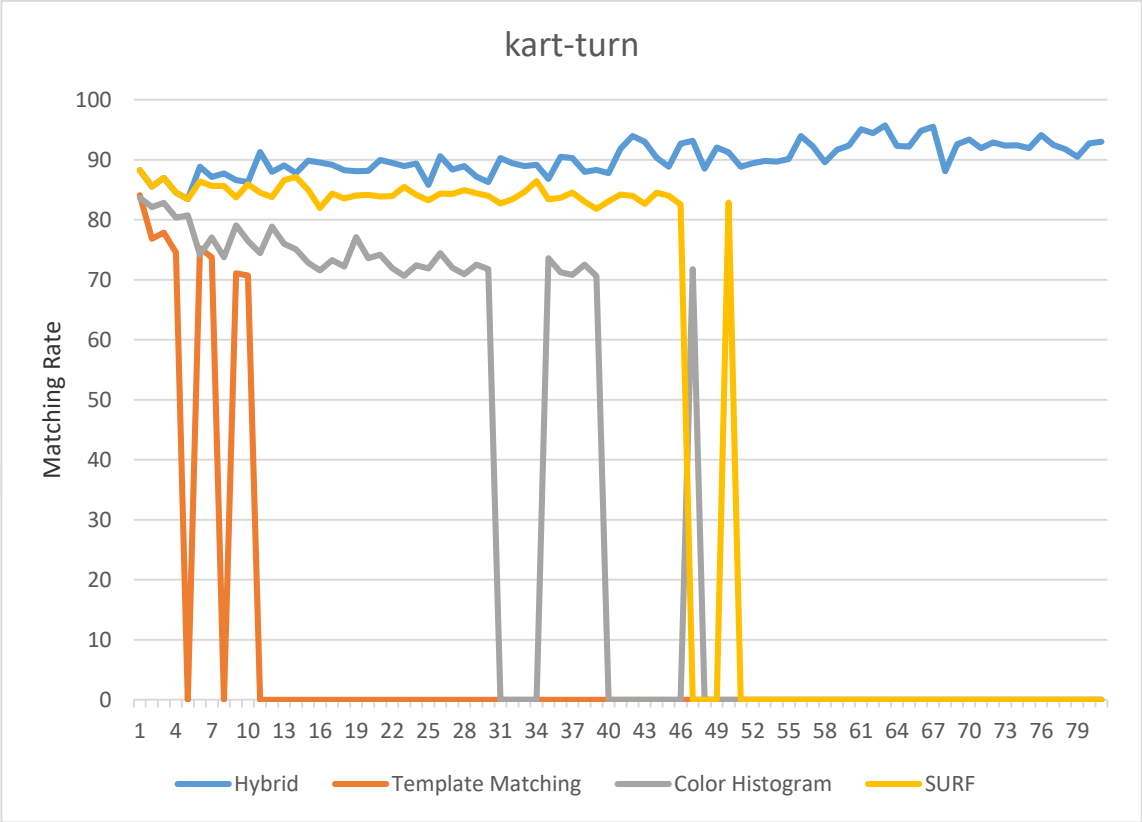


Figure 6-16 Test result of kart-turn

Table 6-8 Object tracking test result of kart-turn video

#8 - kart-turn	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	82	***TM***	2	0	0	**100**	36 Min 23 Sec
		Histogram	42	0			
		SURF	38	0			
Template Matching		8		0	74	9,75	20 Sec
Histogram		36		0	46	43,90	17 Min 39 Sec
SURF		45		2	35	54,87	1 Min 10 Sec

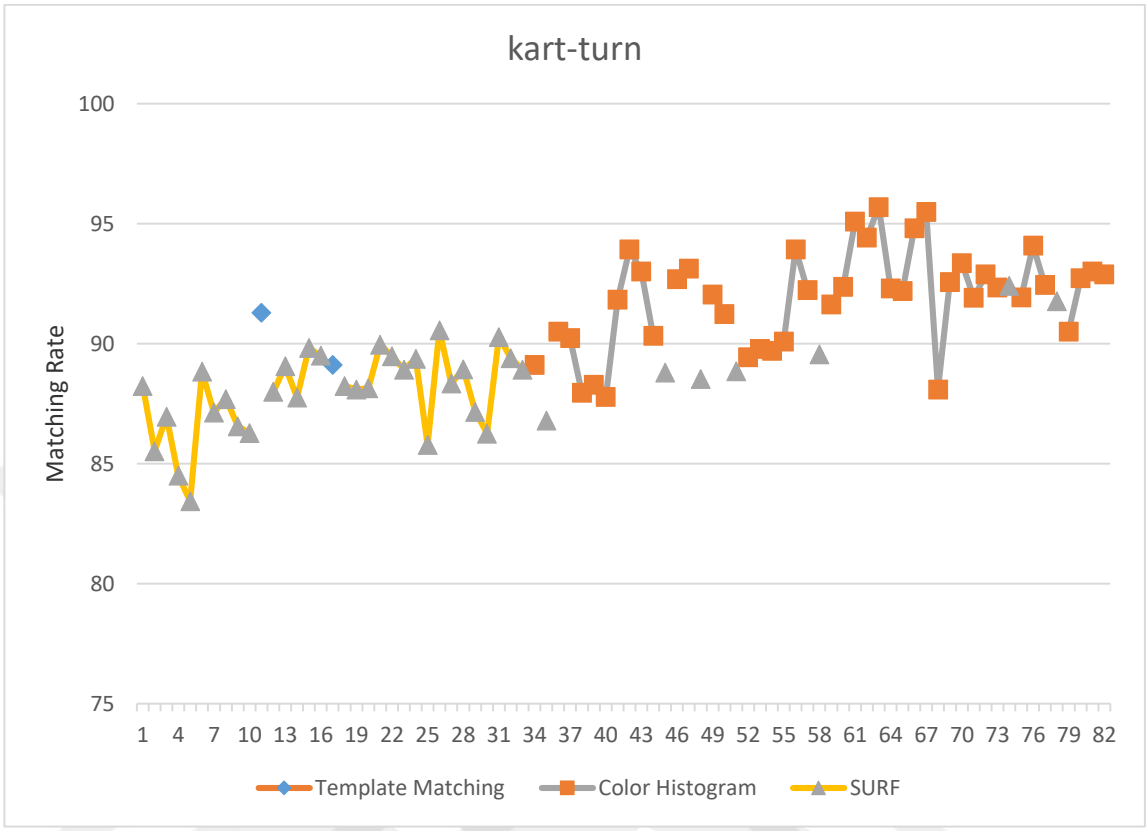


Figure 6-17 Test results of kart-turn for Hybrid System including three algorithms

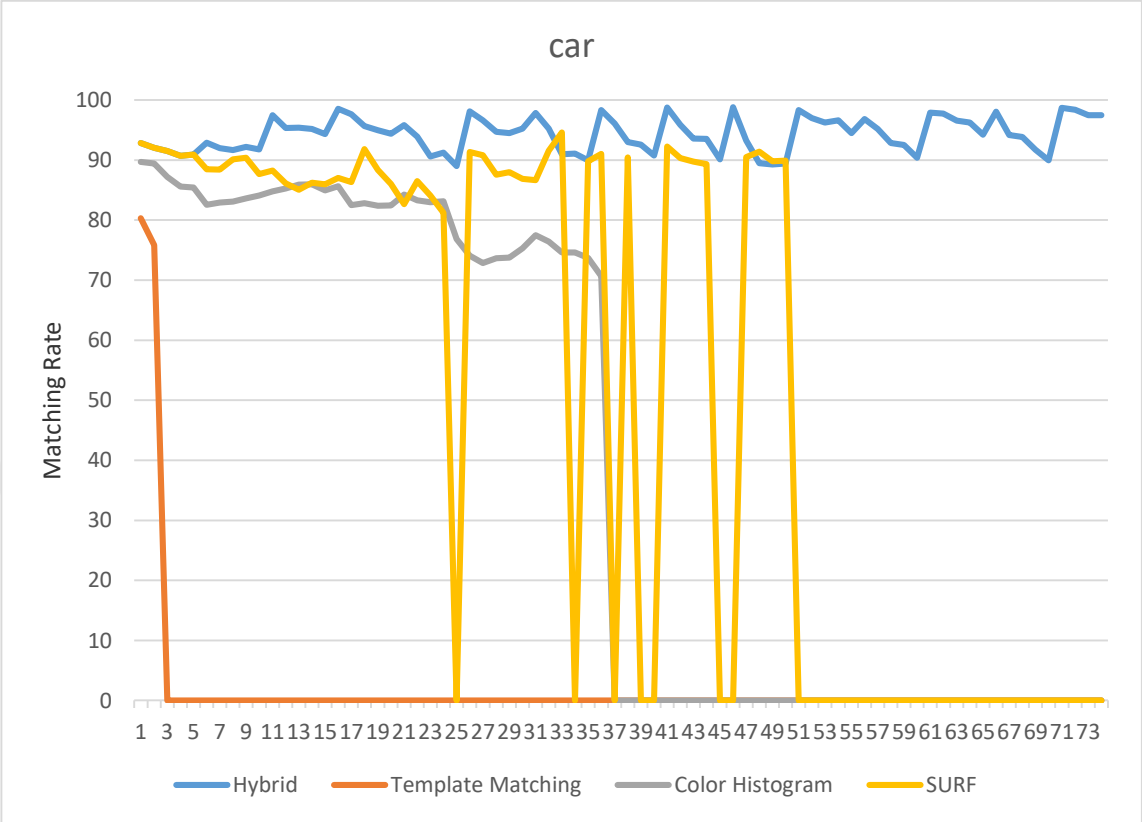


Figure 6-18 Test result of car

Table 6-9 Object tracking test result of car video

#9 - car	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	75	***TM***	0	0	0	**100**	1 Hour 31 Min 18 Sec
		Histogram	65	0			
		SURF	10	0			
Template Matching		2		0	73	2,66	23 Sec
Histogram		36		0	39	48	1 Hour 28 Min 14 Sec
SURF		19		24	32	25,33	1 Min 27 Sec

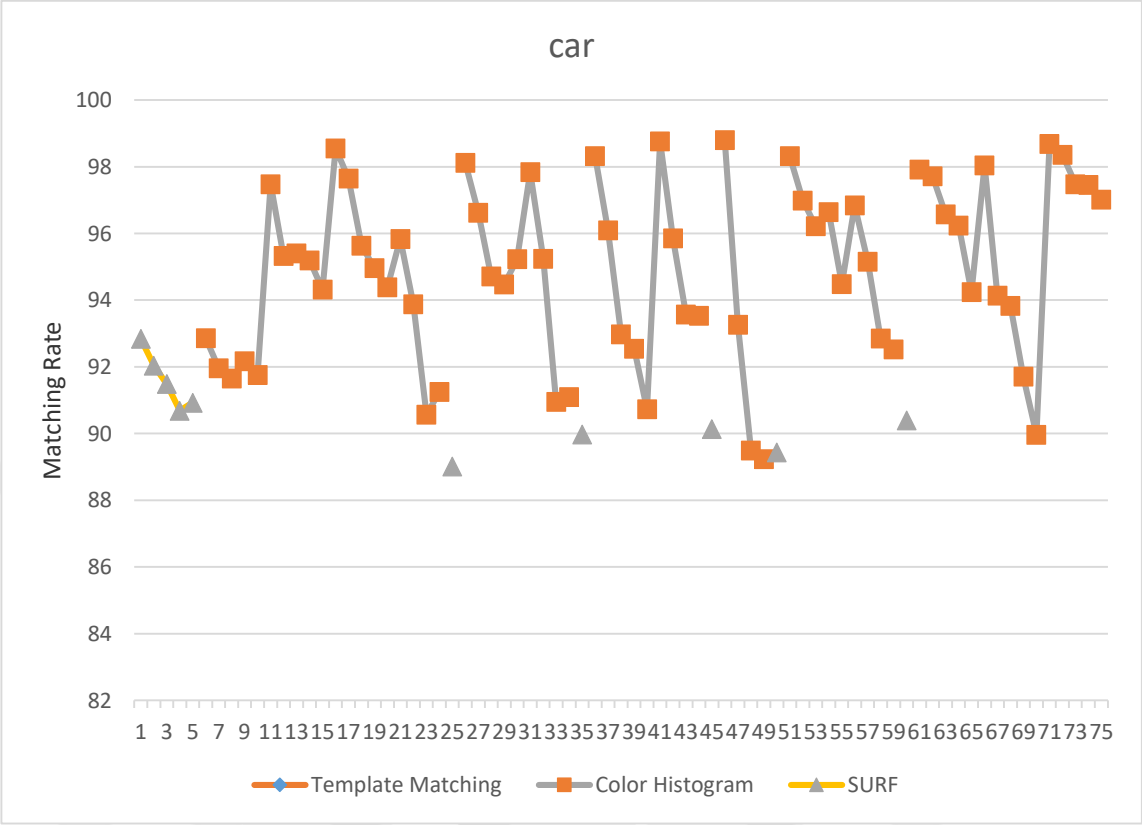


Figure 6-19 Test results of car for Hybrid System including three algorithms

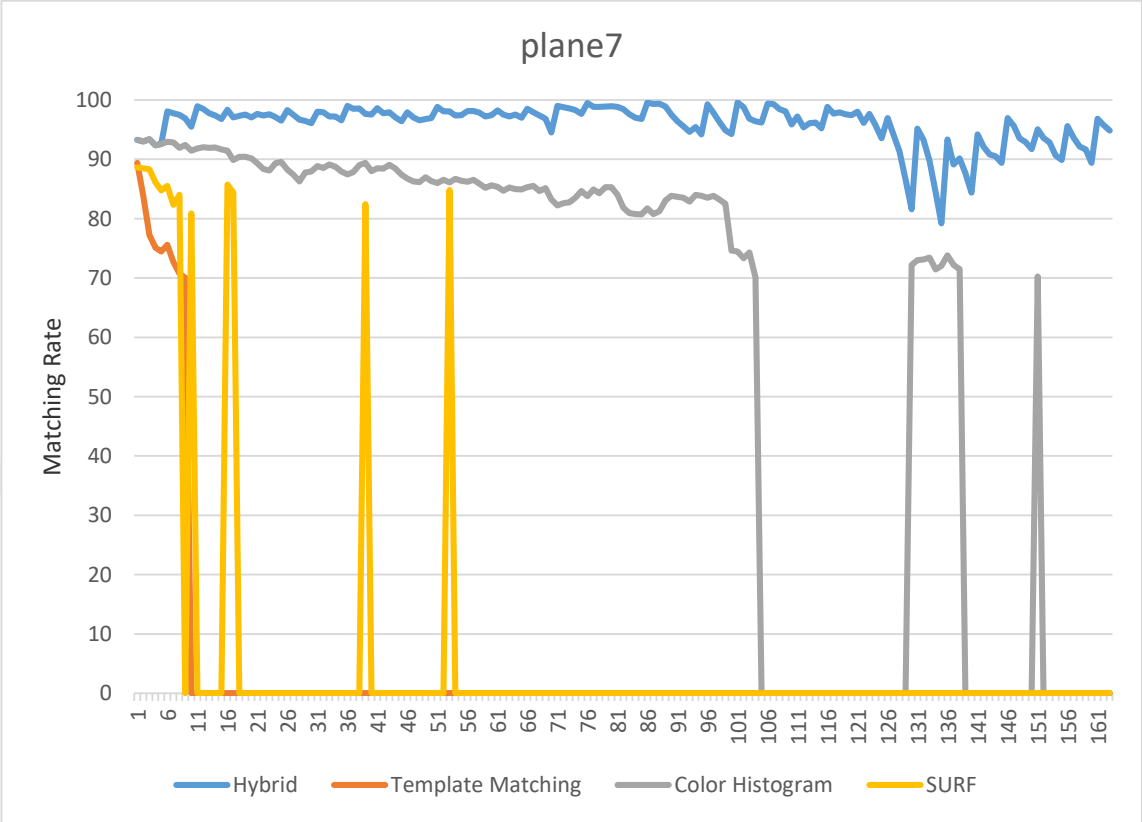


Figure 6-20 Test result of plane7

Table 6-10 Object tracking test result of plane7 video

#10 - plane7	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	163	***TM***	0	4	0	**82,82**	17 Min 27 Sec
		Histogram	132	24			
		SURF	3	1			
Template Matching		9		0	154	5,52	45 Sec
Histogram		104		9	49	63,80	17 Min
SURF		11		2	150	6,74	1 Min 30 Sec

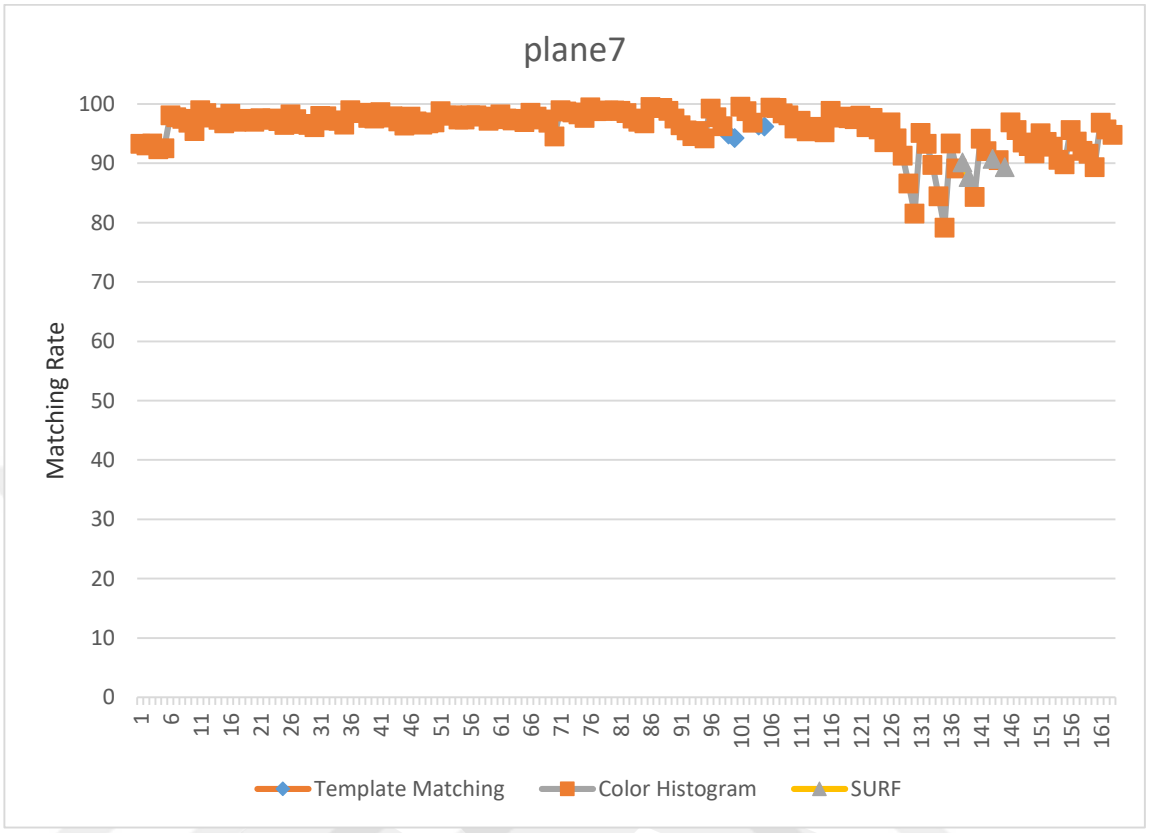


Figure 6-21 Test results of plane7 for Hybrid System including three algorithms

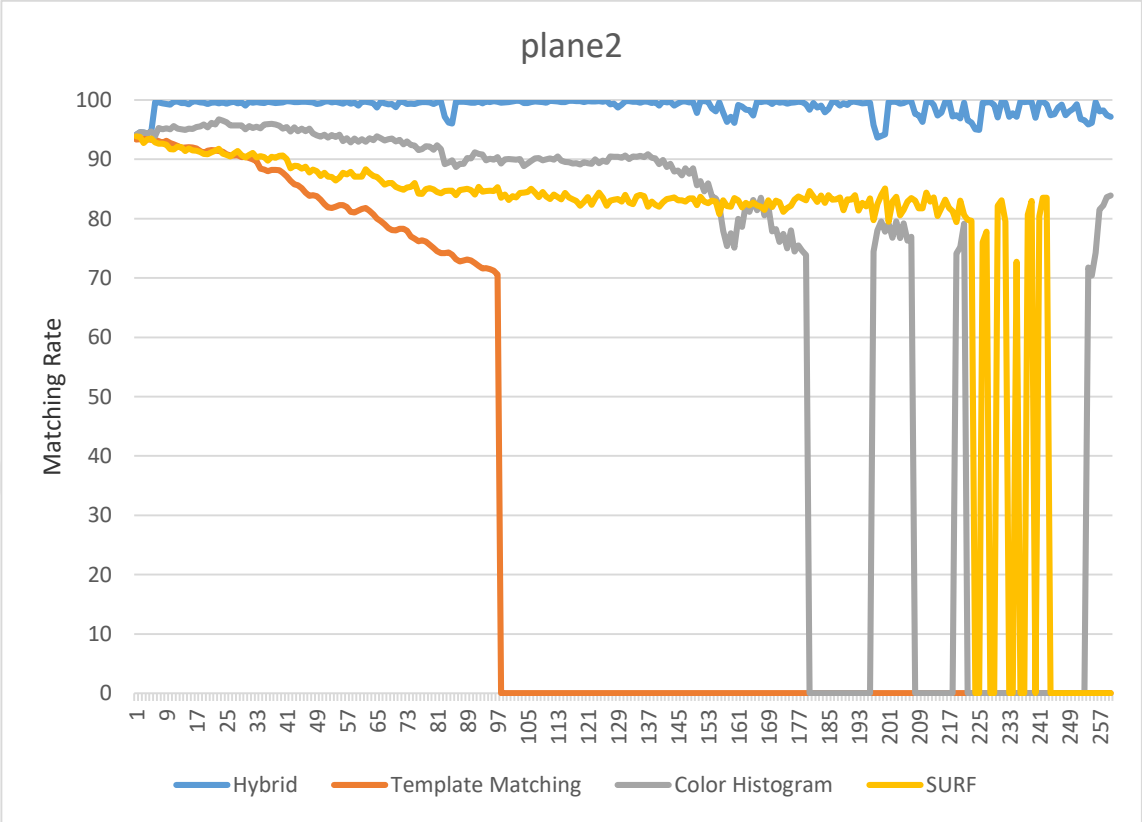


Figure 6-22 Test result of plane2

Table 6-11 Object tracking test result of plane2 video

#11 - plane2	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	260	***TM***	10	0	0	**95,38**	1 Hour 2 Min 31 Sec
		Histogram	238	12			
		SURF	0	0			
Template Matching		97		0	163	37,30	37 Sec
Histogram		193		7	60	74,23	53 Min 51 Sec
SURF		209		25	26	80,38	1 Min 5 Sec

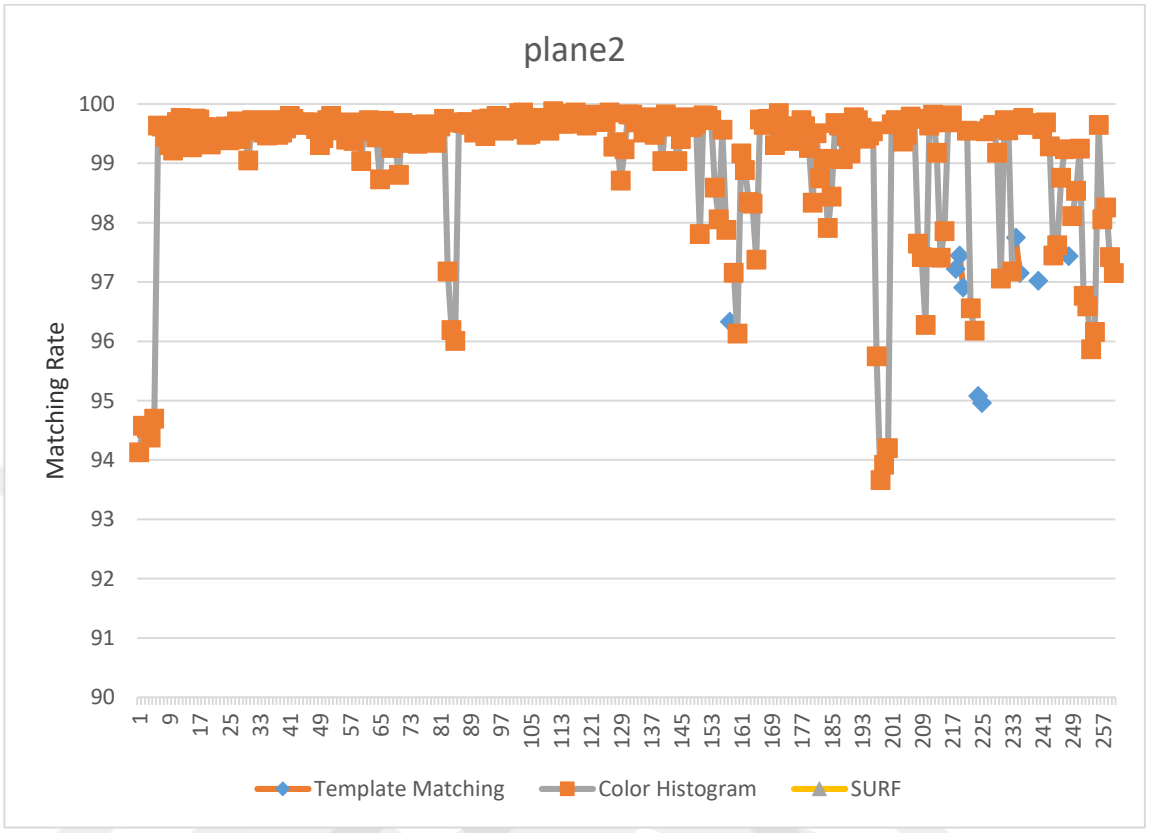


Figure 6-23 Test results of plane2 for Hybrid System including three algorithms

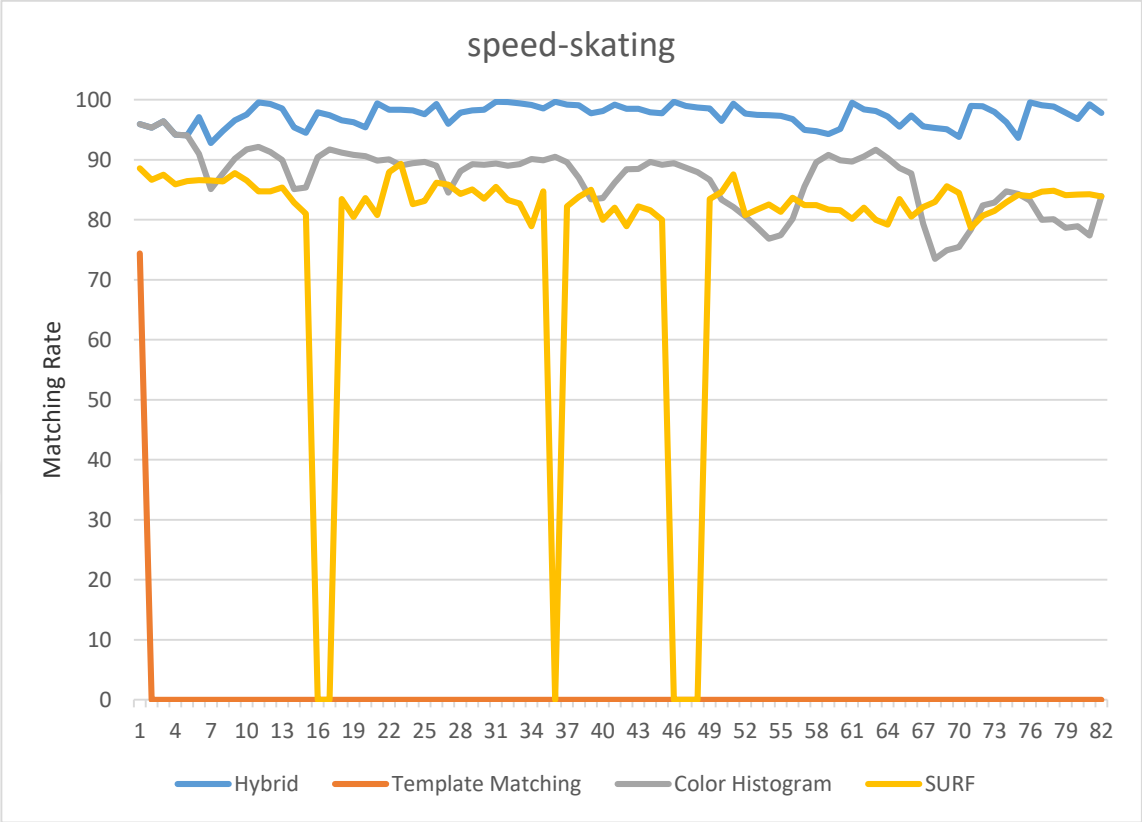


Figure 6-24 Test result of speed-skating

Table 6-12 Object tracking test result of speed-skating video

#12 - speed-skating	Frame Count	True Detection		False Detection	Not Found	DR %	Time Process
Hybrid	82	***TM***	0	0	0	**100**	1 Hour 18 Min 32 Sec
		Histogram	82	0			
		SURF	0	0			
Template Matching		1		0	81	1,21	34 Sec
Histogram		75		7	0	91,46	1 Hour 31 Min 12 Sec
SURF		36		40	6	43,9	1 Min 34 Sec

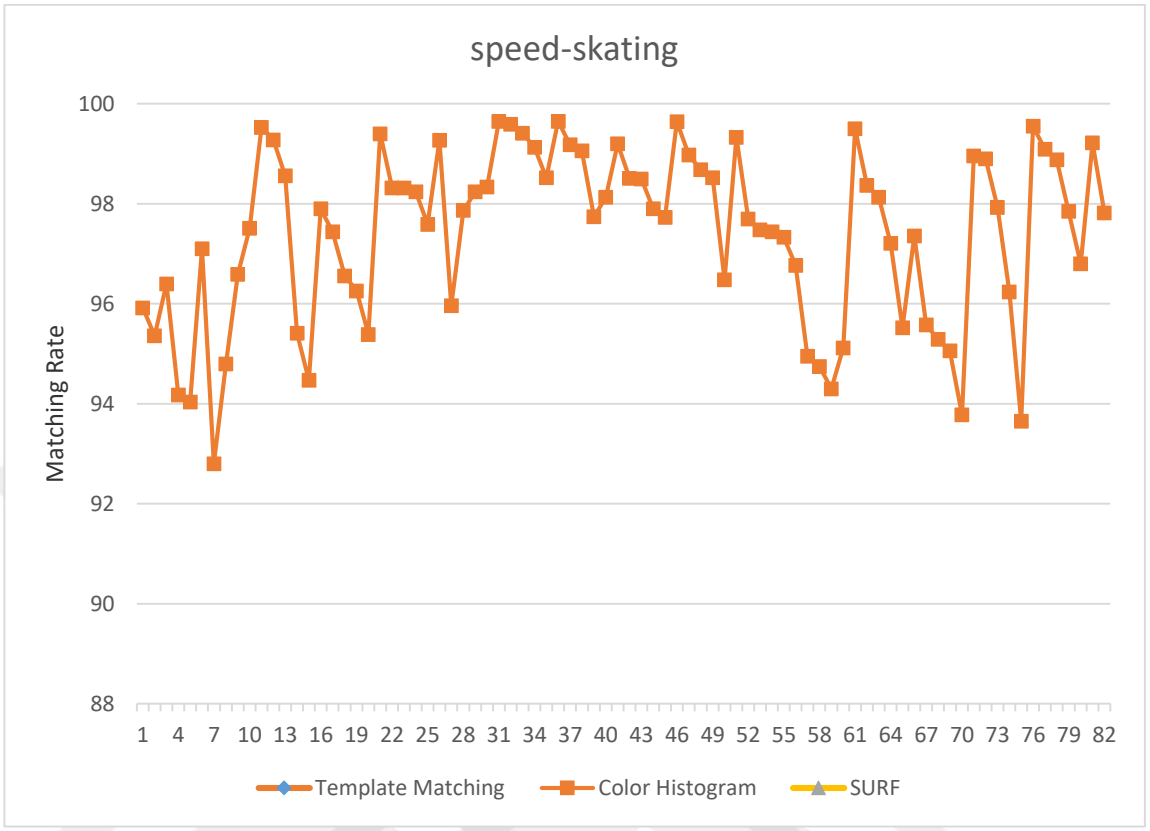


Figure 6-25 Test results of speed-skating for Hybrid System including three algorithms

CHAPTER 7

CONCLUSION

In final chapter, a summary of this thesis is provided. In the first chapter, the subject of object tracking in video, one of the most difficult issues in the field of computer vision, was researched. Some difficulties of object tracking such as low resolution, sudden object motions, noise on object, and occlusion were given.

In chapter 2, literature survey was performed and previous works about object tracking papers and thesis were summarized in brief.

In third chapter, preliminary work was done to better understand our system. For example, template matching and color histogram algorithms work with sliding window technique. Color space conversions are also needed for these algorithms.

In chapter 4, a study has been performed on the three algorithms in image processing which they are template matching, color based histogram and SURF. The process steps in each algorithm are explained. We have implemented the process steps recommended by OpenCV [56]. We have used smoothing and thresholding instead of morphological operations.

In chapter 5, a robust application to track object in video consisting of three algorithms has been developed in OpenCV using Java programming language. In other words, it is provided to continue object tracking in video with other algorithm at the points where an algorithm is insufficient.

The last chapter, the findings of the study were presented. For this, experimental tests have been performed on 12 test videos provided datasets which are NTU-VOI 2018, Visual Tracker Benchmark 2013, NfS 2017, Davis 2017 and one video on YouTube. According to the video test results, our application has more successful and accurate results in object tracking than individual algorithms such as template matching, color histogram and SURF. But the worst part of our application is that the video processing time may last long. Because instead of one algorithm, we run 3 algorithms at the same time.

For future works, we think that, instead of running three algorithms at the same time, it is possible to find object by an algorithm and continue to find object by another algorithm when the algorithm finding object falls below a certain similarity value. Thus, we will also reduce video processing time and improve system performance. Even the hybrid system we developed will be able to track objects in real-time videos.

REFERENCES

[1] X. Li, W. Hu, C. Shen, Z. Zhang, A. Dick, and A. V. D. Hengel, "A survey of appearance models in visual object tracking," *ACM Transactions on Intelligent Systems and Technology*, vol. 4, no. 4, pp. 1–48, Mar. 2013.

[2] M. Yokoyama and T. Poggio, "A Contour-Based Moving Object Detection and Tracking," *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 271–276, Nov. 2005.

[3] M. Jeeva and M. J. Balakrishnan, "A Novel approach for Evaluation of video tracking Under Real world conditions," *IOSR Journal of Computer Engineering*, vol. 16, no. 1, pp. 55–59, Jan. 2014.

[4] M. Parmar, "A Survey of Video Object Tracking Methods," *International Journal of Engineering Development and Research*, vol. 4, no. 1, 2016.

[5] Shilpa, M.R. Sunitha, H.l. Prathap, "A Survey on Moving Object Detection and Tracking Techniques," *International Journal of Engineering and Computer Science*, vol. 5, no. 5, pp. 16376–16382, May 2016.

[6] B. Deori and D. Meitei Thounaojam, "A Survey on Moving Object Tracking in Video," *International Journal on Information Theory*, vol. 3, no. 3, pp. 31–46, Jul. 2014.

[7] S. R. Balaji and S. Karthikeyan, "A survey on moving object tracking using image processing," *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, Feb. 2017.

[8] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A Survey on Object Detection and Tracking Methods," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 2, Feb. 2014.

[9] M. Sundaram and G. Nayagam, "A survey on Real time Object Detection and Tracking Algorithms," *International Journal of Applied Engineering Research*, vol. 10, pp. 8290–8297, Apr. 2015.

[10] T. S. Waykole and Y. K. Jain, "Detecting and Tracking of Moving Objects from Video," *International Journal of Computer Applications*, vol. 81, no. 18, pp. 23–28, Nov. 2013.

[11] S. Thorat and M. Nagmode, "Detection & tracking of moving object," *International Journal of Innovative Research in Advanced Engineering*, vol. 1, no. 1, Apr. 2014.

[12] Y. Sugaya and K. Kanatani, "Extracting Moving Objects from a Moving Camera Video Sequence," in *10th Symposium on Sensing via Image Information*, 2004, vol. 39, pp. 279–284.

[13] B. Karasulu and S. Korukoglu, "Moving Object Detection and Tracking in Videos," *Performance Evaluation Software SpringerBriefs in Computer Science*, pp. 7–30, 2013.

[14] Y. Wang, J. F. Doherty, and R. E. Van Dyck, "Moving object tracking in video," *Proceedings 29th Applied Imagery Pattern Recognition Workshop*, pp. 95–101, 2000.

[15] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Computing Surveys*, vol. 38, no. 4, pp. 1–45, 2006.

[16] Haritaoglu, D. Harwood, and L. S. Davis, "W/sup 4/: real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, 2000.

[17] A. Coifman, D. Beymer, P. Mclauchlan, and J. Malik, "A real-time computer vision system for vehicle tracking and traffic surveillance," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 4, pp. 271–288, 1998.

[18] J.-C. Tai, S.-T. Tseng, C.-P. Lin, and K.-T. Song, “Real-time image tracking for automatic traffic monitoring and enforcement applications,” *Image and Vision Computing*, vol. 22, no. 6, pp. 485–501, 2004.

[19] O. Masoud and N. P. Papanikolopoulos, “A novel method for tracking and counting pedestrians in real-time using a single camera,” *IEEE Transactions on Vehicular Technology*, vol. 50, no. 5, pp. 1267–1278, 2001.

[20] V. I. Pavlovic, R. Sharma, and T. S. Huang, “Visual interpretation of hand gestures for human-computer interaction: a review,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 677–695, 1997.

[21] S. Paschalakis and M. Bober, “Real-time face detection and tracking for mobile videoconferencing,” *Real-Time Imaging*, vol. 10, no. 2, pp. 81–94, 2004.

[22] The NTU Video-Object-Instance (NTU-VOI), 2018. [Online]. Available: https://sites.google.com/site/jingjingmengsite/research/ntu-voi/data. [Accessed: 19-May-2019].

[23] “Features2D Homography to find a known object,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4.2/d7/dff/tutorial_feature_homography.html. [Accessed: 19-May-2019].

[24] Visual Tracker Benchmark, 2013. [Online]. Available: http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html. [Accessed: 19-May-2019].

[25] Need for Speed – NfS, 2017. [Online]. Available: http://ci2cv.net/nfs/index.html. [Accessed: 19-May-2019].

[26] Davis 2017 Semi-supervised, 2017. [Online]. Available: https://davischallenge.org/davis2017/code.html. [Accessed: 19-May-2019].

[27] S. Fazli, H. M. Pour, and H. Bouzari, “Particle Filter Based Object Tracking with Sift and Color Feature,” *2009 Second International Conference on Machine Vision*, pp. 89–93, 2009.

[28] T. Kim, S. Lee, and J. Paik, “Combined shape and feature-based video analysis and its application to non-rigid object tracking,” *IET Image Processing*, vol. 5, no. 1, pp. 87–100, Feb. 2011.

[29] A. J. Lipton, H. Fujiyoshi, and R. S. Patil, “Moving target classification and tracking from real-time video,” *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV98 (Cat. No.98EX201)*, pp. 8–14, Oct. 1998.

[30] F. Jurie and M. Dhome, “A simple and efficient template matching algorithm,” *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, pp. 544–549, Jul. 2001.

[31] M. Ryan and N. Hanafiah, “An Examination of Character Recognition on ID card using Template Matching Approach,” *Procedia Computer Science*, vol. 59, pp. 520–529, 2015.

[32] S. Omachi and M. Omachi, “Fast Template Matching with Polynomials,” *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2139–2149, 2007.

[33] T. Kaneko and O. Hori, “Feature selection for reliable tracking using template matching,” *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, 2003.

[34] N. Prabhakar, V. Vaithiyanathan, A. P. Sharma, A. Singh, and P. Singhal, “Object Tracking Using Frame Differencing and Template Matching,” *Research Journal of Applied Sciences, Engineering and Technology*, 2012.

[35] M. J. Swain and D. H. Ballard, “Color Indexing,” *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

[36] A. Ford and A. Roberts, “Color Space conversions,” August 11, 1998.

[37] T. Mahalakshmi, R. Muthaiah, and P. Swaminathan, “Review Article: An Overview of Template Matching Technique in Image Processing,” *Research Journal of Applied Sciences, Engineering and Technology*, vol. 4, pp. 5469–5473, 2012.

[38] P. Swaroop and N. Sharma, “An Overview of Various Template Matching Methodologies in Image Processing,” *International Journal of Computer Applications*, vol. 153, no. 10, pp. 8–14, 2016.

[39] N. Perveen, D. Kumar, and I. Bhardwaj, “An Overview on Template Matching Methodologies and its Applications,” *International Journal of Research in Computer and Communication Technology*, vol. 2, no. 10, Oct. 2013.

[40] F. Alsaade, “Fast and Accurate Template Matching Algorithm Based on Image Pyramid and Sum of Absolute Difference Similarity Measure,” *Research Journal of Information Technology*, vol. 4, no. 4, pp. 204–211, 2012.

[41] P. Lewis, “Fast Template Matching,” *Vision Interface 95, Canadian Image Processing and Pattern Recognition Society*, pp. 120–123, 1995.

[42] S. K. Lam, C. Y. Yeong, C. T. Yew, W. S. Chai, and S. A. Suandi, “A Study on Similarity Computations in Template Matching Technique for Identity Verification,” *International Journal on Computer Science and Engineering*, vol. 2, no. 8, pp. 2659–2665, 2010.

[43] W. Chantara, J.-H. Mun, D.-W. Shin, and Y.-S. Ho, “Object Tracking using Adaptive Template Matching,” *IEIE Transactions on Smart Processing and Computing*, vol. 4, no. 1, pp. 1–9, 2015.

[44] W. Jia, H. Zhang, X. He, and Q. Wu, “A Comparison on Histogram Based Image Matching Methods,” *2006 IEEE International Conference on Video and Signal Based Surveillance*, 2006.

[45] M. Mason and Z. Duric, “Using histograms to detect and track objects in color video,” *Proceedings 30th Applied Imagery Pattern Recognition Workshop (AIPR 2001). Analysis and Understanding of Time Varying Imagery*, pp. 154–159, 2001.

[46] I. Agbinya and D. Rees, “Multi-Object Tracking in Video,” *Real-Time Imaging*, vol. 5, no. 5, pp. 295–304, 1999.

[47] J. Sangoh, “Histogram-Based Color Image Retrieval,” Psych221/EE362 Project Report, Stanford University, 2001.

[48] P. M. Panchal, S. R. Panchal, and S. Shah, “A Comparison of SIFT and SURF,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 2, Apr. 2013.

[49] Juan and O. Gwun, “A Comparison of SIFT, PCA-SIFT and SURF,” *International Journal of Image Processing*, vol. 3, 2009.

[50] G. Du, F. Su, and A. Cai, “Face recognition using SURF features,” MIPPR 2009: Pattern Recognition and Computer Vision, 2009.

[51] Hassaballah, A. A. Abdelmgeid, and H. A. Alshazly,“ Image Features Detection, Description and Matching,” *Image Feature Detectors and Descriptors Studies in Computational Intelligence*, pp. 11–45, 2016.

[52] A. Kumar, “Image Retrieval using SURF Features,“ M. S. thesis, Dept. Comp. Science and Eng., Thapar Univ., India, 2011.

[53] H. Bay, T. Tuytelaars, and L. V. Gool, “SURF: Speeded Up Robust Features,” *Computer Vision – ECCV 2006 Lecture Notes in Computer Science*, pp. 404–417, 2006.

[54] H. Shuo, W. Na, and S. Huajun, “Object Tracking Method Based on SURF,” *AASRI Procedia*, vol. 3, pp. 351–356, 2012.

[55] J. J. Anitha and S. M. Deepa, “Tracking and Recognition of Objects using SURF Descriptor and Harris Corner Detection,” *International Journal of Current Engineering and Technology*, vol. 4, no. 2, Apr. 2014.

[56] “OpenCV,” *OpenCV*. [Online]. Available: https://opencv.org/. [Accessed: 19-May-2019].

[57] S. Brahmbhatt, Practical OpenCV, Apress, Berkely, 2013.

[58] A. R. Smith, “Color gamut transform pairs,” *ACM SIGGRAPH Computer Graphics*, vol. 12, no. 3, pp. 12–19, 1978.

[59] W. Schwarz, W. B. Cowan, and J. C. Beatty, “An experimental comparison of RGB, YIQ, LAB, HSV, and opponent colour models,” ACM Trans. Graph, vol. 6, pp. 123–158, Apr. 1987.

[60] “Miscellaneous Image Transformations,” *Miscellaneous Image Transformations - OpenCV 2.4.13.7 documentation*. [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html. [Accessed: 19-May-2019].

[61] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, First Edition. O'Reilly Media, 2008.

[62] “Smoothing Images,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.1.0/d4/d13/tutorial_py_filtering.html. [Accessed: 19-May-2019].

[63] “Image Thresholding,” *OpenCV*. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html. [Accessed: 19-May-2019].

[64] G. Stockman and L. G. Shapiro, Computer Vision, First Edition. Prentice Hall, 2001.

[65] Otsu, “A Threshold Selection Method from Gray-Level Histograms,” IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62–66, 1979.

[66] “Image Thresholding,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html. [Accessed: 19-May-2019].

[67] “Template Matching,” *Template Matching - OpenCV 2.4.13.7 documentation*. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html. [Accessed: 19-May-2019].

[68] M. J. Swain and D. H. Ballard, “Indexing via color histograms,” Proceedings Third International Conference on Computer Vision, 1990.

[69] B. Schiele and J. L. Crowley, “Object recognition using multidimensional receptive field histograms,” Lecture Notes in Computer Science Computer Vision — ECCV 96, pp. 610–619, 1996.

[70] J. A. Feldman and Y. Yakimovsky, “Decision theory and artificial intelligence: I. A semantics-based region analyzer,” *Artificial Intelligence*, vol. 5, no. 4, pp. 349–371, 1974.

[71] R. Ohlander, K. Price, and D. R. Reddy, “Picture segmentation using a recursive region splitting method,” *Computer Graphics and Image Processing*, vol. 8, no. 3, pp. 313–333, 1978.

[72] Biederman, “Human image understanding: Recent research and a theory,” *Computer Vision, Graphics, and Image Processing*, vol. 32, no. 1, pp. 29–73, 1985.

[73] J.-D. Chang, S.-S. Yu, H.-H. Chen, and C.-S. Tsai, “HSV-based Color Texture Image Classification using Wavelet Transform and Motif Patterns,” *Journal of Computers*, vol. 20, no. 4, Jan. 2010.

[74] B. Devi, N. S. Paul, and J. S. Y, “Robust Statistical Approach for Extraction of Moving Human Silhouettes from Videos,” International Journal on Information Theory, vol. 3, no. 3, pp. 55–64, 2014.

[75] “Histogram Comparison,” *Histogram Comparison - OpenCV 2.4.13.7 documentation*. [Online]. Available: https://docs.opencv.org/2.4.13.7/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html. [Accessed: 19-May-2019].

[76] N. Damodaran, V. Sowmya, D. Govind, and K. P. Soman, “Scene Classification Using Transfer Learning,” *Recent Advances in Computer Vision Studies in Computational Intelligence*, pp. 363–399, 2018.

[77] C. Evans, “Notes on the OpenSURF Library,” *Technical Report*, University of Bristol, Jan. 2009.

[78] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[79] N. Snavely, “Scene reconstruction and visualization from Internet photo collections,” dissertation, 2008.

GCRIS