Yarmouk University

Hijjawi Faculty for Engineering Technology

Computer Engineering Department

Robust Video Object Tracking Using Open Source Tools

A Thesis Submitted to

The Department of Computer Engineering

In partial fulfillment of the requirements for the degree of

Master Science of engineering

By

Osama Khaled Rashed Nawafleh

Advisor

Dr. Mohammad Al-Jarrah

Co-Advisor

Dr. Abdel-Karim Al-Tamimi

May, 2017

Robust Video Object Tracking Using Open Source Tools

By

Osama Khaled Rashed Nawafleh

This Thesis was submitted to the department of Computer Engineering in Partial Fulfillment of the Requirements for the Master's Degree of Science in Computer Engineering/Industrial Automation, Yarmouk University, Irbid, Jordan.

Approved by:

Dr. Mohammad Al-Jarrah (Chairman) _____
Associate Professor, Computer Engineering, Yarmouk university

Dr. Abdel-Karim Al-Tamimi (Member) _____
Associate Professor, Computer Engineering, Yarmouk University

Prof. Faruq Al-Omari (Member) _____
Professor, Computer Engineering, Yarmouk University

Dr. Ahmad Al-Mosa (Member) _____
Associate Professor, Communication Engineering, Yarmouk University

DEDICATION

I dedicate this thesis to my parents, for their unconditional love and support. Without their efforts, I would not be who I am. Their incessant encouragement helped me to overcome the most stressful moments and complete my Master's degree.

I couldn't have done any of this without your unconditional love and support.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisors Dr. Mohammad Al-Jarrah and Dr. Abdel-Karim Al-Tamimi, for introducing me to one of the marvelous world of computer vision. Their support, guidance, and patience through this journey were endless. I would like to thank my thesis committee members Dr. Faruq Al-Omari and Dr. Ahmad Al-Mosa for all of their discussion, ideas, and feedback has been absolutely invaluable. I would also thank my close friends for their willingness to listen to my misgiving and concerns and their thoughtful suggestions

Last but not least, I would like to thank my family and friends for believing in me and cheering me up when I needed it the most.

Table Of content

List of Tables

List of Figures

Glossary

ABS	Appearance-Based Shape-Filter
ACAT	Augment Color Attributes Tracker
ACT	Adaptive Color Tracker
aStruck	Scale adaptative Struck tracker
BDF	Best Displacement Flow
BRIEF	Binary Robust Independent Elementary Feature
BRISK	Binary Robust Invariant Scalable Keypoints
CAMSHIFT	Continuously Adaptive Mean Shift
CMT	Consensus-based Matching and Tracking
DGT	Dynamic Graph based Tracker
DSST	Discriminative Scale Space Tracker
DynMS	Dynamic Mean Shift
eASMS	Enhanced Scale Adaptive MeanShift
EDFT	Enhanced Distribution Fields for Tracking
FoT	Flock of Trackers
FREAK	Fast Retinal Keypoints
FRT	Fragment Tracking
FT	Fragments-based Tracking
HT	Hough Track
IIVTv2	Initialization Insensitive Visual Tracker Version 2
IMPNCC	Improved Normalized Cross-Correlation Tracker
IPRT	Iterative particle repropagation tracker
IVT	Incremental Learning for Robust Visual Tracking
LGT	Local-Global Tracking
LM	Learn Match
LT-FLO	Long Term Featureless Object Tracker
MatFlow	Matrioska Best Displacement Flow
MCT	Motion Context Tracker
MIL	Multiple Instance Learning Tracking
OGT	Online Graph-based Tracking
ORB	Oriented Fast and Rotated BRIEF
PLT 13	Single scale pixel based LUT tracker (2013)
PLT 14	Size-adaptive Pixel based LUT tracker (2014)
SB	Semi-supervised online Boosting
SIFT	Scale-Invariant Feature Transform
SIR-PF	Sequential Importance Re-sampling Particle Filter
STRUCK	Structured output Tracking
TLD	Tracking-Learning-Detection

Abstract

Nawafleh, Osama Khaled. Robust Video Object Tracking Using Open Source Tools, MSc. Thesis, Yarmouk University, 2017 (Advisor Dr. Mohammad Al-Jarrah, Co-Advisor Dr. Abdel-Karim Al-Tamimi)

Visual tracking is one of the most important topics of research in the field of computer vision. Most of the common tracking methods are based on detecting the tracked object then applying a proper method of tracking. This type of object tracking allows tracking of a specified object with just one initialization by defining the object to be tracked in the first frame. The target of this thesis is to propose a new solution for tracking a specified single targeted object with no pre-trained data. In this work, we use the binary descriptor FREAK to describe the object features keypoints which are detected by the FAST detector. Afterward, we use the pyramidal Lucas-Kanade optical flow estimation method to track the detected keypoints in the subsequent frames. To accurately localize the tracked object in every frame, we use the location of all the neighboring points to estimate a third point to be considered as the object center. Agglomerative cluster approach is used to group the considered points as object centers, and then we compute the center of the resulted cluster representing the objects center.VOT2014 and Vojirtom datasets are used as benchmarking datasets to test our object tracking algorithm accuracy and robustness. Moreover, we conducted comparisons of our object tracking system with the current state-of-the-art object trackers. The results shows that our algorithm achieved 5.9 accuracy and robustness average values in sequence pooled experiment, and 6.29 accuracy and robustness average values in per-attribute experiment of VOT2014 dataset. Our algorithm achieved the highest recall value of 0.882 among the stat-of-the art trackers, and the precision value of 0.9115 in Vojirtom dataset

Keywords: online-object tracking, computer vision (CV), binary descriptors, optical flow

Chapter One

Introduction

The visual cortex of the human brain analyzes the visual information which is sensed by the retina. Objects are located and identified by the output analysis of the brain[1]. Computer vision is the science of mimicking human brain analysis for visual images. The information about physical objects based on camera images is extracted by the mathematical techniques developed by researchers in computer vision [2]. Computer vision (CV) techniques are applied in the fields of optical character recognition, quality inspection, robot guidance, scene reconstruction, object tracking and object categorization [3]. Object tracking is one of the most active domains of research's in computer vision, where methods and techniques are studied to estimate the locations of targets in subsequent video frames [4]. The generation of high-powered computers, and the availability of high quality for inexpensive video cameras increased the need for automated video analysis and to apply object tracking algorithms in automated surveillance, automatic annotation of video data, human-computer interaction, traffic monitoring and vehicle navigation [5].

To perform object tracking, the algorithm analyzes the subsequent video frames and outputs the movement of targets between frames. Several algorithms are published, where each algorithm has strengths and weakness.

Most of the published algorithms supposed the motion and the appearance of the object are smooth, and there are no abrupt changes [5]. Although, objects can be anything cleared to be tracked in the first frame, like humans on roads, boats in rivers,

and vehicles, tracked objects are exposed to many changes, like occlusions, changing the size of the object, and background illumination.

Unchanging objects shape and appearance are suitable to simplify the object tracking methods, where the representations of the tracked objects in subsequent frames can be represented in many shapes including points, Geometric shape, Contours, and Articulated. In points, objects are represented by a point or group of points which it occupies a small region in the image. In geometric shape, objects are represented by rectangle or ellipse. It helps the non-rigid objects to be tracked. In contours, the representation of the tracked object defines the boundary of that object. In articulated, where objects are composed of body parts that are held together with joints [5]. A critical role for objects tracking is to select the right features of the objects, where it can be easily distinguished in the feature space.

Color and edges are good features to track. Color can be used as a histogram based feature, while edges can be used as a contour representation of object's feature. In general, features are chosen depending on the application domain.

Object tracking can be classified into three groups. The first one is point tracking, where the objects appeared in the subsequent frame are represented by points. These points include object position. The second one is kernel tracking, where the objects shapes and representations can be a rectangle or elliptical shape. The third one is Silhouette tracking, where the information of the object color and its background are used in the tracking methods. The classified groups of object tracking depend on the shapes model and density as information to be tracked [5].

Optical flow is a feature that can be used to track an object motion [23]. In general, optical flow describes the feature of points displacement from one frame to

another within subsequent frames. The investigated features must be visible in both frames for getting valid results. The displacement per pixel is given by *x* and *y* offsets respective to time. Several factors influence if the displacement vectors can be detected by a specific algorithm or not. Some of them are the surface color, its texture and the properties of the material when interacting with light and illumination [7]. In related research area, a lot of effort has been put to describe these factors in the salient image locations in a manner that is invariant to scaling, rotation and illumination. These features called local features. The prosperity of the local descriptors to describe the objects within the subsequent frames in a manner that is invariant to scale, rotation and illumination changes has been raised in recent years. Which makes the research of object tracking is very fertile domain.

1.1 Research Problem

In live videos, Object tracking and the associated problems of feature selection, object representation, dynamic shape, and motion estimation are very active areas of research [6]. Many objects tracking algorithms have been investigated and implemented in real time [7]. Real applications nowadays require highly autonomous algorithms to detect and track objects in a scene, which gives a possibility to increase the level of autonomy of intelligent algorithm. The autonomy is identified as ability to extract useful characteristics data from frames environments around the identified objects, and to make a proper tracking methodology in robust manner to track the objects in subsequent frames where the object may appear.

There is no general algorithm can handle all scenarios [7]. The goal of our thesis is to track a single online targeted object within subsequent frames of video to meet efficient and stable objects tracker. Our proposed algorithm has to be tailored to meet

requirements, and the needs of its task. Object tracking defines the state of the target object for each frame I_k as bounding box, ellipses, points or shapes [4].

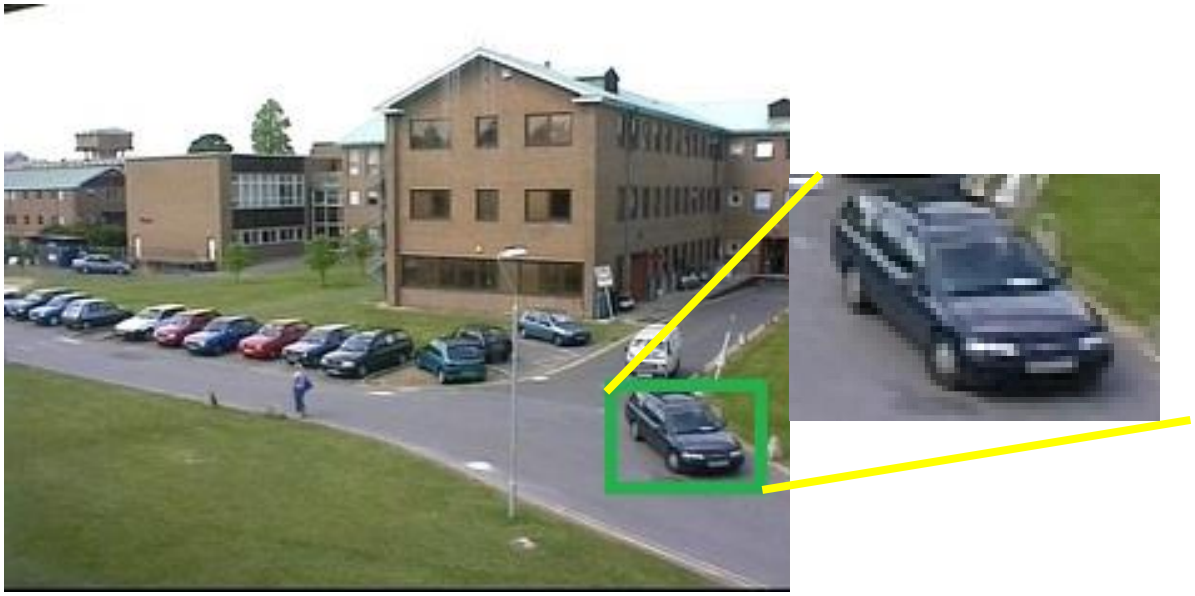

Figure 1.1: The state of target represented as bounding box. The image is part of Vojirtom dataset [38]

Figure 1.1 shows a bounding box around the object of interest, where the state of the target object in this case consist of the upper left corner of the rectangle (x_1,y_1), and the bottom right corner (x_2,y_2).

In this thesis, we focus on **semi-automated** tracking, where the user input is required to initialize the tracking process. While inaccurate foreground object extraction due to shadows, reflectance and occlusions makes the object tracking is a difficult research problem. The design of the tracker should successfully extracts the object location under different conditions and it should be computationally fast with less memory to meet the real-time performance.

1.2 Motivation

Understanding the activities of moving objects in a scene by the use of video is both a challenging scientific problem and a very fertile domain with many promising applications like recognition of interactions between humans, and recognition of specific human activity for instance. Our motivation in studying this problem is to create a visual object tracking system to track a real-time online single targeted object in different conditions in a robust manner.

1.3 Contribution

Object tracking is a very fertile domain with many promising applications. In this thesis, we designed and developed an approach for tracking an online moving object in a live camera or recorded video. In our approach, we built an algorithm to detect the object we want to track and extract their features using binary detectors and descriptors. To track the object efficiently, we used optical flow in both forward and backward estimations. The first main contribution of this thesis is combining optical flow with binary descriptors and color histograms matching in order to introduce a robust object tracking algorithm. The second contribution is finding tracked object centers by using the locations of every neighboring points to estimate a third point which can be considered as an object center candidate.

1.4 Thesis Structure

This thesis is divided into seven chapters. An outline of the remaining chapters is as follows:

- Chapter 2 provides background and literature review.

- Chapter 3 discusses the general approach of object detection and description.
- Chapter 4 discusses the tracking method based on the estimation of optical flow and the color tracking using Mean-Shift algorithm.
- Chapter 5 discusses the methodology we used and it describes what happens during the tracking of an object.
- Chapter 6 shows the experimental results on the used test data and also shows the comparisons of our results to other tracking methods.
- Chapter 7 concludes the thesis and lists our intended future work.

Chapter Two

Background and Literature Review

2.1 Introduction

Image description in object tracking is an important field of computer vision that has a varied approach. Examination of various object tracking approaches leads to an explicit understanding of their advantage and their weakness. Also, studying and understanding the various descriptors, guides us to a clear approach where it might be good for tracking. This chapter discusses most active general object tracking approaches as well as histogram of oriented gradient and binary descriptors as they are the local descriptors family.

2.2 Histogram of Oriented Gradient (HOG)

HOG is a family of descriptors used in computer vision for the purpose of object detection phase, where each pixel in the image is analyzed. Scale-Invariant feature transform (SIFT) and Speed-Up robust features (SURF) are members of this family. SIFT introduced in 2004 by Lowe [14]; it provides a tool for extracting distinctive features from images. It consists of four computation stages:

1. Scale-space extreme detection

2. Keypoint localization

3. Orientation assignment

4. Keypoint descriptor

Scale-space extreme detection phase uses a Different-of-Gaussian (DOG) [79] to identify the interested points that are invariant to scale and rotation. Image pyramid is built and each layer is filtered using Gaussian.

In the second stage, a detailed model for each candidate is used to determine location and scale. Key points are selected based on measures of their stability: the ones which have low contrast or are poorly localized on an edge are eliminated from the previously built list. In the next step each key point is given a consistent orientation based on local image gradient directions [14].

The same gradients are then used to create the key point descriptors. The largest orientation values in the histogram are used as the main orientation of the features descriptors as shown in Figure 2.1.

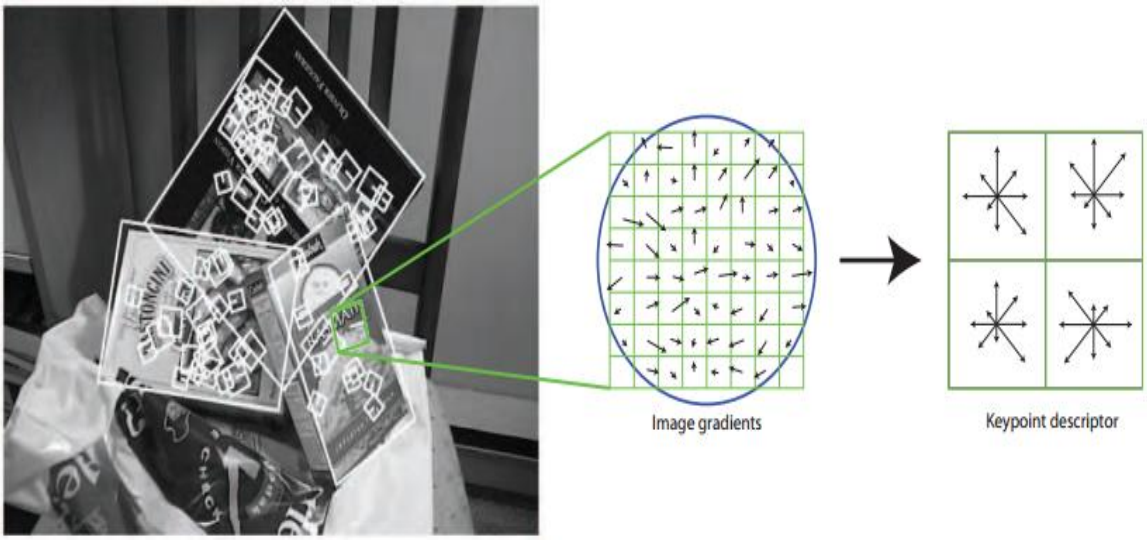

Figure 2.1: SIFT descriptors computation [14].

Speeded-up robust features (SURF) algorithm, proposed by Bay and Tuytelaars [15] is based on the same principles as SIFT, but it has a different scheme and provides better result in time. SURF is not fully invariant but it allows for considerable affine change as it is shown in Figure 2.2.

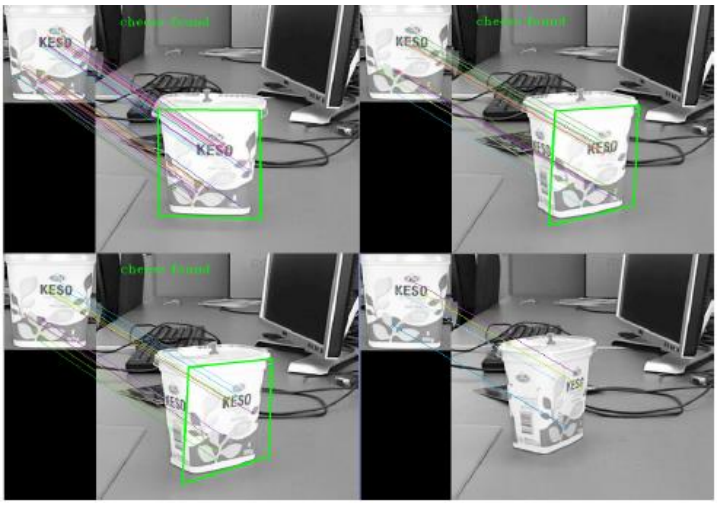

Figure 2.2: SURF behavior with object transformation [15].

In Figure 2.2, we can see in the top left image that the surf detector can recognize the object, in the second top right and in the third bottom left image the detector still recognize the object although there is considerable affine change in the pose of the object, but when there is more change in the pose, the SURF detector cannot recognize the object.

SURF detection's core is based on the determinant of the Hessian matrix [15], which is square matrix of second-order partial derivatives of a scalar field that describes the local curvature of a function for many variables. In SURF, different scales of Gaussian masks are used to build up the pyramid.

2.3 Binary Descriptors

The histogram of oriented gradient (HOG) family costs high computation time and very slow, because the gradient of each pixel need to be computed. Even the Speeded-up robust features (SURF) algorithm speeds up the computation using integral images, it is still not fast enough. In this situation binary descriptors come in handy.

Most of the information is encoded as a binary string using only comparison of intensity images. The distance measure between two binary strings is done using hamming distance, where the matching between two patches of descriptors is done using single instruction. Binary Robust Independent Elementary Features (BRIEF) [72], The Oriented fast and Rotated BRIEF (ORB) [67], The Binary Robust Invariant Scalable Keypoints (BRISK) [19] and The Fast Retina Keypoint (FREAK) [20] are members to binary descriptors.

Binary Robust Independent Elementary Features (BRIEF) descriptor is one of the early works in the binary descriptors work. BRIEF works by comparing the same set of pixels pairs for each patch that it describes. The sampling pairs are chosen randomly. If the first pixel intensity is larger than the second one, BRIEF writes 1 in the final descriptor and 0 otherwise.

The Oriented fast and Rotated BRIEF (ORB) descriptor was built upon BRIEF by adding the rotation invariance. by using the first order moment, the patch orientation is measured.

The Binary Robust Invariant Scalable Keypoints (BRISK) descriptor uses a hand crafted sampling pattern instead of randomness. BRISK has two sampling pairs, the long-distance sampling pairs used to estimate the orientation of the patch, and the short-

distance pairs, which are used to build up the descriptor itself through pixel intensity comparison.

The Fast Retina Keypoint (FREAK) descriptor has a hand crafted sampling pairs similar to BRISK descriptor. The retina structure motivates the sampling pattern to have exponentially more sampling points toward the center.

2.4 Related Works

The definition of object tracking is identifying a region from a frame and then seeking for it frequently in subsequent frames. This leads to a problem of specifying the matching criterion. Thus, many different methods have the effort into adequately finding and matching specific objects of frames in the field such as Mean Shift, Point tracking, Discriminative Foreground-Background methods and Template Methods.

Mean shift is an established algorithm, which exists to form a procedure which helps to find the maxima in a density function. Using mean shift procedure, the nearby points are weighted, and then the estimation of the maxima location of the density is iteratively repeated until convergence. Adaptive Mean Shift (CAMSHIFT)) tracker [8] is similar to Mean shift algorithm, which it attempts to find the peak of the distribution. CAMSHIFT algorithm is designed to be able to change the observation window dynamically, thus making it able to track. CAMSHIFT suffers from extending to similar objects founded in the same searching window while the algorithm tries to track a single object. A solution to this drawback in the CAMSHIFT algorithm is to combine Kalman filter which predicts the object location [8]. Figure 2.3 shows the experimental result between the legacy CAMSHIFT algorithm and the new proposed algorithm after combining Kalman filter with CAMSHIFT algorithm [8].

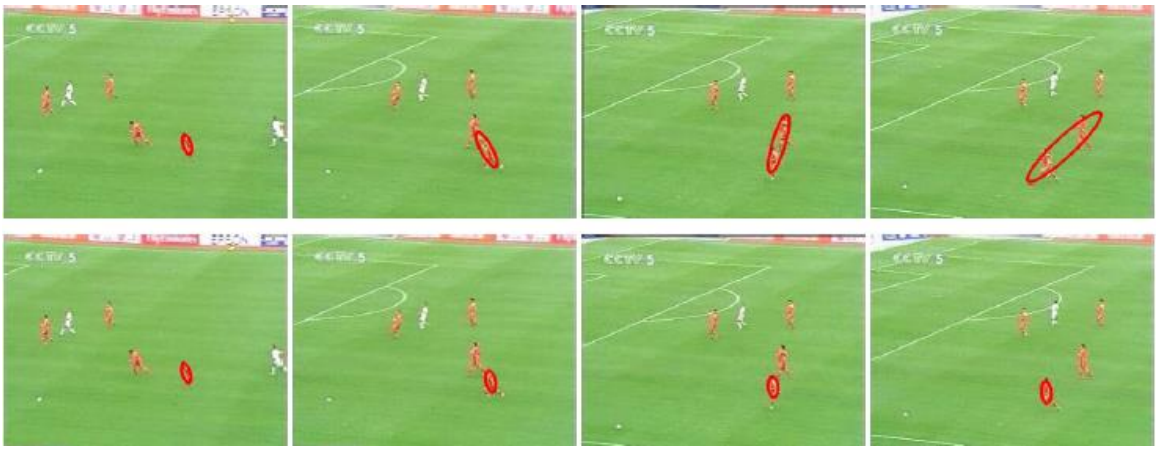

Figure 2.3: Player tracking, first row- CAMSAHIFT tracker, Second row- Kalman Filter with CAMSHIFT algorithm [8].

Points' tracking is a method of tracking where the object's correspondence points are responsible for modeling the object throughout subsequent frames [5]. Point's tracking method served as an inspiration to this thesis. In general, point's trackers can be considered as deterministic [69] or probabilistic [70]. Deterministic methods employ assumption about the shape of the object is not changing once it has been defined. Probabilistic methods attempt to model the properties of the object such as speed and acceleration. In [9], binary descriptor BRISK [19] is used to describe the detected object's points, where the tracking problem starts by defining a bounding box around the specified object. In [9], Pyramidal Lucas-Kanade [24] is used to estimate the optical flow of the detected points over subsequent frames. Matrioska tracker [10] is composed of detector and learning modules. The detector module uses multiple of keypoint-based method like ORB [67], FREAK [20], SIFT [14] and more together to correctly localize the object exploiting the strengths of each method where only sufficient keypoints will be used to localize the object. The learning module is used to update the training pool used by the detector localization. Figure 2.4 shows integration of the detector and the learning module in Matrioska tracker.

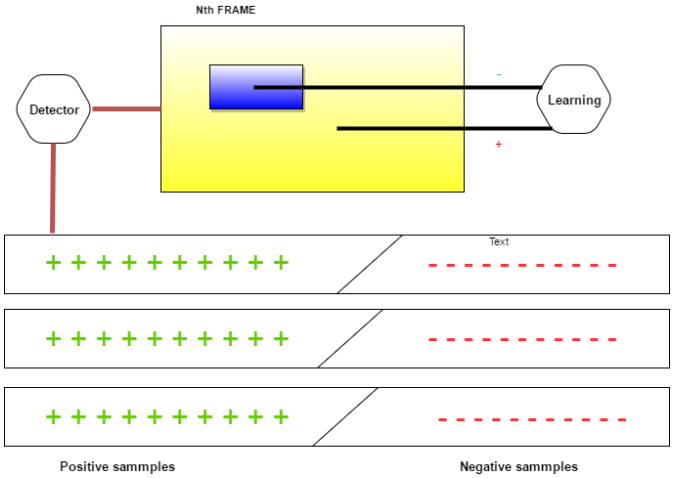

Figure 2.4: Matrioska tracker, detector and learning module [11].

Templates tracking methods create templates for specified objects exist in the subsequent frames. Tsagkatakis and Savakis used descriptors to make the template invariant to scale and rotation [71]. Matching method vary depending on the information stored inside the template.

Some methods used classifiers, which discriminate between the foreground and the background of the image where small changes in the object are closer to the foreground than the background. This type of methods is introduced in [59].

Several state-of-arts trackers decompose the targeted object into parts, these trackers are splitted into Keypoints based trackers like CMT [9], IIVTv2 [74], LT-FLO [55], LGT [49], PT+[78], DGT [44], and OGT [52]. Also, Flock tracker approach like BDF [50], FOT [54], FRT [57].

There are tracker approaches use global generative visual models for target localization EDTF [53], qwsEDTF [48], ACAT and VTDMG [73], eASMS [46], NCC[56], and IVT [58]. HMM-TxD [76] merge flock of trackers with Mean-Shift.

Chapter Three

Object Detection and Description

3.1 Introduction

This chapter shows the method we employ for object detection and description. Object detection enables the re-initialization of the tracker to track the object. While the tracker depends on the location of the object in the previous frame, the object detection applies a searching strategy in order to find the object. Many different object detection methods are applicable, but picking the most appropriate one or making a certain one to meets all object detection conditions are very difficult. The use of local features algorithms helps to achieve object detection within our thesis requirements.

3.2 Local Features

Local features are the basic option on which objects' detector is built. The idea is to describe a parsed image with a set of points called Keypoints. Each keypoint can be described as an image pattern that diverges from its immediate neighborhood point. Color, texture, and intensity are features where algorithms are able to identify the local structures of keypoints in the image [12]. Features should be unique for each object to avoid misconstruction with other image structures. To detect a partially occluded object, a vast number of feature regions are needed to cover the target [12].

The detected keypoints in an image should be equivalent to the detected keypoints to the geometrically transformed version and located in corresponding locations. The

detector required for this thesis purpose needs to be invariant to all or any quite transformations like rotation, translation and scaling.

3.2.1 Features from Accelerated Segment Test (FAST) Keypoint Detector

A robust detected keypoints in an image must be the same in a transformed version if exist. Figure 3.1 shows the transformations that must be handled by the keypoints detector. The beginning of feature detection with the algorithm of Harris and Stephen which called later "Harris Corner Detector I" was in 1988 [17], this method describes the detection and extraction of robust feature points or a corners in any image. But because of detecting corners only where each corner was isolated from the others, the algorithm was suffered from a lack of connectivity for feature points, which would have a limitation for obtaining descriptors such as surfaces and objects.

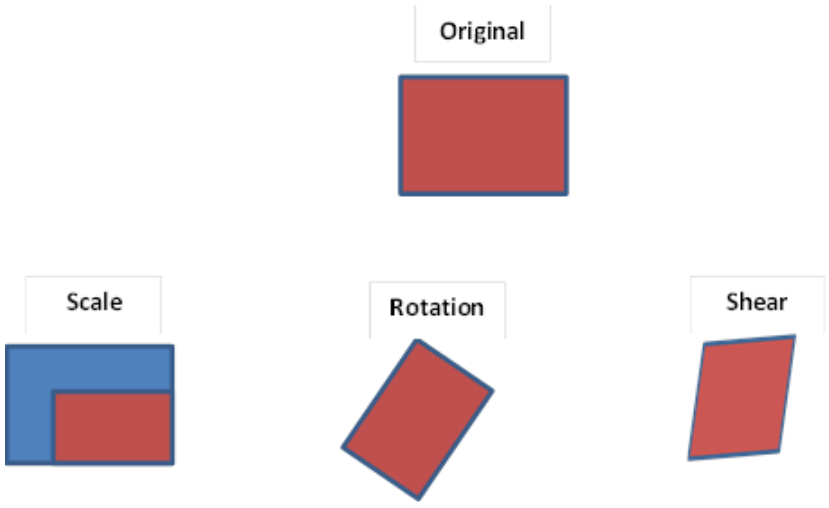

Figure 3.1: Object transformations.

To overcome the limitation of Harris Corner Detector I, Harris published a new state of feature detection. He combined the isolated corners detected with the Harris

detector with a corresponding connection edge. After that several methods for corners and edge detection were published based on Harris detectors. Features from accelerated segment test (FAST) [18] is one of the methods for corner detection which is relying on the Harris detector mechanism [16]. Trajkovic and Hedley stated that to enable feature point matching from a detected corner, the corner detector should satisfy the following criteria: (a) Consistency, detected positions should be insensitive to the variation of noise; (b) Accuracy, corners should be detected as close as possible to the correct positions; (c) Speed, even the best corner detector is useless if it is not fast enough. The main contribution of FAST was the speedup of the computation required in the detection of corners. FAST mechanism is described in Algorithm 3.1.

Algorithm 3.1: Features from accelerated segment test (FAST) mechanism

Input : Current image (img_1), Image_pixels(p_i)

Select_pixel (p_i) ⟶ I_p

Set_Intensity ⟶ T

Define_circle_radius_3 pixels ⟶ tested_ pixels (P_1,……...P_{16})

For *all* p_i *do*

If (I_1, I_5, I_9,I_{13}) > $I_p + T$ **Then**

"I_p is corner"

Else if (I_1, I_5, I_9,I_{13}) < $I_p - T$ **Then**

"I_p is corner"

Else

"I_p is not corner"

End

End

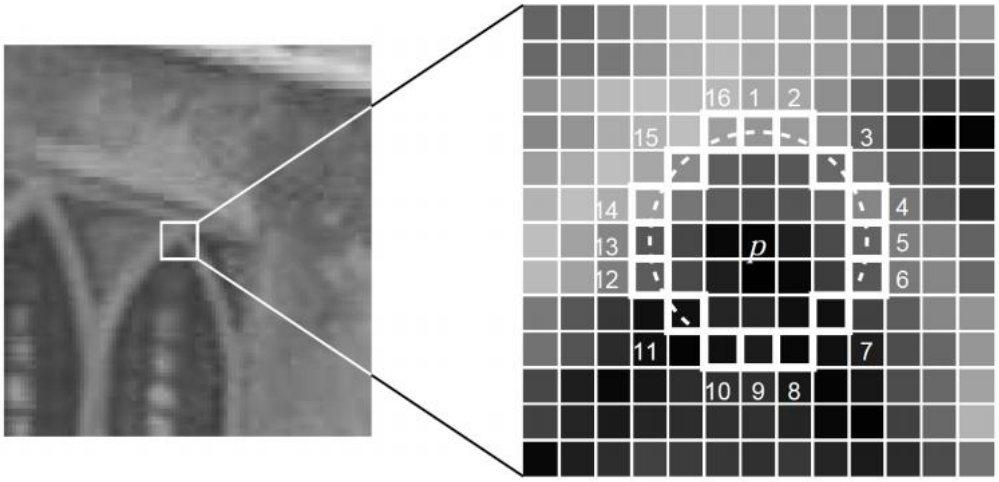

Figure 3.2: Corner detection using FAST [18].

3.3 Binary Local Features Descriptor

The region around the interested point which has been extracted by the detector must be encoded in a feature vector which will be used in the discriminative matching [13]. The most important and distinctive information contents surrounded by the detected salient region are captured by the descriptors. There are two main schools of thoughts for describing the surrounding area around the feature point: local histogram based descriptors which is Histogram of oriented gradients (HOG), and binary descriptors.

SIFT [14] and SURF [15] are two efficient descriptors related Histogram of oriented gradients (HOG) family. They provide a good performance in detecting keypoint and extracting descriptors for each keypoint. Because they depend on gradient histograms, each and every pixel should be analyzed, which cost computational time.

Binary descriptors provide the ability to encode all the information related to the surrounding points to the feature point as binary strings. They are great alternative to

their competitors, because of limited memory they empowering faster computation especially during the development of applications which target embedded vision.

Binary descriptors consist of a sampling pattern. Numbers of pairs of points are chosen on the pattern. The intensity value of each point in the pair is compared with its matched one. If the first result is larger than the second, the value “1” is written in the string, “0" otherwise. When all the pairs have been analyzed, the information describing the area around the key point will be encoded in a string of “0” and “1”. The orientation compensation is a mechanism where the orientation of the interesting area is calculated relatively to some intrinsic feature of the area itself. The chosen pairs are rotated to that same angle, before evaluating the intensity to make sure that the binary descriptor will be rotation invariant. BRISK [19] and FREAK [20] are two binary descriptors. We used these descriptors to describe keypoints in our thesis work. Section 3.3.1 and 3.3.2 show these descriptors in detail to give an overview about their function.

3.3.1 Binary Robust Invariant Scalable Keypoints (BRISK)

BRISK [19] is a binary descriptor, is invariance to global illumination, scale and rotation. To achieve scale invariance, it implements a pyramid based method which presuppose that each key-point is the maximum score when compare to its neighbor in the pyramid level above and below it. The scale is not constant factor at all scales. The pyramids have extra layers in between the main layer. This pyramid can be seen at Figure 3.3. BRISK concatenating the results of the intensity comparisons to create a binary string as mentioned. BRISK differ from other binary descriptor by having a hand-crafted sampling pattern. The sampling pattern collected from concentric rings as shown in Figure 3.4. When each sampling point is analyzed which is the blue point in

the Figure 3.4, it takes a small area around it with a radius equal to the red circle which it illustrates the standard deviation of the Gaussian filter applied to each sampling point.

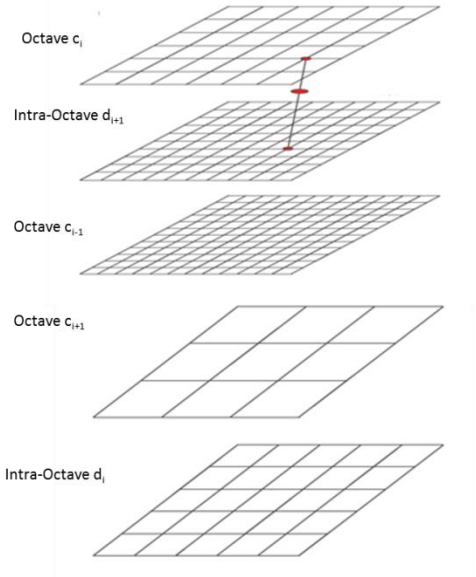

Figure 3.3: BRISK scale pyramids [19].

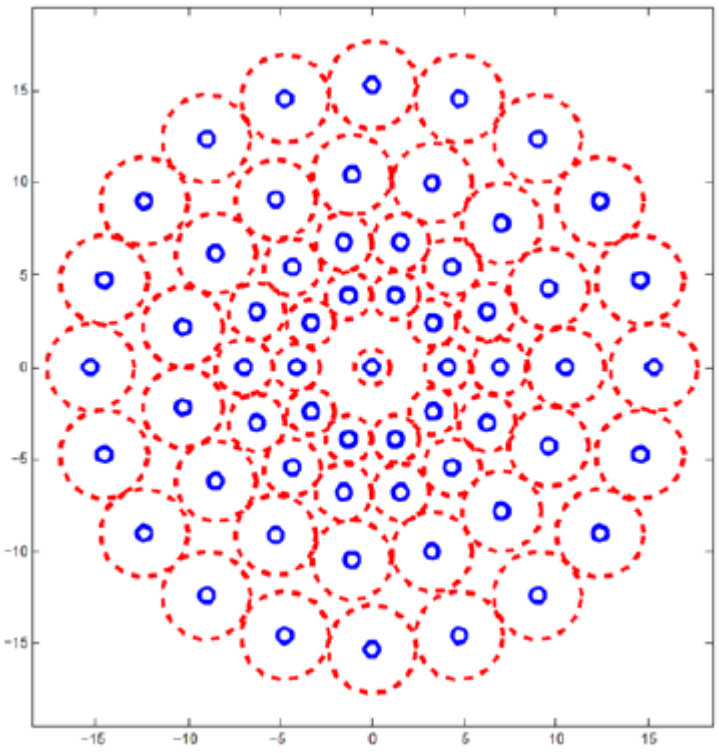

Figure 3.4: BRISK sampling pattern [19].

The hand-crafted sampling pattern creates comparisons systematically via equation (3.1) where A represents all possible combinations:

$$A=\{(p_i, p_j) \in R^2 \times R^2 | \, i < N \wedge j < i \wedge i \, j \in N\} \quad (3.1)$$

BRISK defines pairs as two subsets: *short-distance pair* which is used to compute the intensity comparison to build up the descriptor where the distance is below a certain threshold Δ_{max}.

While *long-distance pair* is used to determine the orientation where the distance is above a certain threshold Δ_{min}. The two threshold are sets as $\Delta_{min} > \Delta_{max}$ so that no short-pair is also long-pair. Figure 3.5 shows the two thresholds [21].

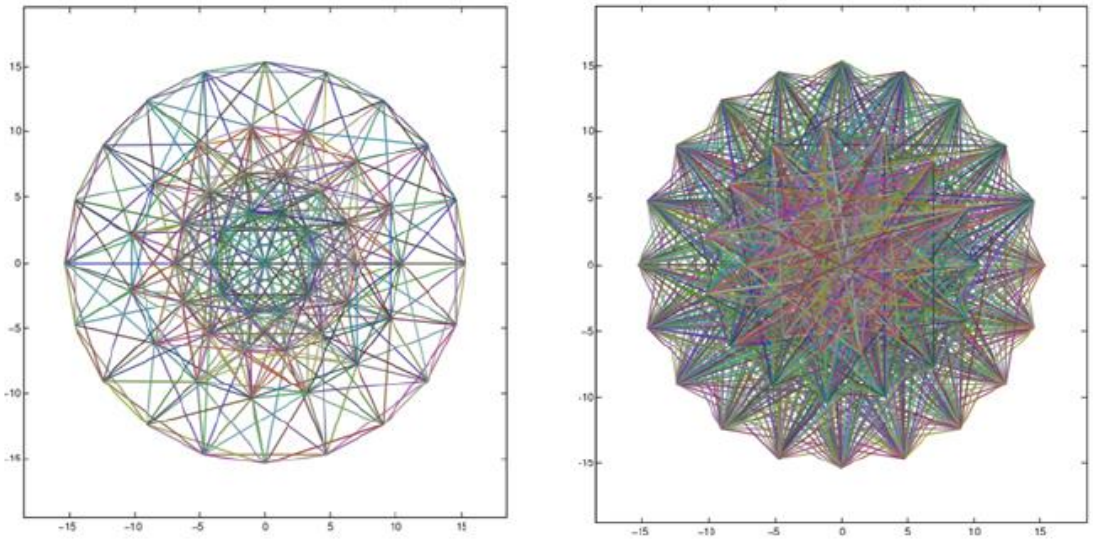

Figure 3.5: Short distance pair (512) on the left and Long distance pair (870) on the right [21].

The orientation is computed via applying a local gradient on the long-distance pair, while the rotation is computed by rotating the short-distance pair at same angle as the key point orientation [21]. The gradient for orientation is computed according to equations (3.2), (3.3) where $g(p_i, p_j)$ is the local gradient and p represents a pixel:

$$g(p_i, p_j) = (p_j - p_i) * \frac{I(p_j, \sigma_j) - I(p_i, \sigma_i)}{\| p_j - p_i \|^2} \tag{3.2}$$

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \sum_{(p_i, p_j) \in L} g(p_i, p_j) \tag{3.3}$$

As mentioned, the binary string is built through performing intensity comparison. BRISK takes the short-distance pairs then it rotates the pairs by the orientation computed earlier and makes the comparison as following equation (3.4):

$$b = \begin{cases} 1, & I(p_j^\alpha, \sigma_j) > I(p_i^\alpha, \sigma_i) \\ 0, & otherwise \end{cases} \tag{3.4}$$

where b is the bit value results from comparison, $I(p_j^\alpha, \sigma_j)$ is the intensity of pixel p at scale σ.

For each short-distance pair, it takes the smoothed intensity of the sampling points and checked whether or not the first point in the pair is larger than the second point. It writes 1 in the corresponding bit of the descriptor if it does, and otherwise 0 [19].

The result is a 512-bit long string that is robust to noise, scale, illumination, and rotation. The matching is via the Hamming Distance which is using one single instruction so it does not suffer any extended matching times.

3.3.2 Fast Retina Key point (FREAK)

FREAK [20] descriptor is similar to BRISK [19] that make use of hand-crafted sampling pattern. It uses a pattern similar to the human retinal sampling grid [20] where the center has the higher density of receptive area. Figure 3.6 represents receptive field's distribution over the retina in a human eye.

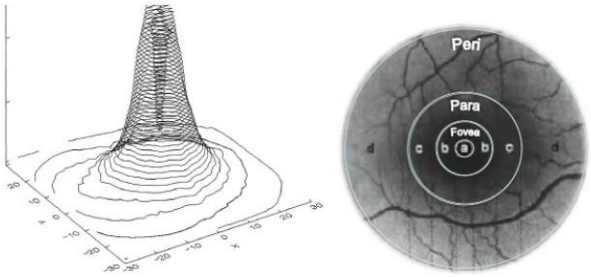

Figure 3.6: Receptive fields distribution over the retina in a human eye [20].

FREAK suggests using retinal sampling grid which is also a circular like BRISK with the difference of the center has a higher density of points and drops exponentially as shown in Figure 3.7. Each sampling point must be smoothed by a Gaussian filter to make it less sensitive to noise. The radius of the red circle illustrates the standard deviation of the Gaussian kernel applied on the sampling points.

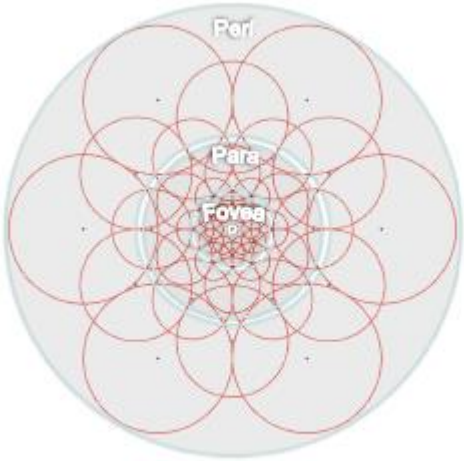

Figure 3.7: FREAK sampling pattern [20].

The redundancy exists in the receptive field of retina has been applied to FREAK algorithm so that it allows to reduce the number of receptive fields.

Coarse-Fine approach is used in the resulting pairs. The primary matched is authorized to the first analyzed pairs which they compare sampling points in the outer

rings of the pattern while last pairs compare points in the inner rings of pattern. This is done similarly to the human eye, where the estimation of the location of the object of interest is done in perifoveal, and then the fovea area is responsible to the validation process [22]. The sampling pairs are illustrated in the following Figure 3.9.

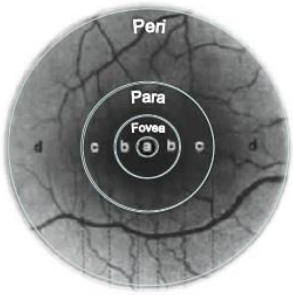

Figure 3.8: Human eye retina [22].

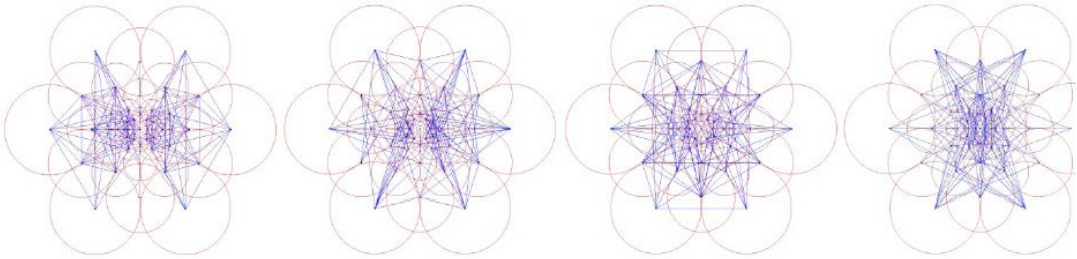

Figure 3.9: FREAK's sampling pairs resulting from Coarse-to-fine analysis [20].

Coarse-fine structure give the FREAK method advantage in speed up the matching using cascade approach: the first 128 bit is compared when matching two descriptors. If the distance is similar to the threshold, the next 128 is compared. Else stop the comparison. Figure 3.10 shows how matched FREAK descriptor is look like in specific object.

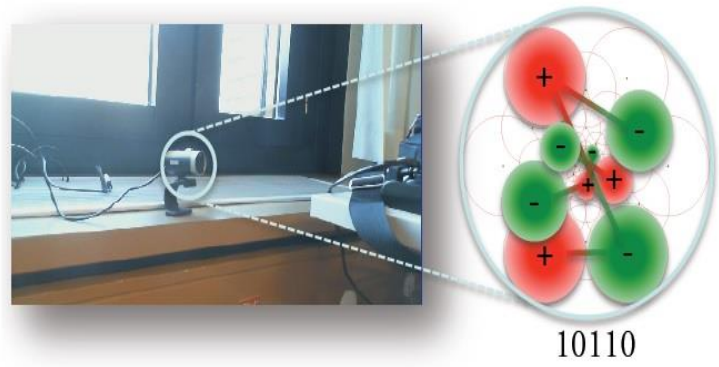

Figure 3.10: Matched FREAK descriptor [20].

FREAK computes the orientation in similar way that BRISK [19] does. However, FREAK doesn't use long-distance pair like BRISK, it uses predefined 45 symmetric sampling pairs

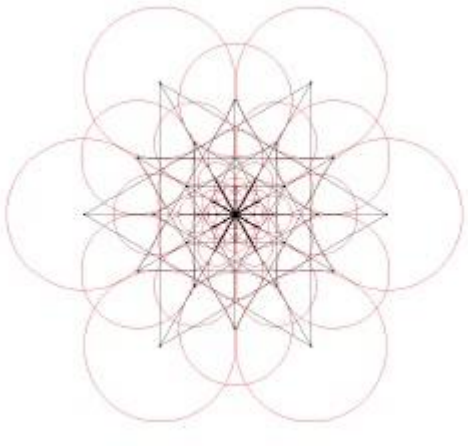

Figure 3.11: FREAK orientation pairs [20].

Equation (3.5) below computes the local gradient *(O)* to allow FREAK descriptor to estimate the rotation of keypoints, where G is the set of all the pairs, M is the number of pairs, and $p_o^{r_i}$ is the 2D coordinate vector of the center of the receptive field:

$$O = \frac{1}{M} \sum_{P_o \in G} (I(p_o^{r1} - p_o^{r2})) \frac{p_o^{r1} - p_o^{r2}}{\| p_o^{r1} - p_o^{r2} \|} \tag{3.5}$$

Chapter Four

Object Tracking

4.1 Introduction

In this chapter, we describe the methods we used to track an online object which means no prior information needed for the object. Object tracking can be used by detecting the object in the frame and extract their features to model the object then track the object using its models. But, in our thesis we continuously detect object in new frames and match it with the list from the previous frame. This tracking system needs an initial input, such as initial location of the object or their model, which is output from object detection (chapter 3). Many features used to track an object, such as template, texture and gradient, but in our thesis we rely on tracking the keypoints related to the object. Also, we used Mean-Shift algorithm as the target is modeled by its color histogram, which represents the probability that a particular color appears in the object area.

4.2 Optical Flow

This method of tracking is used to assess the motion between two frames without a prior knowledge about the content of these frames. Optical flow is classified into two types: dense optical flow, and sparse optical flow. Dense optical flow associates a velocity with every pixel in an image. But in practice calculating dense optical flow is not practical because of the computation time it needs [23]. Sparse optical flow, which we used in our thesis, relies on fewer points which represent the object to be tracked.

"Lucas-Kanade" [24] optical flow is one of the sparse optical flow algorithms which used to assess the motion between two frames using fewer points..

4.2.1 Lucas-Kanade

Lucas-Kanade was originally attempts to produce dense results. But because the method is applied on some subset points in the image, it has become an important sparse technique. This algorithm uses a small window surrounding sparse points of interest, but large motion can move points outside the local window. Thus, it becomes impossible for the algorithm to find these points. This problem led to produce "Pyramidal Lucas-Kanade" [25].

The idea of Pyramidal Lucas-Kanade is based on the following three assumptions:

- **Brightness constancy:** the pixel for the object in an image doesn't change during its acquisition period due to lightening or color variation.

- **Small movement:** the pixel present in the previous frame still appears in the current frame where equation (4.1) represents this assumption.

$$I(x + u\delta t, y + v\delta t, t + \delta t) = I(x, y, t) \tag{4.1}$$

where $I(x, y, t)$ is the brightness of the pixel at (x, y) in image captured at time t and $I(x + u\delta t, y + v\delta t, t + \delta t)$ is the pixel after a displacement $(\delta x = u\delta t, \delta y = v\delta t$ at time $t + \delta t$.

- **Spatial coherence:** Neighboring points in a scene belong to the same surface.

Lucas-Kanade uses the sum of square differences (SSD) at equation (4.2) to minimize the differences in the intensity between corresponding pixels over the images.

$$E_{ssd}(u,v) = \sum_i (I(x_i + u_i\delta t, y_i + v_i\delta t, t+\delta t) - I(x_i, y_i, t))^2 = \sum_i e_i^2 \quad (4.2)$$

In our thesis we applied "Pyramidal Lucas-Kanade" i.e coarse-to-fine, because of its benefit when the object moves a long distance. Figure 4.1 shows pyramid Lucas-Kanade, where the same window is applied on several down-sampled versions of the original image.

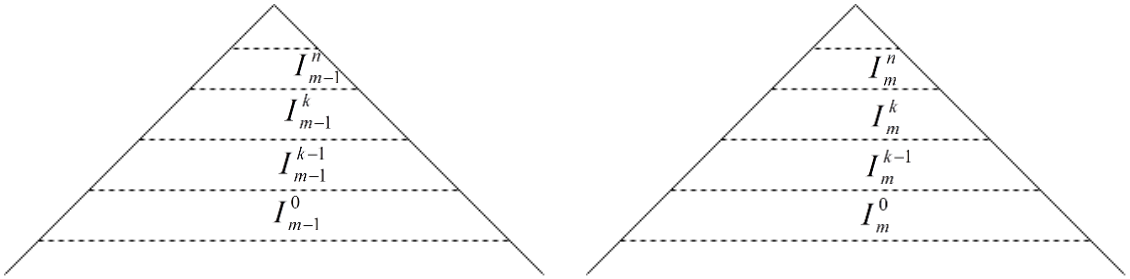

Figure 4.1: Locas-Kanade Pyramids. I_m is the current image, I_{m-1} is the previous image. Computation starts at the top and ends at the bottom of the pyramid [23].

Figure 4.2 shows an example of sparse optical flow, where pyramidal lucas-kanade is used to track selected points inside the bounding box.

Figure 4.2: Frame of image sequence where pyramidal lucas-kanade is applied to follow the green points inside the bounding box. This image is part of VOT2014 Dataset [37]

The window size remains the same over all the pyramid level. Therefore, it is possible to detect large motions. The choice of the number of levels of the pyramid is

therefore dependent on the size of the captured image and on the size of the object that should be tracked [25].

We follow the approach of Kalal et al [26] for recursive tracking. It proposes forward -backward error measure which the tracking of points must be reversible. The proposed error measure is defined as the Euclidean distance ε in equation (4.3) which must be above certain threshold.

$$\varepsilon = \left| p - p^{"} \right| \tag{4.3}$$

Where p is interest pixel and p'' is new location of p. We can find p'' using equation (4.4), where *LK* represents the pyramidal Lucas-Kanade optical flow algorithm.

$$p^{"} = LK(LK(p)) \tag{4.4}$$

When there is full occlusion, the optical flow cannot find the object in the next frame which fails in tracking the object. This drawback can be accomplished using color tracking, the next subsection will describe tracking method using color.

4.3 Object Tracking Using Color

Object tracking using color model is to represent the object in the form of color histogram. This is easy to compute the similarity of two histograms. It is reliable to track after occlusion and it is resist to the change in orientation. Tracking using color model the object in form of color histogram. Color histogram is consistent under changing in translation and motion. The normalized color histogram is defined as:

$$P(u_i \mid Obj) = \sum_{x_j \in \Omega} \delta(b(I_{xj}) - u_i), i := 1:n \tag{4.5}$$

Where $P(u_i \mid Obj)$ is the probability that a color u_i appears in the object, Ω is the area of the object Obj, n in the number of histogram bins, the function $(b(I_{xj})$ maps an intensity I_i to its color bin in feature space, δ is equal 1 at origin and 0 otherwise

After modeling the object, searching technique is required to find the model of unknown probability density function (PDF). Mean-shift [27] is one of the algorithms that search PDF in order to increase the matching scheme.

4.3.1 Mean-Shift

Mean-shift is used to seek the local maxima in a density distribution of data set. It applies hill climbing to a density histogram of the data. Mean-shift ignores the outlier data by processing only the data within a local window and then moving the window. Figure 4.3 shows how the Mean-shift algorithm approach.

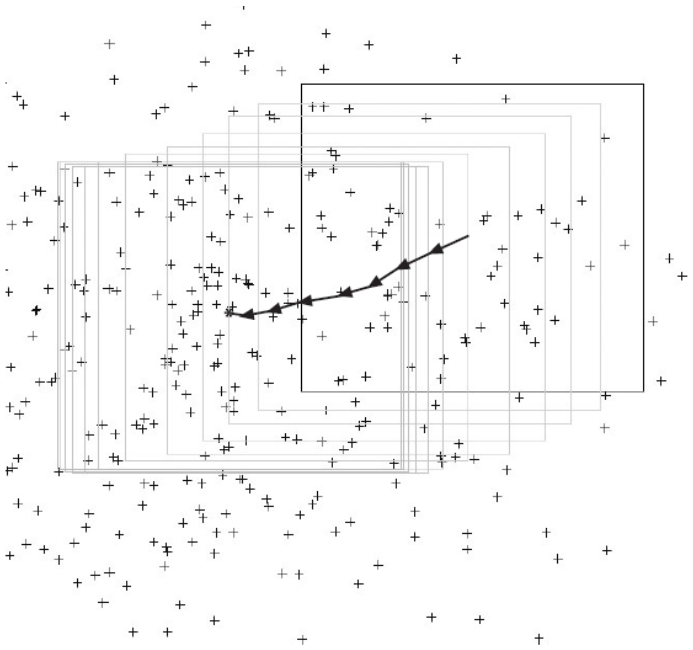

Figure 4.3: Mean-Shift algorithm approach [23]

After defining the initial location of the searching window for the object to be tracked, mean-shift algorithm computes the possibly weighted center of mass for the

window, then it centers the window at the center of mass. The algorithm keeps computing the weighted center of mass and centering the window until the window stop moving.

Bhattacharyya coefficient is used to find the similarity between the target and the model histogram at equation (4.5):

$$\rho[p_u(y_0)q_u] = \sum_{u=1}^{n} \sqrt{p_u(y_0).q_u} \tag{4.6}$$

Where ρ is Bhattacharyya coefficient, q_u is the target color histogram, p_u is the color histogram of the object.

If $\rho > \rho_{th}$ the object location is found at y_0, otherwise the new location is updated by equation (4.6):

$$y_1 = \frac{\sum_{i=1}^{n_h} x_i w_i \left(\| \frac{y_0 - x_i}{h} \|^2 \right)}{\sum_{i=1}^{n_h} w_i \left(\| \frac{y_0 - x_i}{h} \|^2 \right)} \tag{4.7}$$

Where w_i is the coefficient for giving high weight to pixel x_i whose intensity is more similar to the model [27], [68].

Chapter Five

Methodology

5.1 Introduction

This chapter shows our online object tracking system using both live camera streams and recorded videos. Our system is designed to work with non-pre-trained objects, (i.e. real-time objects). Furthermore, our system usage is not only limited to stationary cameras, but works also with recorded video input stream. Moreover, our system can work without skipping any frame that exists in the video. The proposed system is initialized by feeding video imagery from a camera or a recorded video. Our system is able to work on both color and monochrome videos. In order to operate in real-time, our tracking system is based on the matching abilities of binary descriptors and uses both the recursive optical flow and Mean-shift color matching. Figure 5.1 shows the initialization process through selecting an object frame the first frame in the video. The proposed system flowchart is shown in Figure 5.2.

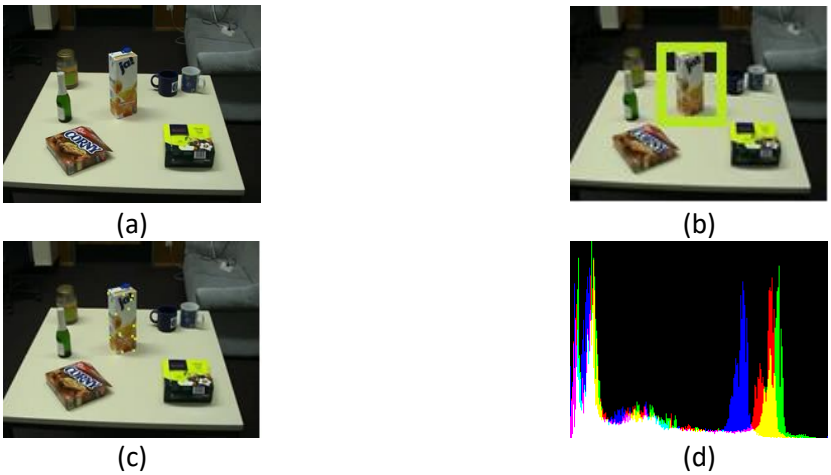

Figure 5.1: Object Identification. *(a) First frame, (b) selecting a specified object, (c) identifying object keypoints, (d) initial object histogram.*

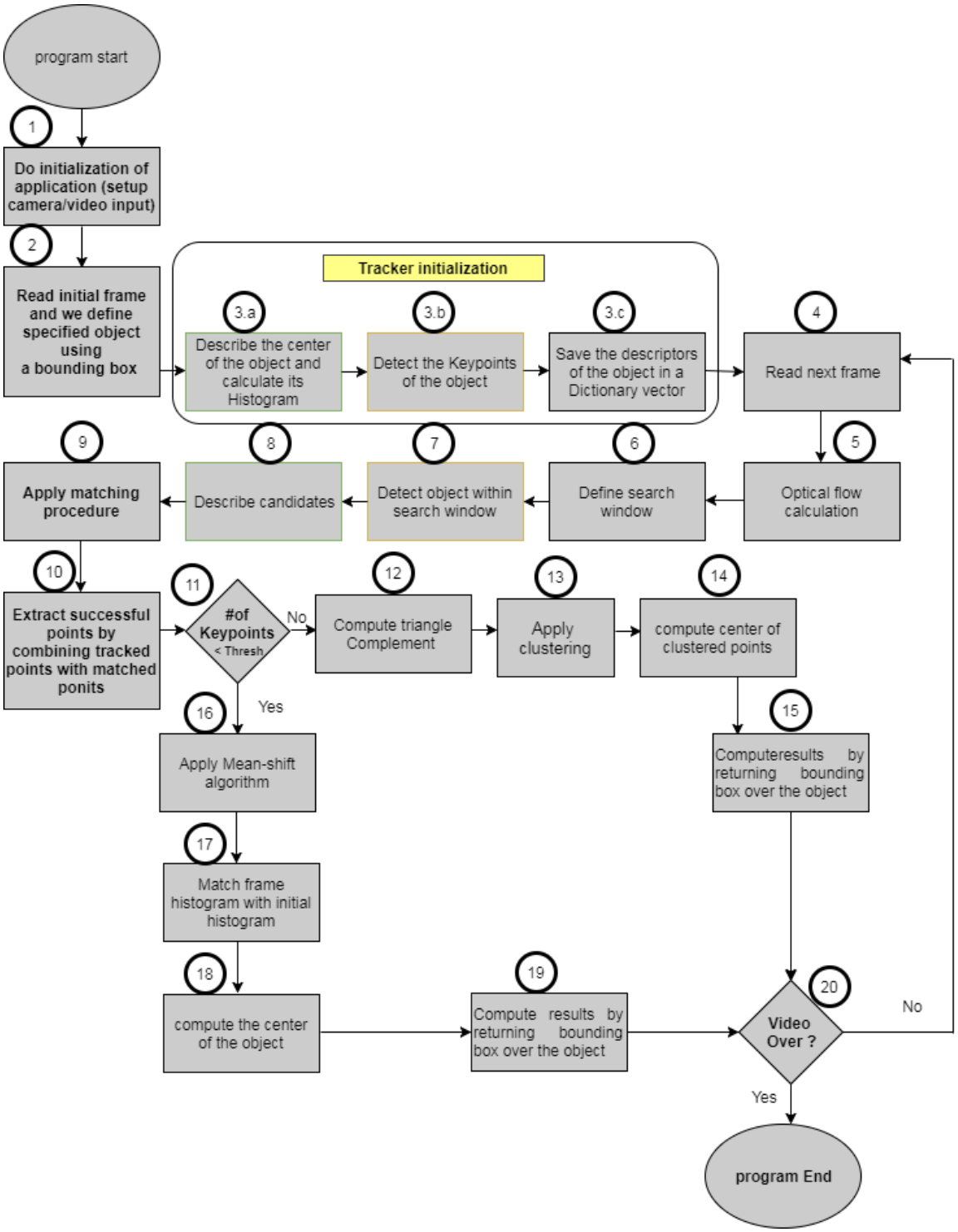

Figure 5.2: Proposed system flowchart.

5.2 Research Design Flowchart

This section describes the flowchart of our proposed object tracking algorithm as shown in Figure2. The steps in our object tracking process are:

STEP 1: The first step in our system is to initialize the video feeding process to apply the required object tracking. Our system can be initialized by live cameras or recorded videos.

STEP 2: Our tracking system can handle any real-time object. We read the initial frame from video and let the user to select the object to be tracked. The object is defined by a bounding box. As we use a specified datasets, we can use the ground truth coordinates to specify the object. Figure 5.3 shows the bounding box around object selected using the ground truth coordinates.

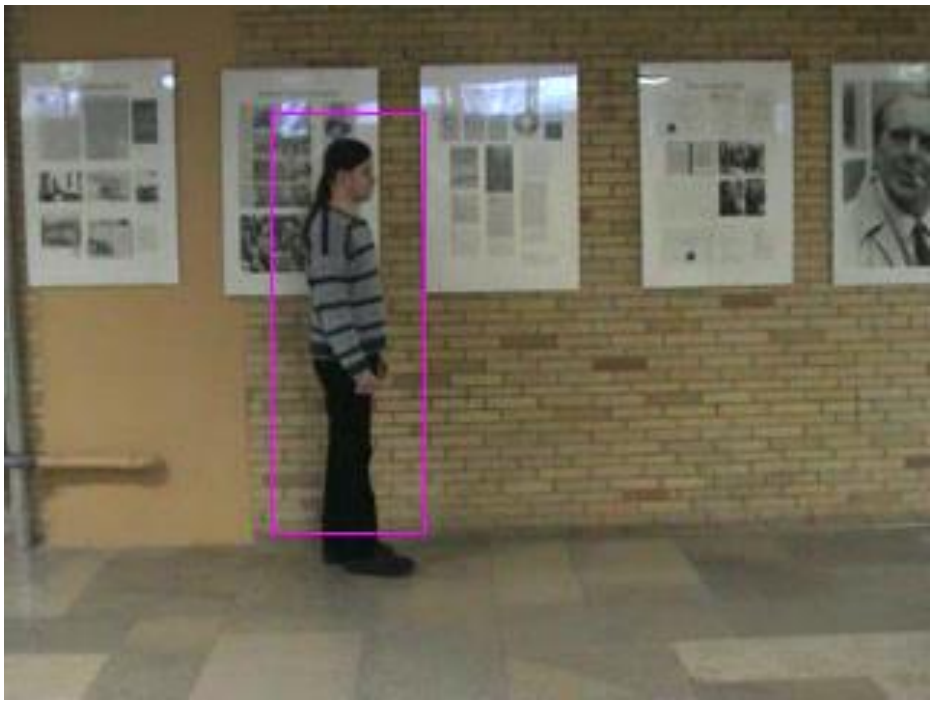

Figure 5.3: Bounding box around object using the ground truth coordinates

STEP 3.a: This step is the tracker initialization, where we initialize our tracker parameters by calculating the selected object center to be used as a reference in the next

frame. And we initialized the Mean-Shift algorithm [27] by taking the histogram of the selected object "*foreground histogram*" and taking the histogram of the background.

STEP 3.b

After specifying the object to be tracked, we apply the object detection algorithm (FAST) [18] as mentioned in chapter 3 to extract the object keypoints and translate these keypoints into foreground and background keypoints.

Our interest object model is represented by a set of keypoints according to equation (5.1):

$$O = \{L_i\}_{i=1}^{N} \tag{5.1}$$

where O is the object model, and L refers to the keypoint position in the image.

The keypoints inside the bounding box are labeled as foreground keypoints, and the keypoints located outside of the bounding box are labeled as background keypoints as shown in Figure 5.4. Algorithm 5.1 shows the detection algorithm using FAST for the first frame.

Figure 5.4: Image keypoints. Green dots represents the foreground keypoints, red dots represents the background keypoints.

Then we describe these keypoints using the binary descriptors FREAK [19] which is described in section 3.3.2.

Algorithm 5.1 : Object detetcion of the first frame

If (*initialized_properly***)**

load image *frame*

detect frame key points, and extract key points local features

store *frame* local features into *frame descriptor matrix (Dictionary descriptors)*

end

STEP 3.c: After using FREAK descriptor to describe the object keypoints, we save these descriptor keypoints to a vector called "Dictionary descriptors", which is responsible for holding the series of descriptors points for the object. An issue with setting up dictionary descriptors to hold descriptors keypoints at the start of a video stream is that most objects will not stay exactly the same throughout the duration of the tracking. The tracked object is likely to move forward or backward (change scale), rotate, succumb to occlusion, or undergo illumination changes. Therefore, we update our dictionary descriptors every N frame.

STEP 4: When the dictionary descriptor is ready, we read the next frame to start our tracking process.

STEP 5: In this step, we apply the first contributor in the frame processing. We apply recursive optical flow (Forward, backward optical flow) which takes two images: a past image fi-1 and the current frame fi as an input parameters, and stores the results into a vector matrix that holds the tracked points. "Tracked points" are the tracked point's resulted from forward optical flow and "Backward points" are the point's resulted from backward optical flow between two the frames. Afterward, we traverse the backward points so that we can remove the points on the fly by applying a threshold on the distance between the previous points and the backward points. The distance between the

backward points and the original points to be tracked which are previous points must be less than a threshold using the following equation:

$$previous\ points - \ backward\ points < \sigma \qquad (12.1)$$

Figure 5.5 demonstrates the recursive optical flow. Algorithm 5.2 defines the recursive optical flow algorithm.

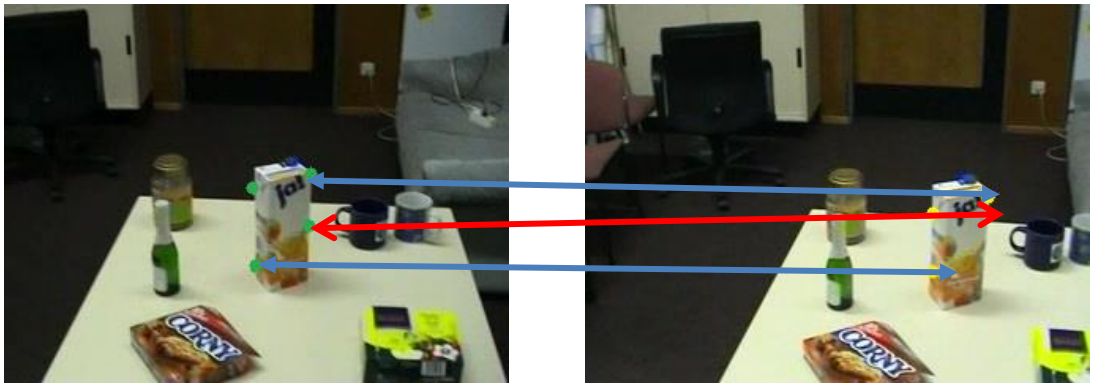

Figure 5. 5. Forward-backward optical flow method

Algorithm 5.2 : Recursive Optical flow

Input: Previous image (img_0), Current image (img_1), detected points (p_i)

Active points = detected points (p_i) ⟶ $p_1 \ldots\ldots\ldots p_n$

For *all* p_i *do*

Pyramidal lucas kanade(LK(p_i)) ⟶ *tracked points (p_i')*

Pyramidal lucas kanade(LK(p_i')) ⟶ *reversed points(p_i'')*

$Error = | p_i - p_i'' |$

If $Error > threshold$

Remove point

end

end

STEP 6: Afterward, we define a search window area. The selection process of this area is one of the most important key actions in efficiently implementing the detection mechanism of our tracking system. For instance, without using a large enough search area a fast moving object may not be tracked. On the other hand, increasing the search area size can have a dampen cost on the speed of the algorithm. Accordingly, without a

properly chosen search area, the tracker can lose the object by not searching sufficiently. A solution proposed for the searching methodology is that we first search the area depends on the user selection of the object borders, and based on equation (5.2) the search window area can be doubled. If we assume *r* is the rectangle which is defined by the user, the searching area **δ** is defined as:

$$\delta = \begin{cases} p_{active} + p_{tracked} < \dfrac{keypoints}{p_{active}}, & r * 2 \\ else\ where, & r \end{cases} \tag{5.2}$$

δ is new size of search window, *r* is default size of search window

STEP 7: when the searching window is ready, we apply the detection algorithm which is described in algorithm 5.3. We are interested in finding a set of keypoints $\mathbf{K_t}$ which are defined in equation (5.3). These keypoints represent the tracked object in the next frame

$$K_t = \{ L_i \}_{i=1}^{N} \tag{5.3}$$

Algorithm 5.3: Object Detection

```
while escape key do
    load first image frame
    detect frame key points, and extract key points local features
    store frame local features into frame descriptor matrix (Dictionary descriptors vector)
    if (image descriptor matrix) match (frame descriptor matrix) then
        store matched keypoints
    end
end
```

STEP 8: In this step, we describe the detected keypoints to apply the matching mechanism. The matching process for the described keypoints are done with their

descriptors which are stored in the dictionary. Depending on which descriptor the algorithm is used to extract the local features, a matching procedure needs to be chosen.

Binary visual descriptor FREAK [20] is used in our tracking system which meant for fast matching. FREAK descriptor computation requires fewer resources in terms of calculation power, and memory to store the resulting feature points [32].

STEP 9: The Hamming distance is calculated between two binary strings having the same length is the number of differing bits [7]. The matching between two FREAK obtained descriptor can be achieved single instruction, which is the sum of the XOR operation between the two binary strings.

$$h_d(f^1, f^2) = \sum_{i=1}^{d} XOR(f_i^1, f_i^2) \tag{5.4}$$

where h_d is the Hamming distance and f is the descriptor.

Brute force matching mechanism [7] is used to match the two descriptors f_i^1, f_i^2. After that we filter out the matched points to establish matches between keypoints k_i^0 from the initial frame exist in the dictionary descriptors, and candidates points k_j^t in the current frame.

We used a threshold on the distance between matches based on [33] where $\mathrm{D}(k_i^0, k_j^t) > \theta$, and we use a second threshold based on [34] where $\frac{\mathrm{D}(k_i^0, k_j^t)}{\mathrm{D}(k_i^0, k_m^t)} > \gamma$, the points that pass the two thresholds are considered to be the *successful matching points* as illustrated in algorithm 5.4 .

STEP 10: The resulted points of the optical flow "tracked points T" and the resulted points of the matching mechanism "successful matched points M " are combined together to produce a "combined points K' " set in size of $\mathbf{N^{K\prime}}$. Typically, K' still

contains outliers as there is some intrinsic ambiguity in the process of matching and tracking keypoints.

Algorithm 5.4: Matching mechanisim

Input : *keypoints_vector*, *keypoints_descriptor*

match(*keypoints_descriptor*, *dictionary_descriptor*)

for ∀ *keypoint* ∈ keypoints_*vector*

D1 = $D(k_i^0, k_j^t)$

D2 = $\frac{D(k_i^0, k_j^t)}{D(k_i^0, k_m^t)}$

If ($D1 > \theta$ && $D2 > \gamma$)

Push(*succsseful_points*)

end

end

STEP 11: In this step we determine if number of the extracted keypoints is enough to complete our tracking keypoints process or to go with image histogram matching path.

STEP 12: In order to locate the object of interest and remove the outliers, we let each two consecutive keypoints $(\mathbf{K_{L_1}}, \mathbf{K_{L_2}})$ in the "combined points vector" K' to poll out for a third point near the object center as shown in Figure 5.6.

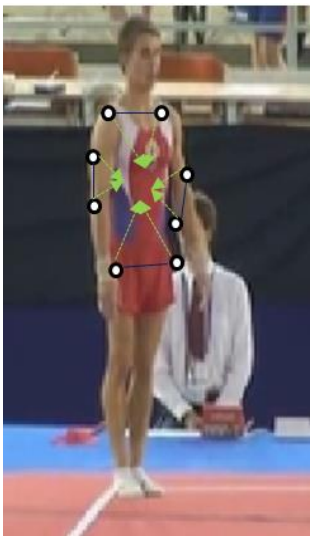

Figure 5.6: The third triangle complement point for each two consecutive keypoints

We find the third point coordinate $P_{L_3(X_3,Y_3)}$ as follow: assume we have two points $P_{L_1(X_1,Y_1)}, P_{L_2(X_2,\ Y_2)}$, we find the angle for P_{L_1}, P_{L_2} respectively using $atan2$ function according to equations (5.5), (5.6) respectively.

$$P_{L_1} Angle\, \theta_1 = atan_2(\frac{y_1}{x_1}) \tag{5.5}$$

$$P_{L_2} \text{ Angle } \theta_2 = atan_2(\frac{y_2}{x_2}) \tag{5.6}$$

After that we find the new coordinates of the third point x_3, y_3 using equations (5.7) and (5.8) respectively:

$$x_3 = \frac{\theta_2}{\theta_1 + \theta_2} \tag{5.7}$$

$$y_3 = \frac{\theta_1 \times \theta_2}{\theta_1 + \theta_2} \tag{5.8}$$

Figure 5.7 illustrates the way of finding the third point $P_{L_3(X_3,Y_3)}$ using two angles θ_1 and θ_2.

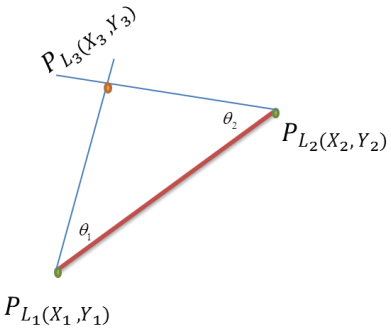

Figure 5.7: Calculating third point geometrically

We iterate these calculations for all keypoints in *K'* to condense the points toward the object center. We store all resulted points into a vector called "*center neighbors points J "*. Despite that the points in *J* are near to the center of the object of interest, it is not target the center.

STEP 13: Later we follow [9] in his unanimity by applying hierarchical agglomerative clustering [35] on "center neighbors points J ". We use Euclidean distance as a dissimilarity measure in the hierarchical agglomerative clustering. The agglomerative means start with all points in their own group $\{\mathbf{J^1}, \ldots\ldots\ldots, \mathbf{J^M}\}$, then we merge these groups until there is only one cluster that have the smallest dissimilarity$\mathbf{J^c}$. Data is then organized in hierarchical structure resulting in a dendrogram. In our thesis, we use single linkage in order to merge the groups together. The nearest-neighbor linkage or "single linkage" is the linkage where the dissimilarity $d_{\sin gle}$ between the two groups is the smallest dissimilarity between two points in the opposite groups. Equation (5.9) represents the single linkage where G is the first group, and H represents the second group.

$$d_{\sin gle}(G,H) = \min_{i\in G,\, j\in H}(d_{i,j}) \tag{5.9}$$

A cutoff threshold δ is used in order to form a flat cluster.

STEP 14: Then we compute the center $\boldsymbol{\mu}$ for the largest cluster using equation (5.10) where n is the cluster size.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} J_i^c \tag{5.10}$$

The center of the largest cluster represents the object center.

STEP 15: We use the calculated object center in the previous step to draw a bounding box around the specified object.

STEP 16: If the numbers of keypoints in K' are less than a threshold $\boldsymbol{\varepsilon}$ results from step 11, we use the Mean-Shift algorithm to search for the object.

STEP 17: In this step, we match the histogram of the current frame I_t with the initial frame I_0. The matching result will produce a bounding box rectangle around the object.

STEP 18: We compute the center of the resulted box to guide us to the center of the object that will be used to determine the searching window in next frame $\mathbf{I_{t+1}}$.

STEP 19: We use the calculated object center in the previous step to draw a bounding box around the specified object.

STEP 20: We loop the steps from STEP 4 – STEP 19 for all the frames related to a video. At the end of the last processing step, we test if numbers of frame are over or not. If yes the program will end.

Algorithm 5.6 illustrates the main loop of our proposed object tracker.

Algorithm 5.6: Main_loop

Input : $I_1 \ldots\ldots\ldots\ldots I_n$

define_object(mouse_curser , groundtruth)

detect(I_1) ⟶ Keypoints (K_t)

describe(K_t) ⟶ dictionary_descriptors

calc_Hist(I_1) ⟶ H_1

for t = 2…….n **do**

track (I_{t-1}, I_t, K_t) ⟶ R_t

detect (I_t) ⟶ K_t

combine(R_t, K_t) ⟶ K'

if ($K' > \theta$) **then**

triangle_complement (K') ⟶ J

cluster (J)

compute_center

return_bounding_box

else

meanshift(I_t) ⟶ H_t

match_HIst (H_t , H_1)

compute_center

return_bounding_box

end

end for

Chapter Six

Experiments and Evaluation Protocols

6.1 Introduction

In this chapter, we show an empirical evaluation for our proposed tracking algorithm. In order to establish a feasible estimation for our implementation, we need a framework in order to evaluate our proposed tracking algorithm. To evaluation our proposed algorithm, we used VOT2014 dataset [37], and Vojirtom dataset [38]. Moreover we applied standard evaluation metrics including accuracy, robustness, recall, and precision, to assess of our proposed tracking algorithm. Then, we compared our results with other state-of-art trackers. We employed accuracy and robustness of the results to compare with VOT2014 dataset, and we employ recall and precision metrics on the results to compare with Vojirtom dataset. A C++ implementation was developed, and all experiments were conducted on Intel® Core Intel i7-2600 4-cores processors 3.4 GHZ with 4.00 GB recall and precision. In the proposed algorithm implementation, we utilized many open source frameworks.

The flowchart of our proposed tracking algorithm in Figure 5.2 shows that we use binary descriptor to match the detected keypoints. Refer to [61], BRISK and FREAK are two descriptors which have the same properties of invariance to scale, rotation and global illumination. Thus, we applied the two descriptors in our tracker separately. The results of both descriptors are very similar.

6.2 Open Source Tools

There exist many open source frameworks which focus on image and video. The main benefit of using such a framework is that many of the basic functions are implemented and tested by the computer vision community. Some of frameworks includes OpenCV, SimpleCV, Torch3Vision, GPU-KLT, CImg [36].

OpenCV [7] is probably the most popular framework which is a free to use, and open source BSD licensed. FreeBSD means it is open and free to use with no restrictions. It is computer vision software library originally developed by Intel, and now maintained by Willow Garage. It has over two thousands algorithms written in C++ and can be installed on Windows, Linux, Mac OS X. Its interface is developed in the same language, but recently, new interfaces in Java, and MATLAB/OCTAVE have been developed.

Applications which are written using OpenCV can run on Microsoft OS, both PC, and Phone, and can run on Linux, Apple operating systems [28], Android [29], Maemo [30] and BlackBerry [31]. OpenCV was designed for computational efficiency and with a strong focus on real time applications. OpenCV deals with fields such security, medical image processing, human-computer interaction, augmented reality, and robotics.

OpenCV provides as mentioned various algorithms to achieve object detection. Although, using them without knowing how they work, and how to set the dedicated parameters, will produce unexpected results with extremely poor performance.

6.3 Performance Evaluation Methods

The proposed object tracking evaluations fulfill the requirements of the single target tracking, where the datasets are pre-frames, and the object location denoted as bounding box. Each video in the two datasets VOT2014 and Vojirtom has its own unique set of challenges ranging from different sizes, different rates of motion, and different resolutions. The effect of changing parameters will be examined on our proposed tracking algorithm where the results will be compared with many state-of-art trackers in order to provide a more telling of how our proposed tracking algorithm works.

6.3.1 Accuracy and Robustness Evaluation Methodology

In this evaluation methodology, we follow VOT2014 evaluation protocol, where two correlated easily interpretable measures are chosen: accuracy and robustness [39].

The accuracy Φ_t in equation (6.1) measures how well the bounding box A_t^T predicted by the proposed tracker overlaps with the ground truth bounding box A_t^G .

$$\Phi_t = \frac{A_t^G \cap A_T^T}{A_T^G \cup A_T^T} \tag{6.1}$$

The evaluation protocol in [39] repeats the tracker in the same dataset multiple times N_{rep} using equation (6.2) , where $\Phi(i,k)$ is the accuracy of the i-th tracker at frame t on experiments repetition k.

$$\Phi(i) = \frac{1}{N_{rep}} \sum_{k=1}^{N_{rep}} \Phi_t(i,k) \tag{6.2}$$

The robustness $F(i)$ is defined as the number of times the tracker failed, i.e., drifted from the target and it had to be reinitialized. The average robustness ρ_R of the i^{th} tracker is defined in equation (6.3).

$$\rho_R(i) = \frac{1}{N_{rep}} \sum_{k=1}^{N_{rep}} F(i,k) \tag{6.3}$$

According to the evaluation protocol [39], when the tracker reinitialized after it failed of overlapping with the ground truth, it skips 5 frames $N_{skip=5}$. Also, 10 frames were burned and not used in the accuracy calculations. Thus, the average accuracy ρ_A can be calculated using equation (6.4).

$$\rho_A = \frac{1}{N_{valid}} \sum_{k=1}^{N_{valid}} \Phi_k(i) \tag{6.4}$$

6.3.2 Precision and Recall Evaluation Methodology

Precision and recall are two basic measuring metrics which are used to evaluate searching strategies. Recall in equation (6.5) means how well the system returned most of the relevant results. It answered the question of how many relevant results are selected.

$$Recall = \frac{TP}{TP + FN} \tag{6.5}$$

While precision in equation (6.6) means how well the system returned substantially more relevant results than irrelevant. It answered the question of how many selected items are relevant.

$$Precesion = \frac{TP}{TP + FP} \tag{6.6}$$

Based on the overlap between algorithmic output and ground truth, each frame of a sequence is categorized as one of the four possible, where Faisal Bashir and Fatih Porikli mentioned in "*Performance Evaluation of Object Detection and Tracking Systems*" [40] these four possible cases, which they can be explained as shown in Figure 6.1.

True positive (TP): The number of frames where both ground truth and system results agree on the presence of one or more objects, and the bounding box of at least one or more objects coincides among ground truth and tracker results.

False negative (FN): The number of frames where ground of truth contains at least one object, while the system either does not contain any object or none of the system's objects fall within the bounding box of any ground truth object.

False positive (FP): The number of frames where system results contain at least one object, while ground truth either does not contain any object or none of the ground truth's objects falls within the bounding box of any system object.

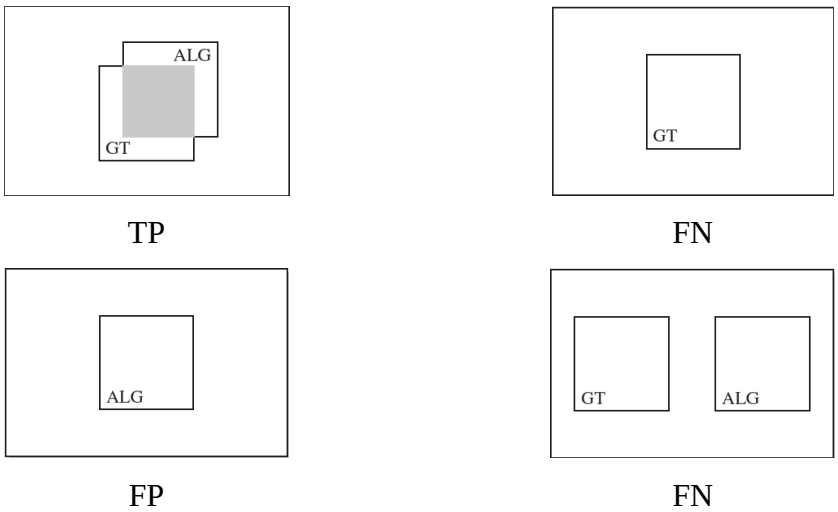

Figure 6.1: Four possible cases when comparing our bounding box to ground truth.

6.4 Datasets

In this section, we describe the datasets which are used to evaluate our proposed object tracking algorithm with other state-of-art object trackers, as well as the results of applying our proposed object tracking algorithm on these datasets. All of the datasets are combined with ground truth. VOT2014 [37] and Vojirtom [38] datasets are very popular datasets, which they are used for the evaluation process of our proposed tracking algorithm with current state-of art trackers.

6.4.1 VOT2014 Dataset

VOT2014 dataset [37] is used by many state-of-art trackers researchers to evaluate their performance in visual object tracking challenge in 2014, where the performance results of the state-of-art trackers are published in [39]. VOT2014 dataset consist of 25 video sequences where various objects in challenging occlusions, interference, and low lightning background attributes exist. The videos sequences are categorized in different level of difficulties as shwon in Table 6.1.

Table 6.1: VOT2014 sequences difficulty [39]

Sequence	Difficulty	Sequence	Difficulty	Sequence	Difficulty	Sequence	Difficulty
Motocross	**Hard**	fish1	**Interm**	trellis	**Interm / Easy**	bicycle	**Easy**
hand2		fernando		basketball		david	
diving		gymnastics		tunnel		ball	
fish2		torus		sunshade		sphere	
bolt		skating		jogging		car	
hand1				woman		drunk	
						surfing	
						polarbear	

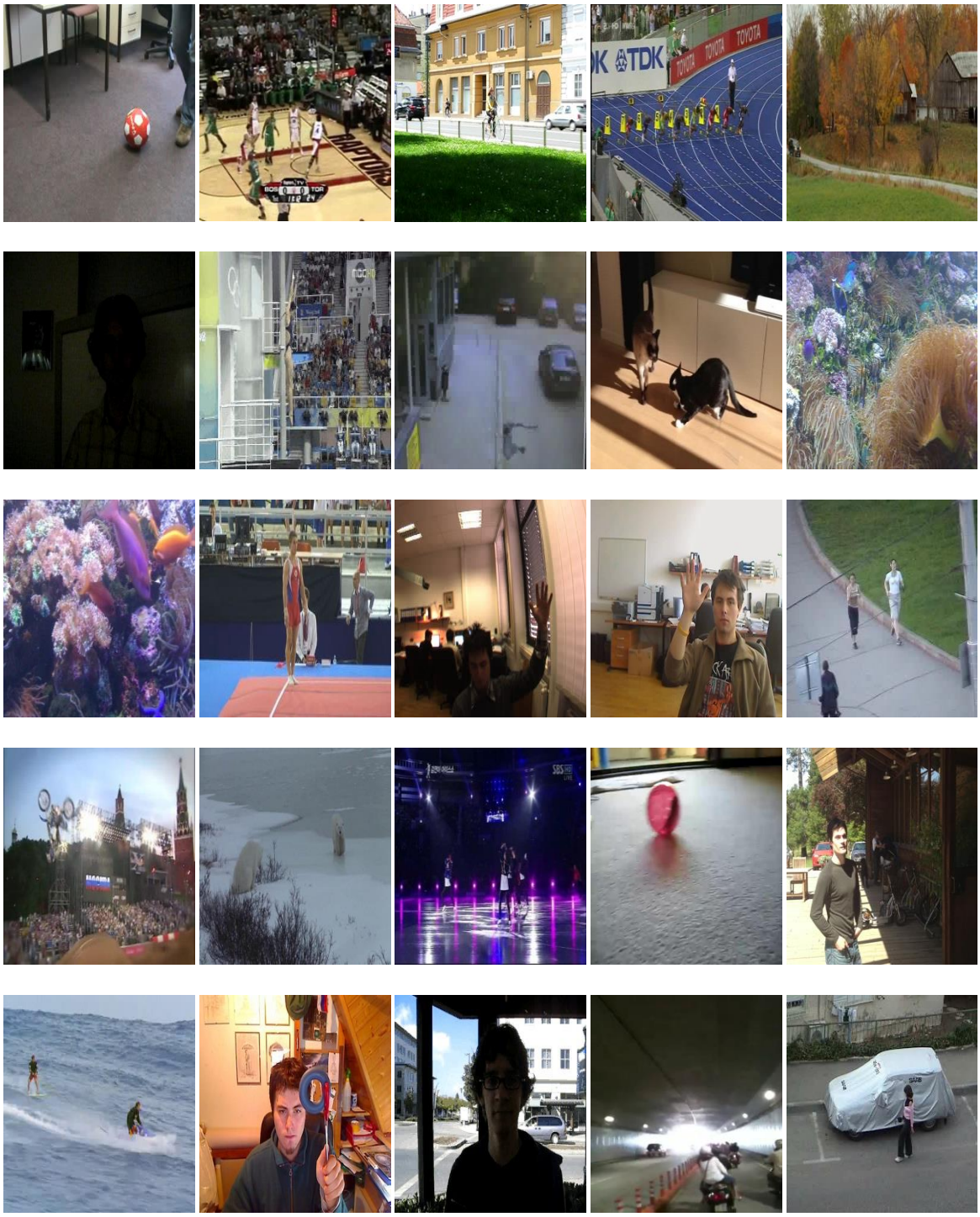

Figure 6.2: VOT2014 sequence used for embirically assesing. From left to right top to bottom: *ball, basketball, bicycle, bolt, car, david, diving, drunk, fernando,fish1, fish2, gymnastics, hand1, hand2, jogging, motorcross, polarbear, skating, sphere, sunsghade, surfing, torus, trellis, tunnel, woman.*[37]

VOT2014 dataset includes various visual phenomena with small number of sequences. Figure 6.2 shows the sequences which are exist in VOT2014 dataset. Each selected objects in each sequence are manually annotated by bounding box. Figure 6.3 shows a summary of number of frames in each video in the dataset.

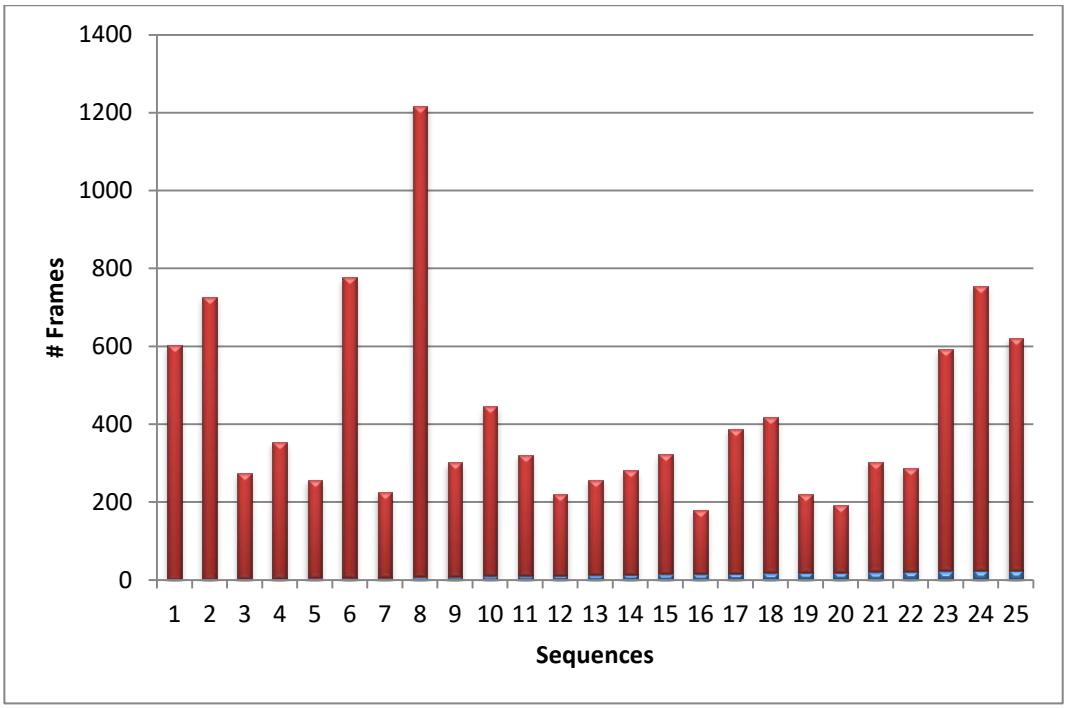

Figure 6.3: Summary for numbers of frames in each video in VOT2014 dataset

6.4.2 Vojirtom Dataset

Vojirtom dataset [38] provides a benchmark for testing and comparing different properties of visual tracking algorithm. Vojirtom dataset consist of 20 videos of sequences, where some videos are in gray-scale model. Table 6.2 shows which videos are in color mode and which of them are in gray-scale. Figure 6.4 shows the sequences which are exist in Vojirtom dataset. Various objects present in several challenging background where all frames have the same size of 320×240 pixels.

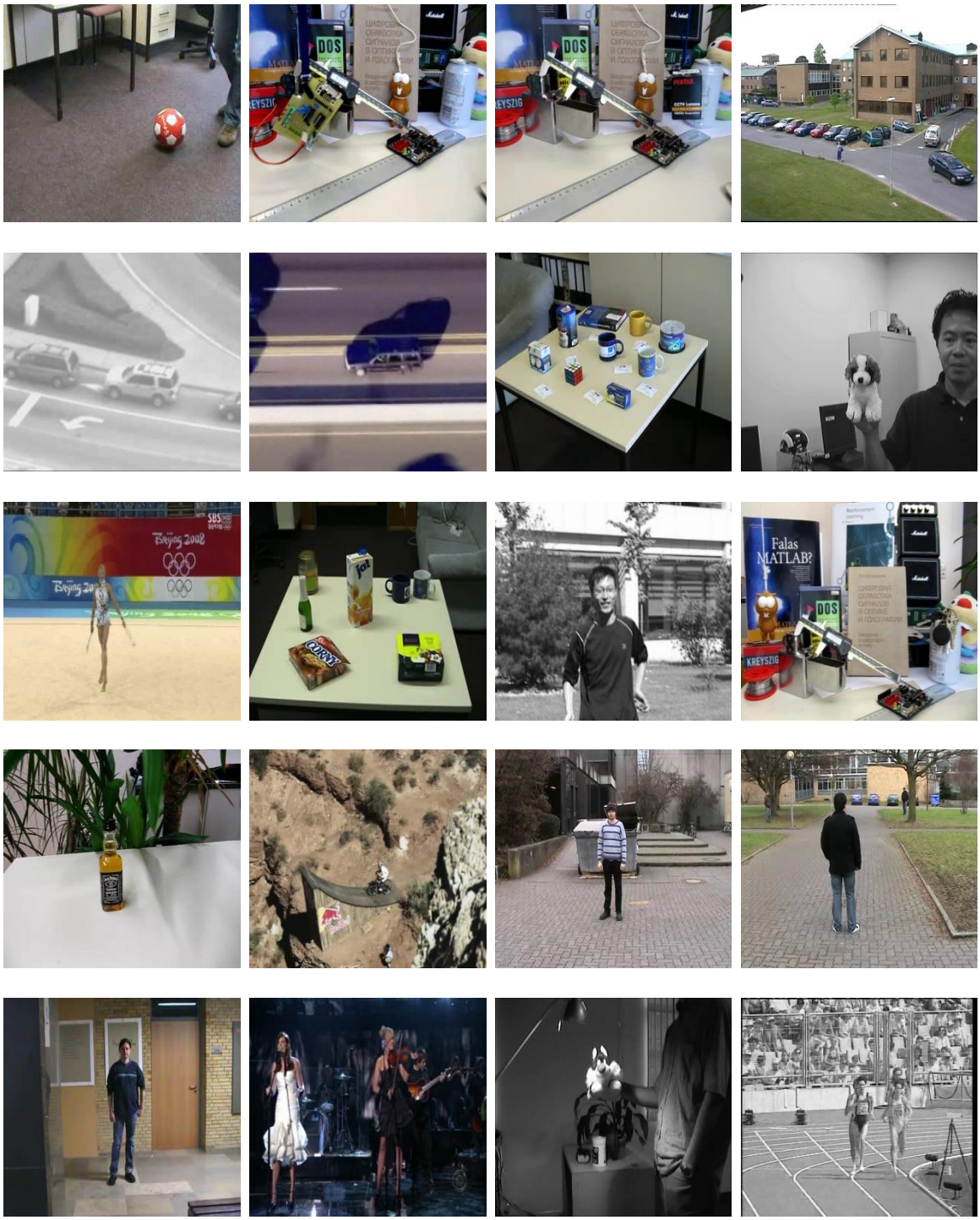

Figure 6.4: Vojirtom sequences: From left to right, top to bottom: *ball*, *board*, *box*, *car*, *car 2*, *carchase*, *cup on table*, *dog1*, *gym*, *juice*, *jumping*, *lemming*, *liquor*, *mountain-bike*, *person*, *person crossing*, *person partially occluded*, *singer*, *sylvester*, *track running* [38].

Table 6.2: Vojirtom Dataset color and gray-scale videos.

Sequence	Color	Sequence	Color
ball	YES	jumping	NO
Board		Sylvester	
Box		Track running	
car			
car 2			
carcahse			
cup on Table			
gym			
juice			
lemming			
liquor			
mountain-bike			
person			
Person Crossing			
person occlusion			
Singer			

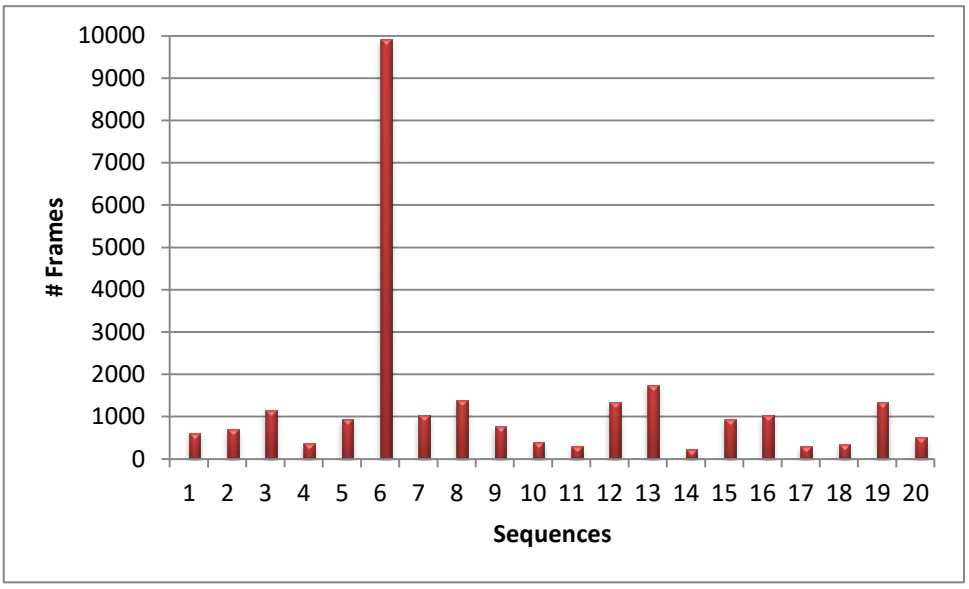

Figure 6.5: Summary for numbers of frames in each video in Vojirtom dataset

In Vojirtom dataset, Numbers of frames vary based on video type. Figure 6.5 shows a summary of number of frames in each video in the dataset.

6.5. Results of Using VOT2014 Sequences

In this section, the results of the sequences which are belonging to VOT2014 dataset are given. In all of the presented frames, a bounding box denotes the results. The performance metrics of accuracy and robustness according to section 6.3.1 are applied on our object tracking algorithm results. The experiments on VOT2014 are divided into two parts: the first part is a baseline sequence pooled, where our tracker runs on all sequences in VOT2014 dataset by initializing it on the ground truth bounding boxes. The second part is a baseline per-attribute, where our tracker runs on six sequence attributes, which are camera motion, illumination change, occlusion, seize change and motion change.

The results of our tracker and other state-of-art trackers in the two parts of experiments are summarized in Table 6.3; the first row of the Table which is highlighted by light green shows our tracker results. The top result at each experiment is highlighted by green. Refer to [39], a tracker which has a better performance is the tracker where the average value for accuracy and robustness is close to one. In the baseline sequence pooled experiment, DSST [41] was the best. In the baseline per-attribute experiment which focuses on the sensitivity of the trackers towards the scene attributes, our tracker shows a better performance than the other tracker. The last four columns in Table 6.3 denote tracker properties which are split into:

1. Localization.
2. Model type.
3. Visual model representation.
4. Scale adaptation.

Our algorithm is a deterministic localization, which is part-based because we used binary descriptor and a tracking method using optical flow and color matching. The visual model representation is generative because we used the keypoints stored in the "*Dictionary descriptors"* and we decide the tracking according to these descriptors, and we don't use scale adaptation. Figure 6.6 shows qualitative results on selected sequences.

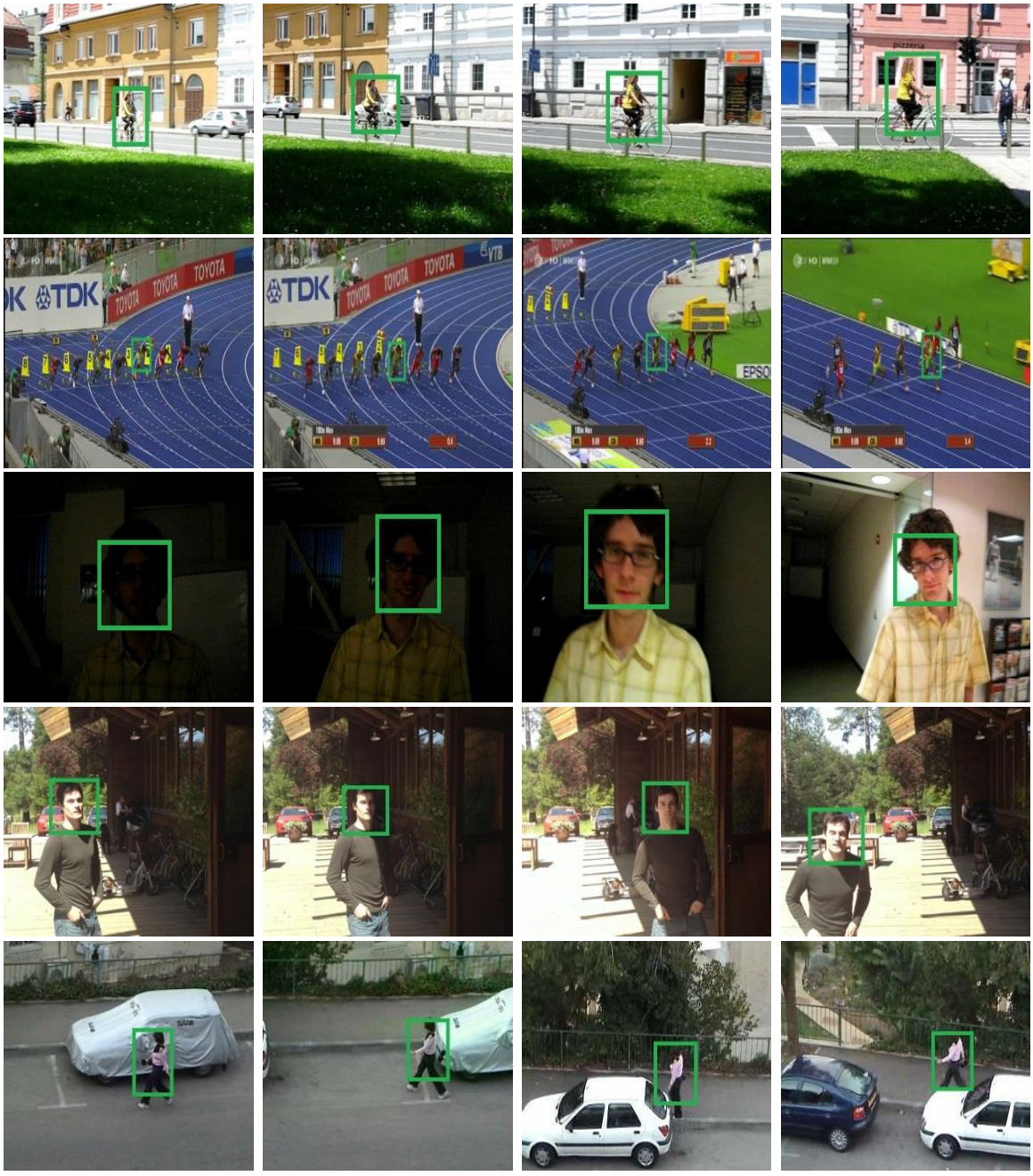

Figure 6.6: Qualitative results on bicycle, bolt, David, sunshade and woman

Table 6.3: Our results of using VOT2014 Dataset with other state-of-art trackers, the last four columns denote tracker properties which are split into: localization (stochastic/deterministic, i.e., S/D); model type (holistic/part-based, i.e., H/P); visual model representation (generative/discriminative, i.e., G/D); scale adaptation (yes/no, i.e., Y/N).

Experiment	Baseline			Baseline			Prosperities			
Normalization	Sequence pooled			Per-attribute						
Algorithm	Accuracy	Robustness	Avg.	Accuracy	Robustness	Avg.	Localization	Model type	Model representation	Scale
Ours	**7.45**	**4.4**	**5.9**	**5.86**	**6.72**	**6.29**	D	P	G	N
DSST [41]	3.67	9	6.33	5.41	12.08	**8.75**	D	H	D	Y
SAMF [42]	**3**	11.91	7.45	*5.3*	13.6	*9.45*	D	P	D	Y
KCF [43]	**3**	12.33	7.67	**5.05**	14.67	9.86	D	H	D	N
DGT [44]	4.62	5	**4.81**	10.76	9.13	9.95	D	P	G	Y
PLT14 [39]	12.29	2	7.15	13.88	*6.2*	10.04	D	H	D	Y
PLT13 [45]	17.5	**1**	9.25	17.54	**3.67**	10.6	D	H	D	N
eASMS [46]	10	6.8	8.4	13.48	13.35	13.41	D	H	G	Y
ACAT [73]	16	17.54	16.77	12.99	14.58	13.79	D	H	G	Y
HMM-TxD [76]	5	16.8	10.9	9.43	19.96	14.7	D	P	G	Y
MCT [47]	17.5	7.83	12.67	15.88	13.61	14.74	S	H	G	Y
MatFlow [39]	21.54	5	13.27	21.25	8.52	14.88	D	P	G	N
qwsEDFT [48]	17.5	16.92	17.21	16.65	18.5	17.58	D	H	G	N
ACT [77]	19.42	14.62	17.02	20.08	15.92	18	D	H	D	N
ABS [39]]	17.5	16.92	17.21	19.72	17.93	18.83	D	H	G	Y
VTDMG [73]	17.5	15.69	16.6	20.77	17.69	19.23	D	H	G	N
LGT [49]	28.63	5.75	17.19	28.12	11.28	19.7	S	P	G	Y
BDF [50]	23.5	15.69	19.6	22.42	17.1	19.76	D	P	G	N
aStruck [73]	22.5	20.45	21.48	21.41	18.43	19.92	D	P	D	N
DynMS [39]	18.54	15.69	17.12	21.54	18.8	20.17	S	H	G	Y
Struck [51]	19.58	24.6	22.09	20.11	20.3	20.21	D	H	D	N
Matrioska [10]	21.54	18.33	19.94	21.15	19.92	20.53	D	P	G	N
TStruck [51]	21.54	25.64	23.59	21.71	19.38	20.55	D	H	D	N
OGT [52]	12.06	29.78	20.92	13.76	29.13	21.44	S	H	G	N
EDFT [53]	18.54	24.43	21.49	19.43	23.71	21.57	D	H	G	N
CMT [9]	20.17	27.44	23.81	18.93	24.53	21.73	D	P	G	Y
SIR-PF [39]	23.5	18.5	21	23.62	20.13	21.88	S	H	G	N
FoT [54]	21	27.44	24.22	18.48	25.67	22.07	D	P	G	Y
LT-FLO [55]	17.5	30.5	24	15.98	29.85	22.91	S	P	G	Y
IPRT [73]	26.67	22.33	24.5	26.68	21.72	24.2	S	H	G	N
IIVTv2 [74]	29.35	30.67	30.01	24.79	24.81	24.8	D	P	G	Y
NCC [56]	17.5	38	27.75	17.74	34.25	26	D	H	G	N
PT+ [78]	32.64	15.69	24.16	32.05	20.68	26.36	D	P	G	Y
IMPNCC [39]	29.73	33.25	31.49	25.56	27.68	26.62	D	H	G	Y
FRT [57]	21	35	28	23.38	30.39	26.89	D	P	G	N
FSDT [73]	31.5	33.4	32.45	23.55	31.16	27.36	D	H	D	Y
IVT [58]	28.05	33.14	30.6	27.23	28.9	28.06	D	H	G	Y
MIL [59]	34.25	28.38	31.31	33.95	24.2	29.08	D	H	D	N
CT [60]	32.64	33.14	32.89	31.51	27.79	29.65	D	H	D	N

In term of accuracy, the top performing trackers in baseline sequence-pooled experiment are KCF, SAMF, DSST and DGT. In term of robustness the top performing trackers in the same experiment are PLT_{13}, PLT_{14}, and our tracker. Averaging the accuracy and robustness yields to show the DGT tracker was the best among other trackers.

In the baseline per-attribute experiment, in the term of accuracy, the top performing trackers are KCF, SAMF, DSST and our tracker. In term of robustness the top performing trackers in the same experiment are PLT_{13}, PLT_{14} and our tracker. Averaging the accuracy and robustness yields to show our tracker was the best among other trackers.

The DSST [41] is an extension to MOSSE [75] which uses gray-scale in addition to HOG. SAMF [42] and KCF [43] address the scale change, SAMF uses HOG features and color naming which are integrated together to boost the overall tracking performance, while KCF uses multi-scale support, sub-cell peak estimation and replacing the model update by linear interpolation.

PLT_{13} [45] and PLT_{14} [39] are extensions to STRUCK [51] tracker which apply histogram back projection as feature selection in SVM training. PLT_{13} runs a classifier at a fixed single scale for each test image which is the Winner in VOT2013 challenge. But, it doesn't adapt the target size, while PLT_{14} in an extension to PLT_{13} adapts the target size.

MCT tracker [47] is discriminative online learning classifier based on Adaboost, while the adaptive color tracker ACT [77] uses temporally scheme for updating the tracking model instead training the classifier separately on samples.

6.6 Results of Using Vojirtom Sequences

In this section, the results of the sequences which are belonging to Vojirtom dataset are given. The performance metrics of Recall and Precision according to section 6.3.2 are applied on the results of our proposed object tracking algorithm. To assess the performance of our tracker, we used the results of Recall values from [9] which is also applied to Vojirtom dataset. The author in [9] applied the overlap between algorithmic output bounding box b_T and ground truth bounding box b_{GT}. $\phi(b_T, b_{GT}) = \frac{b_T \cap b_{GT}}{b_T \cup b_{GT}}$

The overlap $\phi(b_T, b_{GT})$ between algorithmic output and the ground truth must be $> \omega$. In our assessment we took the results where $\omega = 0.25$.

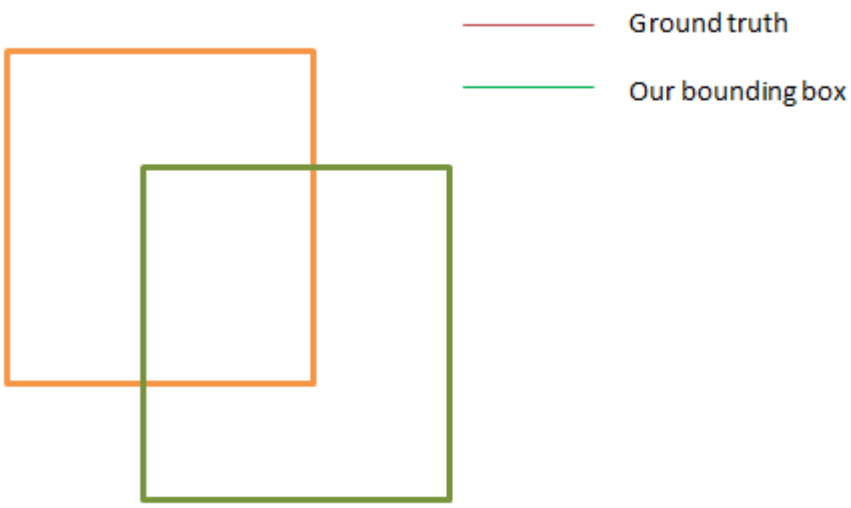

Figure 6.7: Overlap between the ground truth and our bounding box

The author in [16] compare his results quantitatively to the state-of-arts tracking approaches STRUCK (Structured output Tracking) [51], TLD (Tracking-Learning-Detection) [62], LM (Learn Match) [65], FT (Fragments-based Tracking) [63], HT (Hough Track) [64] and SB (Semi-supervised online Boosting) [66]. We summarized

our results compared with other state-of-arts trackers in Table 6.4. Figure 6.8 shows qualitative results on selected sequences.

Table 6.4: Our Recall values results compared to the state-of-the arts tracker in [9]

#	Sequence		Algorithms						
		Ours	CMT [9]	STRUCK [51]	TLD [62]	FT [63]	HT [64]	LM [65]	SB [66]
1.	**ball**	**0.87**	0.98	0.3	0.4	0.31	0.15	0.14	0.3
2.	**Board**	**0.87**	0.84	0.91	0.45	0.82	0.26	0.33	0.15
3.	**Box**	**0.76**	0.94	0.99	0.39	0.07	0.14	0.63	0.37
4.	**car**	**0.84**	0.59	0.98	0.52	0.33	0.57	0.14	0.12
5.	**car 2**	**0.93**	0.9	0.81	1	0.04	0.59	0.46	0.72
6.	**carcahse**	**0.51**	0.3	0.08	0.16	0.04	0.04	0	0.08
7.	**cup on Table**	**1**	0.83	1	0.89	1	1	0.68	0.47
8.	**Dog1**	**1**	1	0.86	0.9	0.84	0.83	0.77	0.51
9.	**gym**	**0.85**	0.93	1	0.76	0.24	0.3	0.1	0.61
10.	**juice**	**1**	1	1	1	0.09	1	1	0.43
11.	**jumping**	**0.93**	0.9	1	0.88	0.39	0.99	0.14	0.07
12.	**lemming**	**0.62**	0.65	0.72	0.15	0.16	0.29	0.02	0.17
13.	**liquor**	**0.91**	0.91	0.7	0.46	0.86	0.43	0.07	0.43
14.	**mountain-bike**	**1**	0.99	0.99	0.37	0.65	0.99	0.11	0.2
15.	**person**	**0.82**	0.95	1	0.92	1	0.49	0.75	0.52
16.	**Person Crossing**	**0.73**	0.76	0.51	0.86	0.88	0.18	0.8	0.96
17.	**person occlusion**	**1**	1	1	1	1	1	1	0.99
18.	**singer**	**1**	1	0.48	0.77	0.52	0.24	0.21	0.15
19.	**Sylvester**	**1**	0.99	0.99	0.97	0.86	1	0.62	0.41
20.	**Track running**	**1**	1	1	0.01	0.94	0.2	0.01	0.31
	Average	**0.882**	0.873	0.816	0.643	0.552	0.5345	0.399	0.396842

Table 6.4 presents the obtained recall values for our algorithm compared to other stat-of-arts trackers empirically in [9], our algorithm achieves highest recall values in 8 videos, and 6 recall values equally to CMT algorithm which it had the best results according to [9] out 20 sequences. Our algorithm attains the highest average recall value in all categories which is highlighted by green.

We also computed Precision values of our track for the sequences in Vojirtom dataset, which is concluded in Table 6.5.

Table 6.5: Precision values.

Sequence	**Precision value**
ball	1
Board	0.8
Box	0.78
car	1
car 2	0.95
carcahse	0.48
cup on Table	1
Dog1	0.97
gym	0.9
juice	1
jumping	0.9
lemming	0.6
liquor	0.9
mountain-bike	1
person	0.95
Person Crossing	1
person occlusion	1
singer	1
Sylvester	1
Track running	1
Average	0.9115

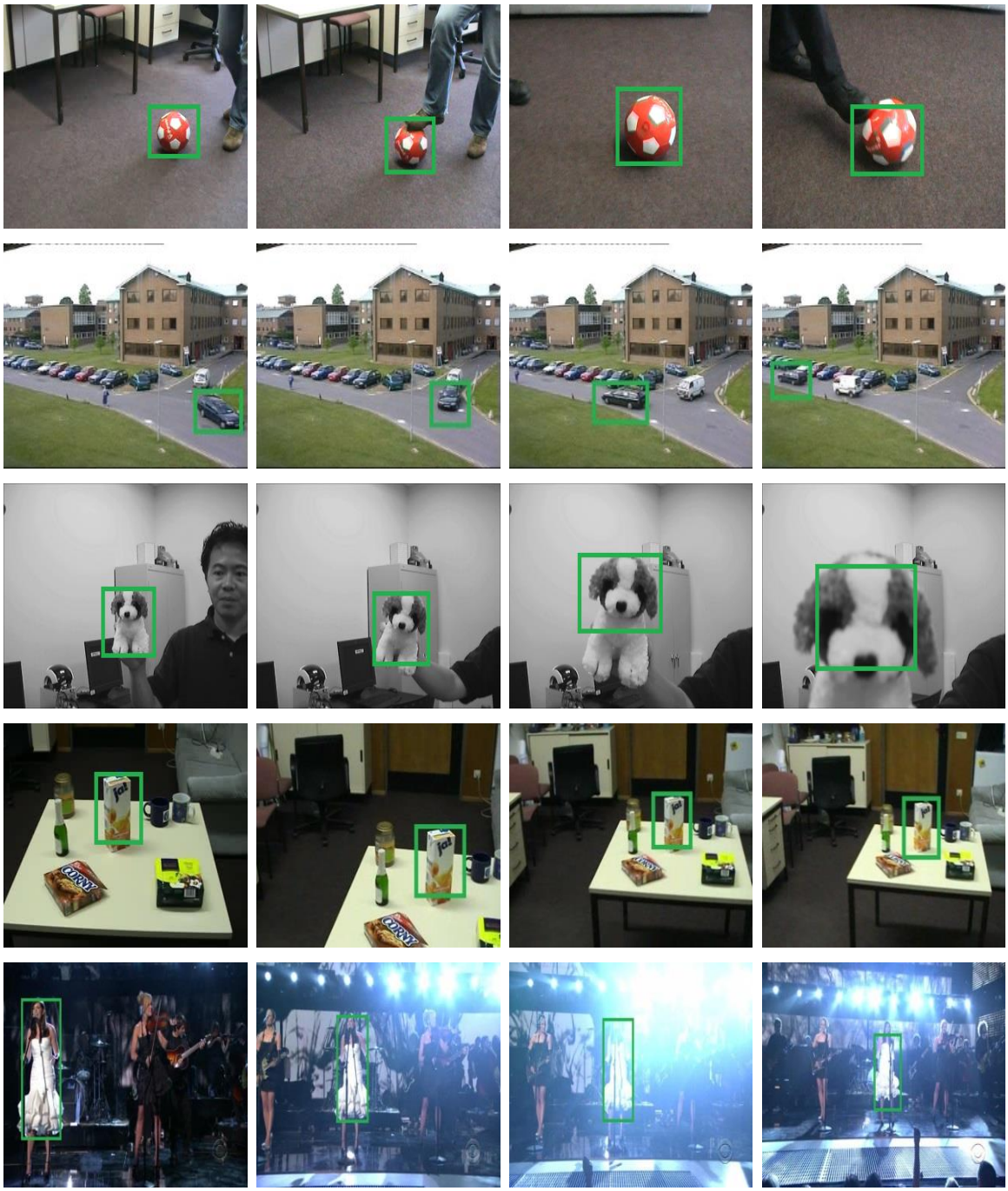

Figure 6.8: Qualitative results on *ball, car, dog1, juice and singer*

Chapter Seven

Conclusions and Future Work

7.1 Conclusions

In this thesis, we introduced a novel object tracking method to track an online single targeted object. Our work employs binary descriptors with pyramidal optical flow in addition to mean-shift color matching to track a specified object within subsequent frames in a robust manner. We use the location of every neighboring points to estimate a third point to be considered as the objects center. Clustering of correspondences approach is used to group the possible centers and to find their center to be considered as the final estimated object center.

The experimental evaluation demonstrated that our proposed object tracking method is able to achieve better overall results than state-of-art algorithms on large number of sequences. The output results demonstrated clearly that our proposed method is successful on diverse datasets when applied on both VOT2014 and Vojirtom datasets.

The results shows that our algorithm achieves 5.9 in accuracy and robustness average values in sequence pooled experiment, and 6.29 in accuracy and robustness average values in per-attribute experiment of VOT2014 dataset. Our algorithm achieves the highest recall value of 0.882 among the stat-of-the art trackers and 0.9115 precision value in Vojirtom dataset.

7.2 Future Work

One possible interesting future research direction is to implement our tracker using FPGA or utilizing GPU powers to reduce the required processing time. Currently, our tracker gives the location of the object of interest but not its orientation, in future research we can aim to modify our tracker give better information about the orientation of the object.

References

[1] Goldstein, E. B. (2009, Feb). Sensation and Perception. Wadsworth Publishing, 8 edition.

[2] Shapiro, L. G. & Stockman, G. C.(2001, Jan). Computer Vision. Prentice Hall.

[3] Szeliski, R. (2010). Computer vision: algorithms and applications. Springer Science & Business Media.

[4] Maggio, E., & Cavallaro, A. (2011). Video tracking: theory and practice. John Wiley & Sons.

[5] A. Yilmaz, O. Javed, and M. Shah. Object tracking*: A survey*. *ACM Computing Surveys*, 38(4), Dec. 2006.

[6] Chen, Y., & Park, P. S. (2013). Object Tracking Based on Online Classification Boosted by Discriminative Features. *International Journal of Energy, Information and Communications*, *4*(6).

[7] Kaehler, A., & Bradski, G. (2014). *Learning OpenCV*. O'Reilly Media, Inc..

[8] Chen, X., Li, X., Wu, H., & Qiu, T. (2012, March). Real-time object tracking via CamShift-based robust framework. In *Information Science and Technology (ICIST), 2012 International Conference on* (pp. 527-530). IEEE.

[9] Nebehay, G., & Pflugfelder, R. (2015). Clustering of Static-Adaptive Correspondences for Deformable Object Tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2784-2791).

[10] Maresca, M. E., & Petrosino, A. (2013, September). Matrioska: A multi-level approach to fast tracking by learning. In *International Conference on Image Analysis and Processing* (pp. 419-428). Springer Berlin Heidelberg.

[11] Maresca, M. E., & Petrosino, A. (2014, June). The Matrioska tracking algorithm on LTDT2014 dataset. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 720-725). IEEE.

[12] T. Tuytelaars and K. Mikolajczyk, Local invariant feature detectors: a survey," Foundations and Trends R in Computer Graphics and Vision, vol. 3, no. 3, pp. 177{280, 2008.

[13] Grauman, K., & Leibe, B. (2011). Visual object recognition. *Synthesis lectures on artificial intelligence and machine learning*, *5*(2), 1-181.

[14] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

[15] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, *110*(3), 346-359.

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

[17] Harris, C., & Stephens, M. (1988, August). A combined corner and edge detector. In *Alvey vision conference* (Vol. 15, p. 50).

[18] Rosten, E., & Drummond, T. (2006, May). Machine learning for high-speed corner detection. In *European conference on computer vision* (pp. 430-443). Springer Berlin Heidelberg.

[19] Leutenegger, S., Chli, M., & Siegwart, R. Y. (2011, November). BRISK: Binary robust invariant scalable keypoints. In *2011 International conference on computer vision* (pp. 2548-2555). IEEE.

[20] Alahi, A., Ortiz, R., & Vandergheynst, P. (2012, June). Freak: Fast retina keypoint. In *Computer vision and pattern recognition (CVPR), 2012 IEEE conference on* (pp. 510-517). Ieee.

[21] Gleason, J. (2011). Brisk (presented by josh gleason). In *International Conference on Computer Vision*.

[22] Gould, S., Arfvidsson, J., Kaehler, A., Sapp, B., Messner, M., Bradski, G. R., ... & Ng, A. Y. (2007, January). Peripheral-Foveal Vision for Real-time Object Recognition and Tracking in Video. In *IJCAI* (Vol. 7, pp. 2115-2121).

[23] Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. " O'Reilly Media, Inc.".

[24] Bruhn, A., Weickert, J., & Schnörr, C. (2005). Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods. International Journal of Computer Vision, 61(3), 211-231.

[25] Bouguet, J. Y. (2001). Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Intel Corporation, 5(1-10), 4.

[26] Kalal, Z., Mikolajczyk, K., & Matas, J. (2010, August). Forward-backward error: Automatic detection of tracking failures. In *Pattern recognition (ICPR), 2010 20th international conference on* (pp. 2756-2759). IEEE.

[27] Comaniciu, D., & Ramesh, V. (2000). Mean shift and optimal prediction for efficient object tracking. In *Image processing, 2000. proceedings. 2000 international conference on* (Vol. 3, pp. 70-73). IEEE.

[28] IOS port for opencv. [Online]. Available: http://www.eosgarden.com/en/opensource/opencv-ios/overview/. [Accessed 23 March 2017]

[29] Android port for opencv. [Online]. Available: http://opencv.willowgarage.com/wiki/AndroidExperimental [Accessed 23 March 2017]

[30] Maemo port for opencv. [Online]. Available: https://garage.maemo.org/projects/opencv [Accessed 23 March 2017]

[31] Blackberry port for opencv. [Online]. Available: https://github.com/blackberry/OpenCV [Accessed 23 March 2017]

[32] D. G. Lowe and M. Muja, Fast matching of binary features," in Computer and Robot Vision (CRV), 2012 Ninth Conference on. IEEE, 2012, pp. 404{410.

[33] Heinly, J., Dunn, E., & Frahm, J. M. (2012). Comparative evaluation of binary features. In *Computer Vision–ECCV 2012* (pp. 759-773). Springer Berlin Heidelberg.

[34] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

[35] Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, *16*(3), 645-678.

[36] Computer vision Algorithms Implementations http://www.cvpapers.com/rr.html [Accessed 15 March 2017]

[37] VOT2014 dataset http://www.votchallenge.net/vot2014/dataset.html [Accessed 15 March 2017]

[38] Vojirtom dataset http://cmp.felk.cvut.cz/~vojirtom/dataset/ [Accessed 15 March 2017]

[39] Kristan, M., Matas, J., Leonardis, A., Vojíř, T., Pflugfelder, R., Fernandez, G., ... & Čehovin, L. (2016). A novel performance evaluation methodology for single-target trackers. *IEEE transactions on pattern analysis and machine intelligence*, *38*(11), 2137-2155.

[40] Bashir, F., & Porikli, F. (2006, June). Performance evaluation of object detection and tracking systems. In *Proceedings 9th IEEE International Workshop on PETS* (pp. 7-14).

[41] Danelljan, M., Häger, G., Khan, F., & Felsberg, M. (2014). Accurate scale estimation for robust visual tracking. In *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press.

[42] Li, Y., & Zhu, J. (2014, September). A scale adaptive kernel correlation filter tracker with feature integration. In *European Conference on Computer Vision*(pp. 254-265). Springer International Publishing.

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

[44] Cai, Z., Wen, L., Lei, Z., Vasconcelos, N., & Li, S. Z. (2014). Robust deformable and occluded object tracking with dynamic graph. *IEEE Transactions on Image Processing*, *23*(12), 5497-5509.

[45] Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., Porikli, F., Cehovin, L., ... & Khajenezhad, A. (2013). The visual object tracking vot2013 challenge results. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (pp. 98-111).

[46] Vojir, T., Noskova, J., & Matas, J. (2013, June). Robust scale-adaptive mean-shift for tracking. In *Scandinavian Conference on Image Analysis* (pp. 652-663). Springer Berlin Heidelberg.

[47] Duffner, S., & Garcia, C. (2014, September). Exploiting contextual motion cues for visual object tracking. In *European Conference on Computer Vision* (pp. 232-243). Springer International Publishing.

[48] Öfjäll, K., & Felsberg, M. (2014, September). Weighted update and comparison for channel-based distribution field tracking. In *European Conference on Computer Vision* (pp. 218-231). Springer International Publishing.

[49] Cehovin, L., Kristan, M., & Leonardis, A. (2013). Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(4), 941-953.

[50] Maresca, M. E., & Petrosino, A. (2014, September). Clustering local motion estimates for robust and efficient object tracking. In *European Conference on Computer Vision* (pp. 244-253). Springer International Publishing.

[51] Hare, S., Golodetz, S., Saffari, A., Vineet, V., Cheng, M. M., Hicks, S. L., & Torr, P. H. (2016). Struck: Structured output tracking with kernels. *IEEE transactions on pattern analysis and machine intelligence*, *38*(10), 2096-2109.

[52] Nam, H., Hong, S., & Han, B. (2014, September). Online graph-based tracking. In *European Conference on Computer Vision* (pp. 112-126). Springer International Publishing.

[53] Felsberg, M. (2013). Enhanced distribution field tracking using channel representations. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (pp. 121-128).

[54] Wendel, A., Sternig, S., & Godec, M. (2011, February). Robustifying the flock of trackers. In *16th Computer Vision Winter Workshop. Citeseer* (p. 91).

[55] Lebeda, K., Hadfield, S., Matas, J., & Bowden, R. (2013). Long-term tracking through failure cases. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (pp. 153-160).

[56] Briechle, K., & Hanebeck, U. D. (2001, March). Template matching using fast normalized cross correlation. In *Aerospace/Defense Sensing, Simulation, and Controls* (pp. 95-102). International Society for Optics and Photonics.

[57] Adam, A., Rivlin, E., & Shimshoni, I. (2006, June). Robust fragments-based tracking using the integral histogram. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on* (Vol. 1, pp. 798-805). IEEE.

[58] Ross, D. A., Lim, J., Lin, R. S., & Yang, M. H. (2008). Incremental learning for robust visual tracking. *International Journal of Computer Vision*, *77*(1-3), 125-141.

[59] Babenko, B., Yang, M. H., & Belongie, S. (2011). Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, *33*(8), 1619-1632.

[60] Zhang, K., Zhang, L., & Yang, M. H. (2012, October). Real-time compressive tracking. In *European Conference on Computer Vision* (pp. 864-877). Springer Berlin Heidelberg.

[61] Spang, V., & Henry, A. (2014). Object Tracking Using Local Binary Descriptors.

[62] Kalal, Z., Mikolajczyk, K., & Matas, J. (2012). Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, *34*(7), 1409-1422.

[63] Adam, A., Rivlin, E., & Shimshoni, I. (2006, June). Robust fragments-based tracking using the integral histogram. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on* (Vol. 1, pp. 798-805). IEEE.

[64] Godec, M., Roth, P. M., & Bischof, H. (2013). Hough-based tracking of non-rigid objects. *Computer Vision and Image Understanding*, *117*(10), 1245-1256.

[65] Hare, S., Saffari, A., & Torr, P. H. (2012, June). Efficient online structured output learning for keypoint-based object tracking. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (pp. 1894-1901). IEEE.

[66] Grabner, H., Leistner, C., & Bischof, H. (2008, October). Semi-supervised on-line boosting for robust tracking. In *European conference on computer vision*(pp. 234-247). Springer Berlin Heidelberg.

[67] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. (2011, November). ORB: An efficient alternative to SIFT or SURF. In Computer Vision (ICCV), 2011 IEEE International Conference on (pp. 2564-2571). IEEE.ISO 690

[68] WANG, J. H., FENG, F., & LIANG, W. (2012). Moving Object Tracking Based on Mean-Shift. *Science Technology and Engineering*, *3*, 022.

[69] Veenman, C. J., Reinders, M. J., & Backer, E. (2001). Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(1), 54-72.

[70] Broida, T. J., & Chellappa, R. (1986). Estimation of object motion parameters from noisy images. *IEEE transactions on pattern analysis and machine intelligence*, (1), 90-99.

[71] Tsagkatakis, G., & Savakis, A. (2011). Online distance metric learning for object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, *21*(12), 1810-1821..

[72] Calonder, M., Lepetit, V., Strecha, C., & Fua, P. (2010). Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, 778-792.

[73] LIRIS, F. The Visual Object Tracking VOT2014 challenge results.

[74] Moo Yi, K., Jeong, H., Heo, B., Jin Chang, H., & Young Choi, J. (2013). Initialization-insensitive visual tracking through voting with salient local features. *In Proceedings of the IEEE International Conference on Computer Vision (pp. 2912-2919).*

[75] Bolme, D. S., Beveridge, J. R., Draper, B. A., & Lui, Y. M. (2010, June). Visual object tracking using adaptive correlation filters. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (pp. 2544-2550). IEEE.

[76] Vojir, T., Matas, J., & Noskova, J. (2016). Online adaptive hidden markov model for multi-tracker fusion. Computer Vision and Image Understanding, 153, 109-119. ISO 690

[77] Danelljan, M., Shahbaz Khan, F., Felsberg, M., & Van de Weijer, J. (2014). Adaptive color attributes for real-time visual tracking. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1090-1097). ISO 690

[78] Duffner, S., & Garcia, C. (2013). Pixeltrack: a fast adaptive algorithm for tracking non-rigid objects. In Proceedings of the IEEE international conference on computer vision (pp. 2480-2487).

[79] Davidson, W., & Abramowitz, M. (2006). Molecular expressions microscopy primer: Digital image processing-difference of gaussians edge enhancement algorithm. Olympus America Inc., and Florida State University.

تتبع كائن في الفيديو باستخدام ادوات مفتوحة المصدر

إعداد	المشرف	مشرف مشارك
اسامة خالد راشد نوافلة	الدكتور محمد الجراح	الدكتور عبد الكريم التميمي

الملخص

يعتبر التتبع لكائن مرئي من احدى المجالات البحثية المهمة في مجال رؤيا الحاسوب (Computer Vision). تستند معظم طرق التتبع الشائعة على الكشف عن الكائن المراد تتبعه (object detection) ثم تطبيق طريقة مناسبة للتتبع (object tracking). هذا النوع من التتبع للكائن يسمح بتتبع الكائن المحدد (specified object) من خلال تحديده في الصورة الاولى. يهدف العمل في هذه الأطروحة على ايجاد حل جديد لتتبع كائن واحد محدد مع عدم وجود معرفة أو بيانات خاصة مسبقة لهذا الكائن. في هذه الأطروحة تم استخدام الواصف الثنائي (FREAK descriptor) لوصف ملامح الكائن المحدد لتعتبر كنقاط مفتاحية لهذا الكائن و التي تم الكشف عنها باستخدام (FAST detector). بعد ذلك، تم استخدام التدفق البصري الهرمي (pyramidal lucas-kanade) لتتبع النقاط المفتاحية الخاصة للكائن المحدد في الصور المتتابعة حيث قد يوجد الكائن المحدد فيها. ليتم تتبع الكائن بشكل دقيق في كل الصور المتتابعة. تم استخدام النقاط المفتاحية المعبرة عن هذا الكائن لايجاد نقطة مركزية لتعتبر مركز الكائن المتتَبع. و قد تم استخذام نهج (Agglomerative cluster) لتجميع جميع النقاط المركزية المرشحة لان تكون مركز الكائن المراد تتبعه، ومن ثم تم حساب مركز المجموعة الناتجة لتمثل مركز الكائن المتتبع. في هذه الاطروحة تم استخدام قاعدتي البيانات المعيارية (VOT2014 dataset, Vojirtom dataset) لاختبار نظامنا الخاص لتتبع للكائن المحدد في مجموعة الصورالمتتابعة. تم استخدام معايير الدقة، المتانة، الاحكام والاستدعاء (Accuracy, Robustness, Precision and Recall) لمقارنة النظام الخاص بنا بالانظمة الرائدة في مجال تتبع الكائنات في الصور المتعاقبة (state-of-art object trackers). اظهرت النتائج لنظامنا المقترح المطبق على قاعدة البيانات (VOT2014 dataset) ان نظامنا حصل على 5.9 كمعدل لمعيار الدقة والمتانة في تجربة (sequence pooled) و حصل على 6.29 كمعدل لمعيار الدقة والمتانة في تجربة (per-attribute). واظهرت النتائج لنظامنا المقترح المطبق على قاعدة البيانات (Vojirtom dataset) ان نظامنا حصل على اعلى درجة من الاحكام بقيمة 0.882 بين الانظمة الاخرى الرائدة في هذا المجال، و حصل ايضاً على قيمة 0.9115 للاستدعاء. تظهر هذه النتائج ان النظام الخاص بنا قادر على ان يكون رائد في مجال تتبع كائن محدد في مجموعة من الصور المتعاقبة.