Situation Aware Object Tracking

A thesis submitted in partial fulfillment of
the requirements for the degree of

Bachelor of Technology

by

M. Ajay Narasimha, N. Rohit, T. Saicharan
(Roll No. 130102041,130102044,130102072)

Under the guidance of
Dr. Prithwijit Guha

DEPARTMENT OF ELECTRONICS & ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
April 2017

CERTIFICATE

This is to certify that the work contained in this thesis entitled

Situation Aware Object Tracking

is the work of

M. Ajay Narasimha, N. Rohit, T. Saicharan
(Roll No. 130102041,130102044,130102072)

for the award of the degree of Bachelor of Technology, carried out in the Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.

_____

Guide

Date: _____

Place: _____

DECLARATION

The work contained in this thesis is our own work under the supervision of the guides. We have read and understood the "B. Tech./B. Des. Ordinances and Regulations" of IIT Guwahati and the "FAQ Document on Academic Malpractice and Plagiarism" of EEE Department of IIT Guwahati. To the Best of our knowledge, this thesis is an honest representation of our work.

_____

Author

Date: _____

Place: _____

Abstract

This thesis is aimed at devising a robust algorithm for tracking objects. This has been one of the substantially challenging tasks in the field of Computer Vision so far. Most tracking algorithms are robust to some of the challenging situations or are trained to become robust. Our aim is to detect a challenging situation(if any) before it has occurred or as soon as it has occurred and switch to an algorithm that is robust to that situation if necessary. The challenges addressed are Occlusion and Illumination Change and the tracking algorithms used are combination of variations of Mean Shift and Stochastic Search.

Contents

List of Figures

Chapter 1

Introduction

1.1 Object Tracking

An ‘object’in this context is the set of pixels that are chosen in the first frame of a video which will be the ‘target’that is to be tracked in the further frames. This target chosen is usually a physical entity such as a car or a human. Tracking refers to predicting where the target object is located in each frame. The predicted region where the object is located is represented using a bounding region, which in the case of the data sets used is a rectangle.

1.2 Challenges in Object Tracking

The challenges faced frequently while tracking objects in videos addressed here are:

Occlusion:

Occlusion occurs when the target overlaps with the background or another object.We have provided detailed analysis of this challenge in the next chapter.

Illumination Change:

When an object moves from a dark region to an illuminated region or vice-versa, the intensities of its pixels change significantly. This change could pose a problem in tracking the target object.We will discuss in detail related to illumination change in the chapter 3.

Other challenges include scaling,deformation of object, rotation or motion of the camera etc.

Here we plan to detect the situation before it occurs or when it is about to occur and switch to the corresponding tracking algorithm that can handle it.

Chapter 2

Occlusion Detection Challenge

2.1 Previous works

2.1.1 Introduction

There are many tracking Algorithms which can handle various challenging scenarios. In most of these cases, such challenges are handled without detecting them i.e, trackers are trained to face such a challenge. Here we are trying to detect such a situation before it occurs or when it is about to occur. The problem with occlusion is that an object cannot be said that it would be occluded until some part of it is already occluded(In the case where the depth information in not derived).

There are many studies on occlusion in the past which can be classified on the basis of the types of occlusion that occur.Here we will not be much concerned about the self-occlusion in our discussions.We will mainly look into Inter-Object or Object-Background occlusions.Perhaps occlusion can easily be detected in a given scenario if it is recorded from multiple devices at various inclination, but this is not feasible in most of the real-life applications . So detecting the occlusion from a given 2D video is the real challenge.

Inter-Object Occlusion ie, occlusion between two moving objects in a given background. Here the occlusion can be detected by analyzing the distance between the center of objects and its 2d projection area w.r.t to camera viewpoint.

Optical Flow has been the traditional method of analyzing the 3D activity from a 2D video . Many of the studies related to occlusion are mainly based upon various types of optical flow and its patterns.As we know occlusion occurs mainly at edges, studying the flow of pixels

adjacent to the edges will help us to classify an edge from an occluding edge or non-occluding edge. But these study will be successful only in cases where the video has high frame rate or less pixel displacement between consecutive frames. Here we deal with data-sets with larger pixel displacements.In our experiments we use GunnerFarne Back Optical which is a dense i.e, operates over all the pixels of an image.

2.1.2 Problems with Classifier Training

Though occlusion detection is a binary class problem, we must be very specific in selecting the features for training. Training on image dependent features will mostly work for the similar object or similar scenario and doesn't provide any positive results on general videos with occlusion. In case of L1 minimization tracker[8], Even though training was done on the classifier with few specific videos,the results were universal because they were trained on the patterns of likelihood,not image features. Moreover we need a larger database of occlusion data-set to collect the data related to it.

2.2 Histogram of Oriented Optical Flow

To study the nature of motion of the tracking object,we use the vector property of the optical flow. Similar to HOG, we construct a histogram with a fixed bin size which represents the direction and also the nature of the tracking object whose coordinates are obtained from main tracking algorithm. HOOF feature is generally used to analyze the object movements on the video data.

2.2.1 Camera Motion Nullification

As discussed earlier, to compare the motion of the tracking object in series of frames we need to remove the external factors that cause changes in the optical flow. Previously there are many works based on optical flow techniques to estimate the camera motion. Here we approximate it by taking the average of all the flow magnitudes along x and y coordinates and subtract this from every flow vector of tracking object before constructing the HOOF feature.

$$f\prime_x = f_x - f_{x,cam} \approx f_x - f_{x,avg}$$

$$f\prime_y = f_y - f_{y,cam} \approx f_y - f_{y,avg}$$

2.2.2 Edge Flow Magnification

As we are using Farne-back optical flow which is dense,it is clearly observed that flow in non-edge areas is quite irrelevant to nature of object motion due to similar color/texture in the neighbourhood pixels. To overcome this, we give more weight to flow at edge pixels in the current tracking window to compute the histogram

$$|f\prime(x,y)| = \sqrt{(f\prime_x^2 + f\prime_y^2)}$$
$$\angle f\prime(x,y) = tan^{-1}\frac{f\prime_y}{f\prime_x})$$

2.2.3 Histogram Normalization and Scaling

Once the histogram is constructed,To overcome the variations in optical flow which might be caused because of change in frame rate or object motion w.r.t the direction of the camera, we normalize the obtained histogram and scale to a constant value.This will help us to check the variations in the histogram obtained from tracking windows in different frames.

2.2.4 Other features - Histogram of Oriented Gradient

Though we compensate few external effects (other than nature of object motion) by performing above operations in the construction of HOOF histogram, still there might be cases where the optical flow is getting effected due to exceptional cases such as sudden tilt or shake in the camera. This will surely interfere the optical flow in few consecutive frames. To avoid such scenarios, we further check the distortion in the Gradient histogram to confirm the occlusion. To perform this we also construct Histogram of Oriented Gradients.

2.3 Observations

Let M be the bin width of both the histograms constructed above and N be the total no of bins.

$$N = \left\lceil \frac{360}{M} \right\rceil$$

For every frame,we obtain a histogram, say H: [h_{f1},h_2,...,h_N].Here we need a way to compare the histograms obtained from series of frames[3]. Obviously we cannot proceed by assuming histograms as simple vectors as in Euclidean space. Space of Histograms is usually called Riemannian manifold with non-trivial structure. Usually Riemannian metric between points R1 and R2 is defined by $d(R1, R2) = cos^{-1}(R1^T R2)$.

[$\sqrt{h_1}$,$\sqrt{h_2}$,...,$\sqrt{h_N}$] is the square root representation of histograms. Using inner product between square root representation of histograms we can obtain kernel

$$K_s(H1, H2)) = \sum_{u=1}^{m} \sqrt{H1, H2}$$

Such histogram comparisons are mostly used for the Recognition of Human Actions in the videos [3].There are various other methods to compute the distances between two histograms and conclude in the variation of HOOF. We will discuss a method that is more relevant to appreciate the nature of the object motion.

2.3.1 HOOF and HOG

Based upon the nature of motion of the tracking object, each bin $h_{fi}(i = 1, 2...N)$ has a fixed means [μ_{f1}, μ_{f2},...,μ_{fN}] and variances [σ_{f1},σ_{f2},...,σ_{fN}].
Let us say,a static object moving towards right (car here), bins closer to 0° will have a higher mean and less variance.

Figure 2.1: HOOF1 $0^\circ \rightarrow 360^\circ$ - car sequence

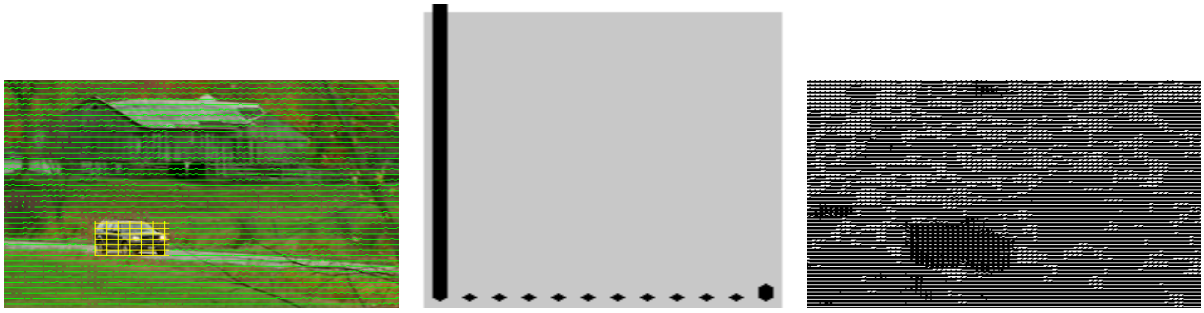


Figure 2.2: HOOF2 $0° \rightarrow 360°$ - car sequence

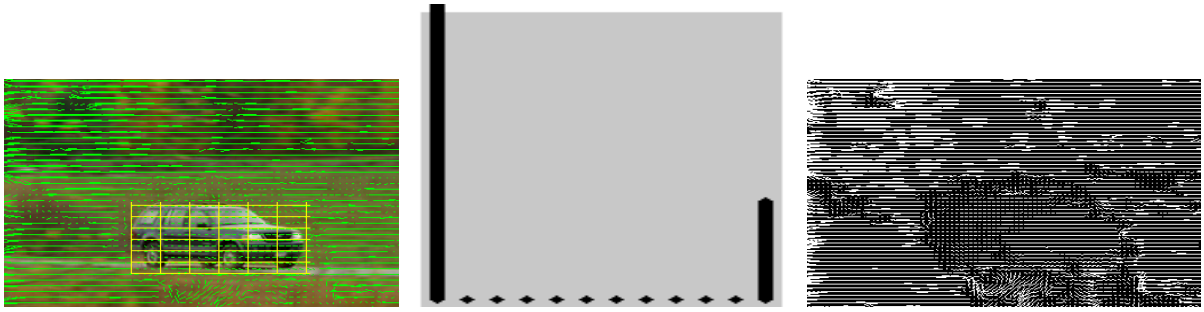


Figure 2.3: HOOF3 $0° \rightarrow 360°$ - car sequence

Now let us consider a woman jogging or walking,because of forward/backward of the limbs bin in the range of $90° \rightarrow 180°$ (forward motion) and $0° \rightarrow 90°$(backward motion) will have higher variance with a mean.

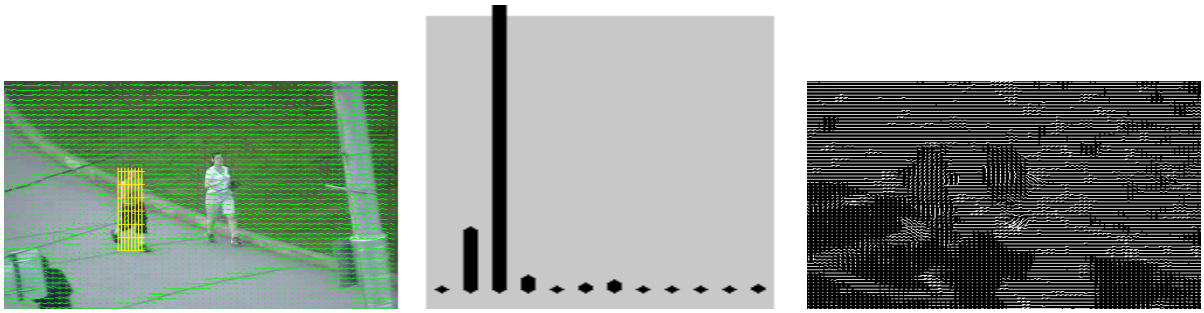


Figure 2.4: HOOF1 $0° \rightarrow 360°$ - jogging sequence

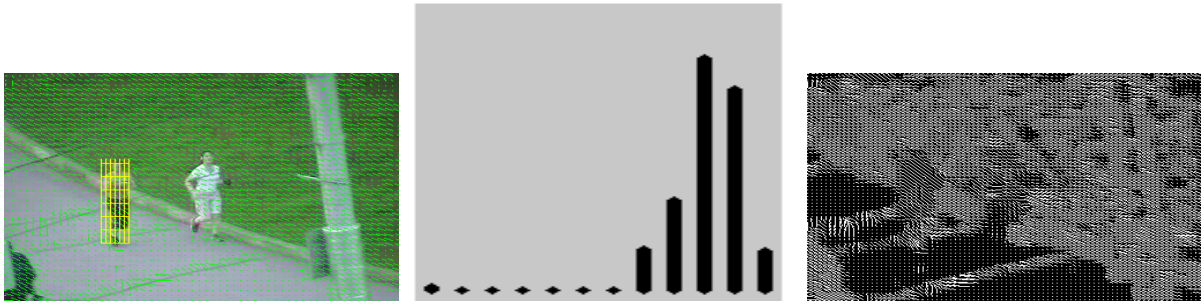


Figure 2.5: HOOF2 $0° \rightarrow 360°$ - jogging sequence

This occlusion detection method works for all the cases where the tracking object exhibits a unique property in the nature of motion through out the video (say walking, jogging, cycling, skating...etc). The means [μ_{f1}, μ_{f2},...,μ_{fN}] and variances [σ_{f1},σ_{f2},...,σ_{fN}] are learned in the first X frame (say X= 50) with an assumption that there is no occlusion occurring in the first few frames. Through our experiments by applying the Chebyshey's inequality to the bin values obtained in the x1,x2, . in the upcoming frames. we conclude the no of bins going out of the range of normal motion of the body which gives a chance of getting occluded.

Pr(| h_{fN} - μ_{fN} | $\geq K\sigma_{fN}$) = $\frac{1}{K^2}$

In few cases,flow change may occur due to sudden change in camera motion or any other circumstances. To further confirm the occlusion we also check the variation in the HOG histogram using the same Chebyshev's inequality on the histogram bin of HOG [h_{g1},h_{g2},...,h_{gN}] in the similar manner.Thus we conclude on the possibility of occlusion occurrence.

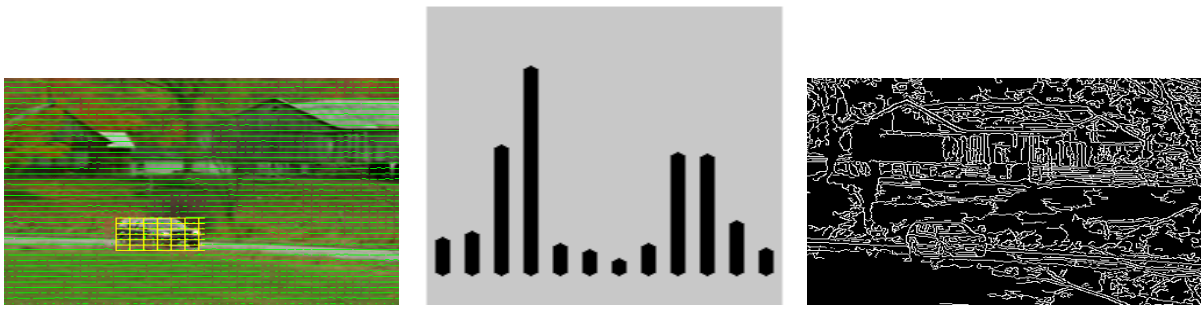


Figure 2.6: HOG $0° \rightarrow 360°$- car sequence

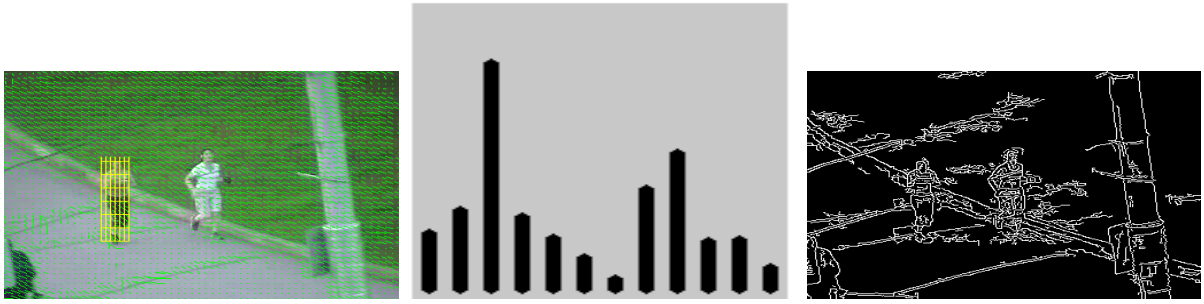


Figure 2.7: HOG $0° \rightarrow 360°$ - jogging sequence

2.4 Occlusion Results and Conclusions

VOT 2014- Jogging

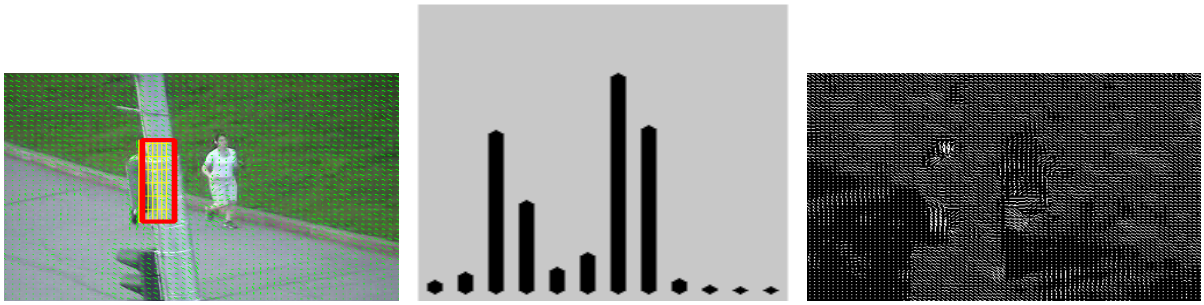


Figure 2.8: Occlusion Detection - jogging sequence

VOT 2014- Car

Figure 2.9: Occlusion Detection - car sequence

VOT 2014- bicycle

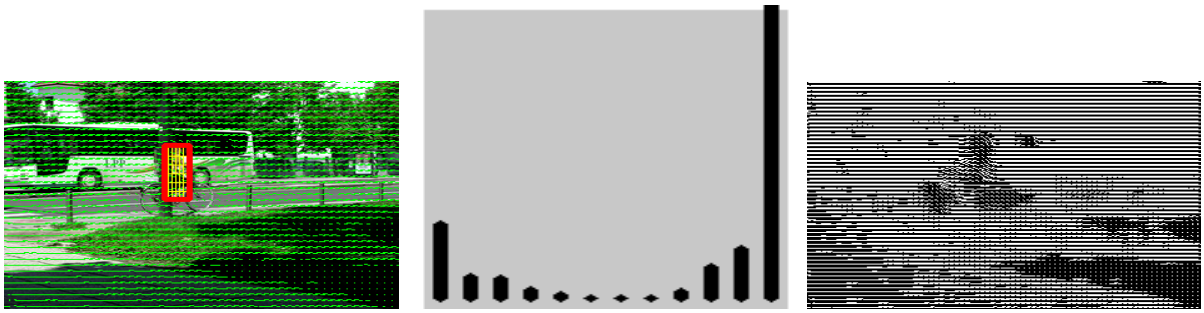


Figure 2.10: Occlusion Detection - bicycle sequence

VOT 2014- surfing

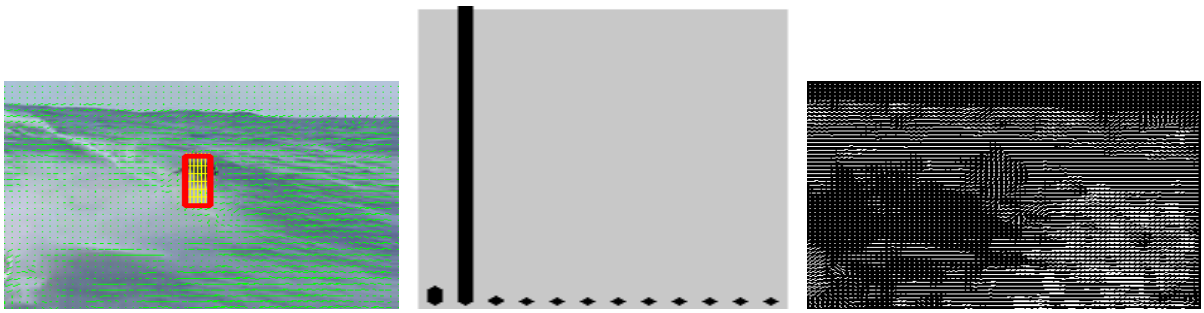


Figure 2.11: Occlusion Detection - surfing sequence

Results are more accurate only for the vot 2014 datasets where tracking object exhibit same nature of motion throughout the video. This method doesn't provide accurate results for the data-sets such as fernando.

Chapter 3

Illumination Change Detection

3.1 Introduction

Many tracking algorithms assume the presence of a single prevailing source of light with almost constant illumination in a scenario. However, this is not the case always. Change in illumination of the object or its surroundings in one of the significant challenges faced in object tracking.This challenge is faced due to the substantial change in the intensity values of the pixels of the same object when the illumination changes. This often leads to dissimilarity in the same object in different frames, posing a challenge in tracking it. An example of a significant change in illumination of the same object is shown below:

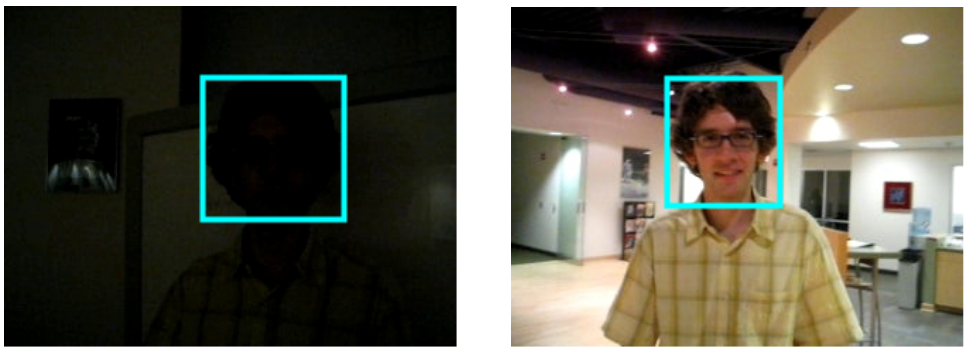


Figure 3.1: Same object with different illumination levels

The algorithm used to detect the change in illumination of the object is discussed in this chapter, the result of which will further be used to switch the tracking algorithm accordingly.

3.2 Concept

When an image is converted into frequency domain using the Discrete Cosine Transform, the information corresponding to the illumination level is predominantly in the lower frequency coefficients, including the DC coefficient[1] of the transform. Similarly, when an image is split into reflectance and illumination components[10], the illumination component has the information corresponding to the illumination level while the reflectance component is almost independent of it. These properties of images have been used to implement the algorithm to detect illumination change between frames in a tracking scenario.

3.3 Features

3.3.1 DCT Coefficients

The Discrete Cosine Transform[2] transforms an image into spectral sub bands of differing significance in the frequency domain. The 2D DCT transform for an NxN block is given by:

$$D(u,v) = \frac{1}{\sqrt{2N}} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) cos[\frac{(2x+1)u\pi}{2N}] cos[\frac{(2y+1)v\pi}{2N}]$$

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} \text{ for } u = 0 \\ 1 \text{ otherwise} \end{cases}$$

The features considered from DCT are the DC coefficient and three low frequency AC components i.e., the coefficients $AC_{0,1}$, $AC_{1,0}$ and $AC_{1,1}$ of the DCT as the information regarding the illumination predominantly lies in these.

3.3.2 Illumination Component

The intensity of an object in an image is due to reflection of light by that object in the scene. The pixel value of Lambertian objects can be modelled by:

$$p_j(k) = i_j(k).r_j(k)$$

for where $p_j(k)$, $i_j(k)$ and $r_j(k)$ denote the pixel intensity, illumination and reflectance component of the k^{th} pixel in the j^{th} frame. To separate the illumination component, a Homomorphic

filter is used. The image is first converted to logarithmic domain:

$$log(p_j(k)) = log(i_j(k)) + log(r_j(k))$$

The resultant is filtered using a 3x3 binomial kernel low pass filter and taking the exponent of the filter output gives the illumination component of the image. The pixel values of the illumination component have been used as features.

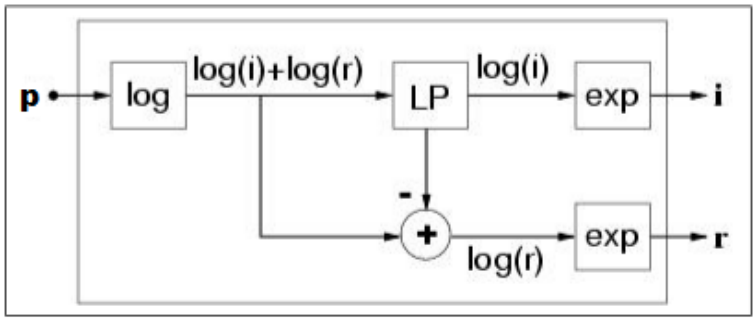


Figure 3.2: Splitting of Image into Reflectance and Illumination Components

3.4 Training Data and Testing

The training data has been generated using patches from different data sets where an illumination change is observed within the image(not between frames). Two patches of similar texture from a single image are selected such that there is a substantial difference in the illumination between the two. Each patch is divided into 8x8 blocks and the features mentioned above are extracted for each of the blocks i.e., the DC component $AC_{0,1}$, $AC_{1,0}$ and $AC_{1,1}$ of the DCT and the pixel values of the illumination component.

Figure 3.3: Selection of patches differing in illumination level

The feature vector which is input to the classifier contains the features of two blocks, taking all possible pair permutations among them. The label given is 0 for two blocks chosen

from the same patch since there is no illumination change between them and the label is 1 for two blocks chosen from either patch since there is an illumination change. This data is used to train a classifier and test the illumination change in the present tracking scenario. The tracker used when an illumination change is detected is discussed further.

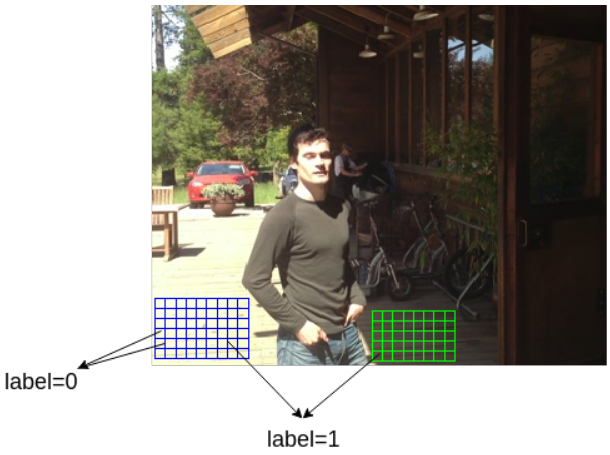


Figure 3.4: Illustration of Labels in Training

Chapter 4

Features

4.1 Edge Colour Oriented Histogram[7]

In the previous model, i.e., colour based Mean Shift tracking, the target model used an Epanechikov kernel weighted colour distribution from the pixels in the elliptical kernel region which encloses the target. In our proposition we use the pdf of Edge and Colour Orientation (HOG[4]) instead of colour distribution.

The Edge and Orientation data i.e., the magnitude and angle of the gradient for all pixels within the bounding box is evaluated as follows:

$$G_x(x,y) = I(x,y) - I(x-1,y)$$

$$G_y(x,y) = I(x,y) - I(x,y-1)$$

where I(x,y) is the Intensity, $G_x(x,y)$ is the gradient along x direction and $G_y(x,y)$ is the gradient along y direction at the location (x,y)

$$|G(x,y)| = \sqrt{(G_x^2 + G_y^2)}$$

$$\angle G(x,y) = tan^{-1}\frac{G_x}{G_y})$$

The magnitude ranges from 0-255 and the angle ranges from 0°-360°.

The colour orientation data was generated using the HSV colour model. The hue value ranging from 0°-360° represents the colour tone and the saturation values ranging from 0-255 represent the colour magnitude, assigning larger values for purer colours. The Edge colour Oriented Histogram(ECOH) is generated considering edge orientation as one channel and colour

orientation as another channel. ECOH is calculated as follows:
Histogram bin index B(i) is calculated from the gradient angle and hue values at any pixel location p

$$B(i) = H(i)n + \angle G(x_i, y_i)$$

where n is the number of bins per channel. It can take values from 1-360. The histogram is normalised like in the previous case. The ECOH represents the pdf of the edge and colour orientation.

Now the histogram is given by

$$q_u = \sum_{i=1}^{N} Ck(i)(S_i + G(x_i, y_i))\delta[B(i) - u]$$

where S_i is the saturation of the i^th pixel and the other notations are the same as above. This model is used to implement the mean shift tracking.

4.2 Local Binary Pattern

Local Binary Pattern(LBP) [9] is an operator that determines the value of a pixel based on the difference in the pixel value and the pixels in its 8-Neighbourhood. The value is given by

$$p = \sum_{i=1}^{8} 2^i N(i)$$

where N(i) is 1 if the pixel has a lower intensity value than its neighbour at i^{th} position and 0 otherwise. The position i starts at any one of its neighbours and is taken in the clockwise or anticlockwise direction. Since this function is dependent on the difference in the intensity values of a pixel and its neighbours, it is robust to uniform changes in the intensities of the pixels. If this image is used for tracking, it is expected to be robust to illumination changes since the intensities of all the pixels vary when the illumination changes.

4.3 SIFT

The scale invariant features are extracted using staged filtering approach. With various σ values, Laplacian of Gaussian (LoG) is found for the entire image. LoG acts as a blob detector which detects blobs in various sizes due to variations in σ. σ is a sclaing factor Let the scale space of

an image be $L(x, y, \sigma)$ resulting from the convolution of a Gaussian space $G(x, y, \sigma)$ Let the image be $I(x, y)$

$$L(x, y, \sigma) = G(x, y, \sigma) \times I(x, y)$$

As LoG requires a lot of computations, we approximate it to Difference of Gaussians (DoG).

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \times I(x, y)$$

where $D(x, y, \sigma)$ is the DoG blurring of image at scales σ and kσ.

Extract the local extremum over the scale and space. Assign those points the respective scales at which they are the extremum. Those points are the potential key points.

As the size of the object is very small, only few keypoints can be extracted. The data from these few descriptors is not sufficient for fitting a model. So, instead of entire object space, we considered 5x5 grids of object space and extracted local maximum in the grid space.

A 16 x16 neighbourhood around the keypoint is considered to create a 128 bin oriented histogram. It is represented as a vector to represent keypoint descriptors which hold partial invariance to local variations, such as affine or 3D projections, by blurring image gradient locations.

4.3.1 Creating Histogram using SIFT features[6]

All the SIFT keypoint descriptors extracted from the grids in the object space are clustered into M clusters using k-means clustering. For the object histogram, the centers of these clusters becomes the bins and the bin values depend on the number of features belonging to that cluster. This is similar to the histogram creation process in a bag-of-words approach used for object recognition

4.3.2 Keypoint Matching

Consider a candidate window around target window with small variations in centre of the target window. The keypints are extracted from the next frame from the candidate window and are

matched to the keypoints of the target window. The sum of squares of differences matched keypoint descriptors can be used as a metric for dissimilarity between the target model and candidate model.

4.4 Fusion of Features

All the features are fused to obtain the object model. Let RGB_{m_1} represents the color(RGB) histogram where

$$\sum_{i=1}^{m_1} RGB_{m_1} = 1$$

Let $ECOH_{m_2}$ represents the edge color oriented histogram where

$$\sum_{i=1}^{m_3} ECOH_{m_2} = 1$$

Let $SIFT_{m_3}$ represnts the SIFT features based histogram where

$$\sum_{i=1}^{m_3} SIFT_{m_3} = 1$$

Let q_u represents the object model

$$q_u = RGB_{m_1} \oplus ECOH_{m_2} \oplus SIFT_{m_3}$$

where the best feature is selected depending on the situation.

Chapter 5

Tracker Algorithms

5.1 Mean Shift Algorithm

The mean shift algorithm can be used for segmentation as well as tracking objects.

5.1.1 Kernel Function

The target is modelled by a kernel function (ellipsoidal function)[5] which determines weights of the pixels based on their distance from the centre of the bounding box given by

$$k(i) = 1 - (\frac{x_i}{a})^2 - (\frac{y_i}{b})^2$$

where x_i and y_i are the Cartesian co-ordinates of the pixel relative to the centre of the bounding box. This function assures higher weight to a pixel which is closer to the centre as it is more likely to belong to the object and vice versa.

5.1.2 Weighted Histogram

The weighted histogram is calculated using the weights obtained from the kernel function and RGB feature space. The size of this histogram is determined by the number of bins per channel(R, G, B). The histogram is calculated using

$$q_u = \sum_{i=1}^{N} Ck(i)\delta[B(i) - u]$$

where $u = 1....m$ are the bin numbers, B(i) is the bin index of the i^{th} pixel and C is the normalisation factor obtained from the constraint $\sum_{u=1}^{m} q_u = 1$. The bin index is given by

$$B(i) = bn^2 + gn + r$$

where b, g, r are the bin numbers of blue, green and red channels respectively and n is the number of bins per channel.

5.1.3 Similarity Function

The location of the object is predicted by maximising a similarity function which is the metric that determines degree of similarity between a given region and the target object. The maximisation is done iteratively in each frame and the predicted region is determined. The function used is the Bhattacharya Coefficient given by

$$\rho = \sum_{u=1}^{m} \sqrt{q_u * t_u}$$

where t is the target histogram and q is the histogram of the given region.

5.1.4 Updating the Predicted Region

The centre of the predicted region is iteratively updated until the Bhattacharya Coefficient is maximised. The new centre after the j^{th} iteration Y_{j+1} is given by

$$Y_{j+1} = \frac{\sum_{i=1}^{N} w_i^{(j)} g_i^{(j)} X_i}{\sum_{i=1}^{N} w_i^{(j)} g_i^{(j)}}$$

where $w_i = \sqrt{\frac{t_u}{q_{u_j}}} \delta[B(i) - u]$ and $g_i^{(j)} = -k'(i)$ for pixels in the bounding region centred at Y_j and X_i is the corresponding co-ordinate of the i^{th} pixel.

5.2 Stochastic Search Algorithm[11]

Search techniques, which explore multi-modal spaces are capable of finding the global optimum of an objective function. These techniques sample the search space in such a way that there is high probability of finding the near optimal solutions.

An important class of such techniques are Stochastic Algorithms where choices are done based on the randomness. In particular, Stochastic search algorithms refers to the minimization(maximization) of an objective function in the presence of randomness in the optimization

process. Even though there is no guarantee of convergence or finding the global optimum, stochastic search algorithms have been effective in finding the near optimal solutions to the complex problems. Let us describe the searching algorithm in detail.

Let us consider the problem of finding the optimal solution by minimizing an objective function y = f (x). For problems where maximization is required, the objective function is considered as negation of the original function.

Based on the prior knowledge about the problem domain, a search space $S = [x_{min}, x_{max}]$ is to be selected with x_{min} and x_{max} as the minimum and maximum values of the solution for the selected function respectively.

Consider a parent population P_t of size n, generated within the search space at t^{th} iteration as

$$P_t = \{x_i(t) = x_{min} + \eta_i(t)x_{max}; i = 1, ...n\}$$

where $\eta_i(t)$ is a random value from any specific distribution generated at t^{th} iteration.

Let Y_t be the set of objective function values for population P_t . A sequence of values U_t can also be obtained by removing the negative function values in Y_t given as

$$Y_t = \{y_i(t) = f(x_i(t)); i = 1, ...n\}$$

$$U_t = \{u_i(t) = y_i(t)min(Y_t); i = 1, ...n\}$$

The population is assigned a set of weights $w_i(t)$ such that

$$W_t = \{w_i(t) = \frac{u_i(t)}{\sum_{j=1}^{n} u_j t}; i = 1, ...n\}$$

Now,further search is done by generating children around each parent. The children are localized around the parent within a hypersphere of radius (typically set to small fraction of the search space size). The children around each parent $x_i(t)$ and their number are calculated as

$$C_i(t) = \{x_i(t) + \eta_i(t)\sigma\}$$

Better the objective function as parent has, the better to seach more in its neighbourhood than with parents with worse objective function value. The number of children generated around a parent is determined by the weight associated to it. Higher the weight more number of children will be generated around that parent. The number of children generated around the i^{th} parent is givemn as

$$|C_i(t)| = round(nxw_i(t))$$

Along with the children, new population is also generated by the linear combinations and random re-initializations. The population by the linear combinations is generated by selecting the random indices within the parent population as

$$P_t^{RLC} = \{x_i^{RLC}(t) = \eta_i(t)P_t(\eta_{r1}) + (1 - \eta_i(t))P_t(\eta_{r2}); i = 1, ...n_1\}$$

where η_{r1} and η_{r2} are the random indices within the parent population and n_1 stands for the new population generated by the linear combinations. Along with it, for injecting the new random solutions to the function along the search space, population has been randomly initialized within the search space as

$$P_t^{RRI} = \{x_i^{RRI}(t) = \eta_i(t)x_{min} + (1 - \eta_i(t))x_{max}; i = 1, ...n_2\}$$

where n 2 stands for the number of population generated by the random initializations. Thus, we will have a set of solutions corresponding to parents and children along with linear combinations and random initializations i.e.P_t , C_t , P_t^{RLC}, P_t^{RRI}

Then, select the best values Y_{optt} and start the next iteration with population corresponding to Y_{optt} as the new parent population P_t . This has been done iteratively for obtaining the optimal solution for the specified function.

Chapter 6

Proposed Tracker

We propose to track an object combining the mean shift and stochastic and sift feature matching algorithms, where the scale changes can be estimated using the stochastic search. SIFT feature matching gives the data of the point-wise correspondence which helps in tracking the object when it occludes. Consider an object bounding box with center (c_x,c_y) and width s_x and height s_y The state vector of the object can be represented as

$$\mathbf{x} = [c_x, c_y, s_x, s_y]$$

The objective function

$$Y = f(x) = \alpha \times d_{BhatCoeff} + (1-\alpha) \times d_{SIFT}$$

where $d_{BhatCoeff} = 1-\rho[q, p(y)]$,

d_{SIFT} represents sum of squares of difference between matched descriptors[12] The search space S is defined by the vectors x_{min} and x_{max} such that $x_{min} \leq x \leq x_{max}$

The proposed tracker updates the translation of the object using mean shift and stochastic iterations and updates the scale of the object using stochastic iterations. The entire procedure is explained in given algorithm below:

Algorithm 1 Proposed Tracker

1: Generate an initial set of n parent state vectors P_0 by randomly drawing samples from the search space S.

2: For each parent state vector, implement meanshift algorithm to compute Y_0

3: Start Stochastic Search Iteration: For each parent state vector P_t, generate m child state vectors C_t where m depends on the weight of parent sate vector from equation no

4: For each child vector, implement mean shift algorithm to compute Y_t^C

5: For each parent state vector P_t, genearate P_t^{RLC} , P_t^{RRRI}. Implement mean shift algorithm to compute Y_t^{RLC} , Y_t^{RRRI}

6: As Y is the dissimilarity function , find the least n from [Y_t , Y_t^C, Y_t^{RLC} , Y_t^{RRRI}] and store them in Y_{t+1} . Store the corresponding state vectors in P_{t+1}

7: IF [t ¡ T_{max}] THEN $t \leftarrow t$+1 AND CONTINUE the stochastic search from STEP 2 again.
ELSE TERMINATE AND RETURN $Y_{T_{max}}$ and corresponding $P_{T_{max}}$
where $P_{T_{max}}$ would represent the location of the object in the current frame estimated from the Proposed Tracker.

Chapter 7

Results

The performance of the proposed tracker was evaluated on different sequences in VOT_2014 Dataset with the performance measure Overlap Measure. Overlap measure is the ratio of intersection and union of the tracking bounding box and ground truth bounding box. The Average overlap is computed from the correctly tracked frames. Once the tracker completely drifts away from the object, we re-initialize the target model with groundtruth values of the frame.

Parameters Choosen for implementation:

For the proposed tracker,

The maximum number of iterations $T_{max} = 20$,

Parent population size n = 10,

Child population size = 25,

Range of search space of center (c_x, c_y) is $(c_x \pm 5, c_y \pm 5)$

Range of search space of width s_x is $s_x \pm 5$

Range of search space of height s_y is $s_y \pm 5$

The frame is said to be correctly tracked if overlap measure is greater than the threshold 33%. If for 5 consecutive frames, the overlap measure evaluated is less than the threshold 20%, the object is unable to track.Then the target model is re-initialized with the ground truth bounding box of the current frames.

Table 7.1: Performance Measure of Object Tracking using Edge and Color Orientation Feature space on different sequences of VOT_2014 dataset

Seq. No	**Sequence**	**Correctly Tracked Frames**	**Average Overlap Percentage**	**Re-initializations**
1	ball	600/602	64.53	0
2	car	203/251	44.96	1
3	bicycle	173/270	41.30	10
4	david	675/769	58.56	6
5	polarbear	353/370	63.51	1
6	sunshade	130/171	52.36	22
7	trellis	412/568	53.22	47
8	woman	395/596	62.45	14
9	fernando	204/291	57.65	28
10	tunnel	574/730	45.80	16

Results of Mean Shift Tracker using proposed algorithm for VOT_2014 dataset. The tracker bounding box is of cyan and ground truth bounding box is of yellow.

(a) (b) (c) (d)

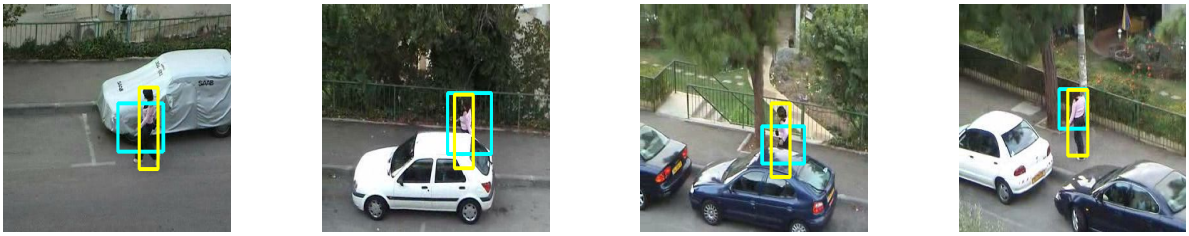
Figure 7.1: ball Sequence

(a) (b) (c) (d)

Figure 7.2: car Sequence

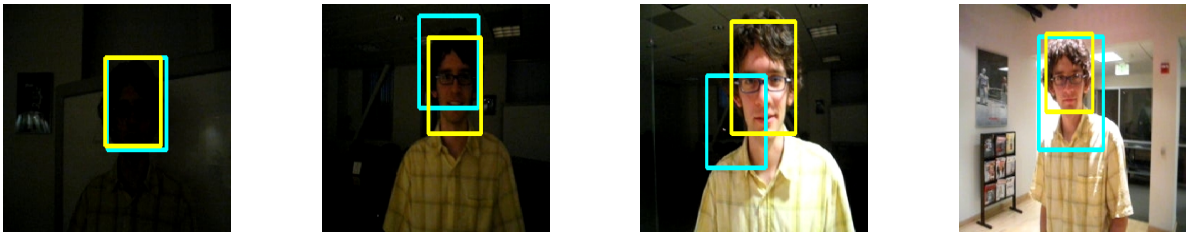
(a) (b) (c) (d)

Figure 7.3: bicycle Sequence



(a) (b) (c) (d)

Figure 7.4: woman Sequence



(a) (b) (c) (d)

Figure 7.5: david Sequence

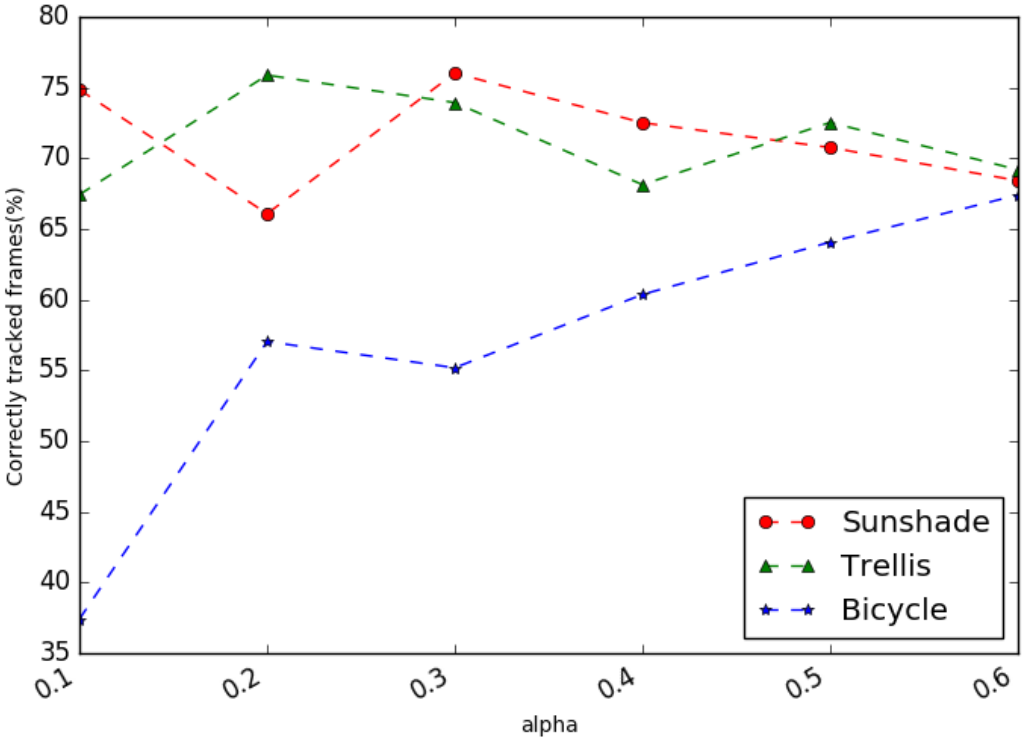


Figure 7.6: Variation in α for different data sets

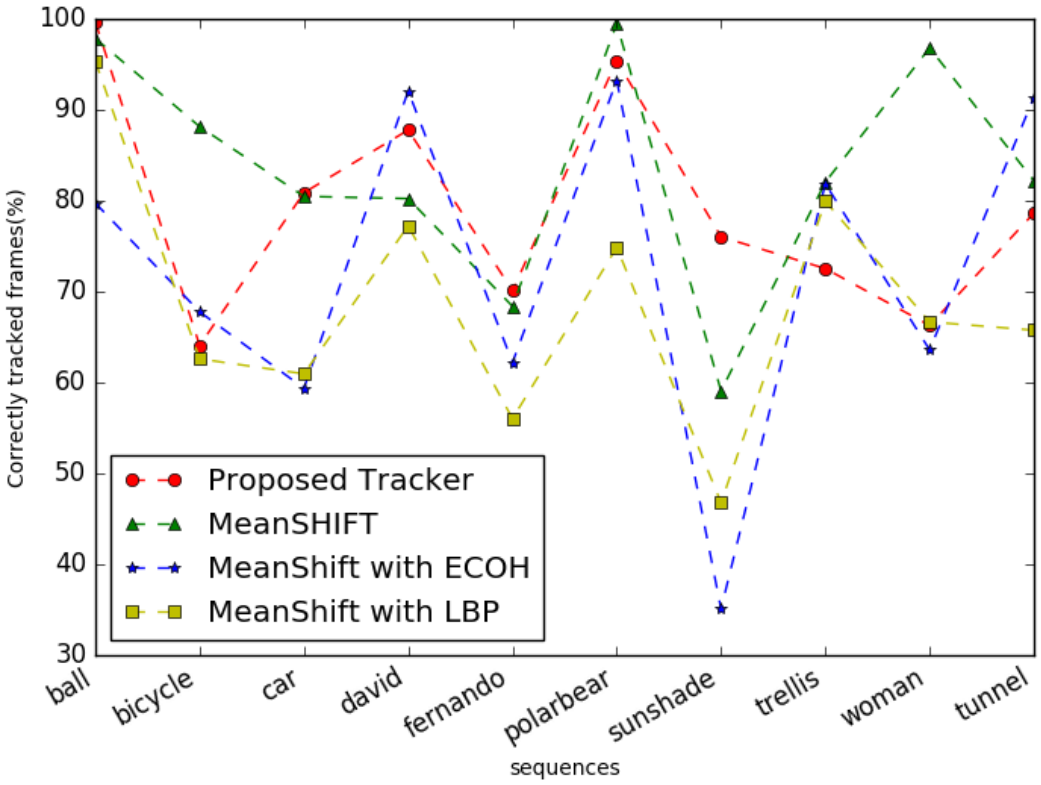


Figure 7.7: Comparison of percentage of correctly tracked frames evaluated with different trackers on various sequences of VOT2014 dataset

Chapter 8

Conclusion

The proposed tracker has outperformed MeanShift in sequences like ‘ball ’and ‘car’of the VOT_2014 dataset. The tracker adapts to the scale change in the object model and also the sudden drift of the object (fast movement) from the previous frame. For sequences like ‘woman ’in which occlusion occurs, the Mean shift algorithm along with SIFT feature correspondence performs better. Futher improvement can be expected when the rotation of the object is also added to the state model in the Stochastic Search algorithm.

Bibliography

[1] Javier Bracamonte et al. “A low complexity change detection algorithm operating in the compressed domain”. In: *Proceedings of the 8th COST 276 Workshop on Information and Knowledge Management for Integrated Media Communication*. Vol. 276. 8. COST (European Cooperation in the field of Scientific and Technical Research). 2005, pp. 7–12.

[2] Ken Cabeen and Peter Gent. “Image compression and the discrete cosine transform”. In: *College of the Redwoods* (1998).

[3] Rizwan Chaudhry et al. “Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions”. In: *computer vision and pattern recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE. 2009, pp. 1932–1939.

[4] Yu-Ting Chen and Chu-Song Chen. “Fast human detection using a novel boosted cascading structure with meta stages”. In: *IEEE Transactions on image processing* 17.8 (2008), pp. 1452–1464.

[5] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. “Kernel-based object tracking”. In: *IEEE Transactions on pattern analysis and machine intelligence* 25.5 (2003), pp. 564–577.

[6] Sourav Garg and Swagat Kumar. “Mean-shift based object tracking algorithm using surf features”. In: *Recent Advances in Circuits, Communications and Signal Processing* (2013), pp. 187–194.

[7] Wonjun Kim, Jimin Park, and Changick Kim. “A novel method for efficient indoor–outdoor image classification”. In: *Journal of Signal Processing Systems* 61.3 (2010), pp. 251–258.

[8] Suha Kwak et al. “Learning occlusion with likelihoods for visual tracking”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1551–1558.

[9] Timo Ojala, Matti Pietikäinen, and David Harwood. “A comparative study of texture measures with classification based on featured distributions”. In: *Pattern recognition* 29.1 (1996), pp. 51–59.

[10] Richard J Radke et al. “Image change detection algorithms: a systematic survey”. In: *IEEE transactions on image processing* 14.3 (2005), pp. 294–307.

[11] Ratnakaram Rajesh, Mathew Francis, and Prithwijit Guha. “Tracking under scaling and rotations using stochastic mean shift”. In: *India Conference (INDICON), 2015 Annual IEEE*. IEEE. 2015, pp. 1–6.

[12] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. “Object tracking using SIFT features and mean shift”. In: *Computer vision and image understanding* 113.3 (2009), pp. 345–352.