A GENERAL OBJECT TRACKER FOR LOCATING OBJECTS IN DIGITAL VIDEO

by

Işık Barış Fidaner

Bachelor of Science, Computer Engineering, Boğaziçi University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University

2008

A GENERAL OBJECT TRACKER FOR LOCATING OBJECTS IN DIGITAL VIDEO

APPROVED BY:

Prof. Lale Akarun .....
(Thesis Supervisor)

Dr. Ali Vahit Şahiner .....

Prof. Bülent Sankur .....

DATE OF APPROVAL: 20.06.2008

ACKNOWLEDGEMENTS

I would like to thank my supervisor Lale Akarun, for showing me a point to start and always guiding me for the next step until the end.

I thank Başar Uğur, Eser Aygün and Koray Balcı for their collaboration in previous projects and helping to grow the special kind of motivation that made this thesis possible; Uğur Güney for patiently listening and understanding all the mathematical details of my work; Melike Özen Çelik for helping me with the translation; Bahadır Maşa for his delicious meals, and other friends for asking me how the thesis was going.

ABSTRACT

A GENERAL OBJECT TRACKER FOR LOCATING OBJECTS IN DIGITAL VIDEO

Tracking a human head in a complicated scene with changing object pose, illumination conditions, and many occluding objects, is the subject of this thesis. A general tracking algorithm is presented, which uses a combination of object color statistics and texture features with motion estimation. The object is defined by an ellipse window that is initially selected by the user. Color statistics are obtained by calculating object color histogram in the YCrCb space, with more resolution reserved for chroma components. In addition to the conventional discrete color histogram, a novel method, Uniform Fuzzy Color Histogram (UFCH) is proposed. The object texture is represented by lower frequency components of the object's discrete cosine transform (DCT), and local binary patterns (LBP). By using the tracker, performances of different features and their combinations are tested. The tracking procedure is based on constant velocity motion estimation by condensation particle filter, in which the sample set is obtained by the translation of the object window. Histogram comparison is based on Bhattacharyya coefficient, and DCT comparison is calculated by sum of squared differences (SSD). Similarity measures are joined by combining their independent likelihoods. As the combined tracker follows different features of the same object, it is an improvement over a tracker that makes use of only color statistics or texture information. The algorithm is tested and optimized on the specific application of embedding interactive object information to movies.

ÖZET

SAYISAL VİDEODA NESNE YER TAYİNİ İÇİN GENEL BİR NESNE İZLEYİCİ

Bu tezin konusu, nesnenin duruşu ve ışıklandırmanın değiştiği, önünü kapatan başka nesnelerin bulunduğu koşullarda, karmaşık bir sahnede insan başını izlemektir. Hareket tahmini ile birlikte nesnenin renk dağılımı ve doku özelliklerinin birleşimini kullanan genel bir izleme algoritması sunulmaktadır. Nesne kullanıcının başlangıçta seçtiği elips bir pencere ile tanımlanır. Renk dağılımı kroma bileşenlerine daha çok çözünürlüğün ayrıldığı bir YCrCb uzayında nesnenin renk histogramının hesaplanması ile elde edilir. Bilinen kesikli histogramın yanısıra yeni bir yöntem olarak Tekdüze Bulanık Renk Histogramı (UFCH) önerilmektedir. Nesne dokusu nesnenin kesikli kosinüs dönüşümünün (DCT) düşük frekans bileşenleri ve yerel ikili örüntüler (LBP) ile temsil edilir. İzleyiciyi kullanarak farklı özelliklerin ve birleşimlerinin verimleri denenmektedir. İzleme yordamı, örneklem kümesinin nesne penceresinin ötelenmesi ile elde edildiği Koşullu Yoğunluk Tahmini (Condensation) parçacık süzgeci ile sabit hızlı hareket tahminine dayanır. Histogramların karşılaştırılması Bhattacharyya katsayısına, DCTlerin karşılaştırılması ise farkların kareleri toplamına (SSD) dayanır. Benzerlik ölçütleri bağımsız olabilirlikler olarak birleştirilmektedir. Birleşik izleyici aynı nesnenin farklı özelliklerini izlediği için yalnızca renk dağılımı ya da sadece yapı bilgisini kullanan izleyicilerden daha gelişmiştir. Algoritma filmlere etkileşimli nesne bilgileri gömülmesi üzerinde denenmekte ve buna uygun hale getirilmektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES vii
LIST OF TABLES x
LIST OF SYMBOLS/ABBREVIATIONS xi
1. INTRODUCTION 1
1.1. Organisation of the Thesis 4
2. OBJECT TRACKING 5
2.1. Feature extraction and selection 7
2.1.1. Color Histogram 8
2.1.2. Local Binary Patterns 17
2.1.3. Discrete Cosine Transform 18
2.1.4. Other features 19
2.2. Appearance modeling in time 21
2.3. Motion estimation 24
2.3.1. Kalman filter 25
2.3.2. Conditional Density Estimation 28
3. THE COLOR AND TEXTURE TRACKER 35
4. INTERACTIVE CONTENT CREATOR 40
4.1. Object Tracking Issues in ICC 40
5. EXPERIMENTS 42
5.1. Experiments using Video 1 44
5.2. Experiments using Video 2 46
5.3. Additional experiments 47
6. CONCLUSION 64
REFERENCES 66

LIST OF FIGURES

Figure 2.1. A conventional histogram and its corresponding fuzzy histogram . 10

Figure 2.2. 1D triangular function 11

Figure 2.3. 2D fuzzy membership function 12

Figure 2.4. A distorted cuboid and a regular cube 14

Figure 2.5. All possible combinations of 8-point LBP. [1] 17

Figure 2.6. Basis functions of an 8x8 DCT. 18

Figure 2.7. A simple 1D application of the Kalman filter 27

Figure 2.8. Processes in an iteration of Kalman filter. [2] 29

Figure 2.9. Processes in an iteration of Condensation. [2] 30

Figure 2.10. Steps of an iteration of Condensation. [2] 31

Figure 3.1. The filter for selecting lower frequency components of DCT. 36

Figure 3.2. Example tracking results for motion estimator that runs on color statistics and texture measurements 37

Figure 3.3. The processes in our tracking method. 39

Figure 4.1. Interactivity Content Creator 41

Figure 5.1. An example frame of Video 1 in three methods. 43

Figure 5.2. Fuzzy Ratio Histogram tracking in Video 1 49

Figure 5.3. FRH+DCT+LBP tracking in Video 1 50

Figure 5.4. Different histogram algorithms' performances on Video 1. 51

Figure 5.5. Fuzzy Ratio Histogram, LBP and DCT separate performances in Video 1. 52

Figure 5.6. Combined performance of Fuzzy Ratio Histogram with LBP and DCT in Video 1. 53

Figure 5.7. Comparison of our particle filter motion estimation with a Kalman Filter-based approach on Video 1. 54

Figure 5.8. FH+DCT tracking in Video 2 55

Figure 5.9. Different histogram algorithms' performances on Video 2. 56

Figure 5.10. Fuzzy Histogram, LBP and DCT separate performances in Video 2. 57

Figure 5.11. Combined performance of Fuzzy Histogram with LBP and DCT in Video 2. 58

Figure 5.12. Fuzzy Histogram + DCT tracking of Video 3 (Foreman) 59

Figure 5.13. Fuzzy Histogram with DCT tracking in Video 3 (Foreman). 60

Figure 5.14. Fuzzy Histogram + DCT tracking of Video 4 and 5 61

Figure 5.15. Fuzzy Histogram with DCT tracking in Video 4 (Coastguard). . . 62

Figure 5.16. Fuzzy Histogram with DCT tracking in Video 5. 63

LIST OF TABLES

Table 2.1. Table of invariances 8

Table 5.1. Single feature results for two videos 44

Table 5.2. Combined results for Video 1 45

Table 5.3. Combined results for Video 2 45

LIST OF SYMBOLS/ABBREVIATIONS

$h_{R,G,B}(r,g,b)$	RGB histogram value at the bin (r,g,b).
$n_{r,g,b}$	Number of pixels that fall in the bin (r,g,b).
$h_{A,B,C}(a,b,c)$	Histogram value at the bin (a,b,c) in ABC color space.
$P_{a,b,c\|j}$	Histogram membership function
$\wedge(x)$	Triangular function
$\sqcap(x)$	Rectangular function
$center_{a,b,c}$	A cluster center in UFCH.
$BC(h_{A,B,C}, q_{A,B,C})$	Bhattacharyya histogram similarity measure.
$r_{A,B,C}(a,b,c)$	Ratio histogram in ABC color space
$F(u,v)$	DCT response function.
μ_i	Mean of the ith cluster in K-means.
$v_{i,j}$	Membership function in K-means.
$\mathbf{x}_t$	Object state at time t in a Bayesian filter.
w_t	Process noise at time t in a Bayesian filter.
$\mathbf{z}_t$	Measured state at time t in a Bayesian filter.
v_t	Measurement noise at time t in a Bayesian filter.
$f_t(\mathbf{x}_{t-1}, w_t)$	Process function to obtain x_t in a Bayesian filter.
$h_t(\mathbf{x}_t, v_t)$	Measurement function to obtain z_t in a Bayesian filter.
$\mathbf{S}_t$	The sample set at time t in a particle filter.
$\mathbf{x}_t^n$	State of the nth sample after correction in a particle filter.
$\mathbf{x}_t'^n$	State of the nth sample after prediction in a particle filter.
π_t^n	Weight of the nth sample in a particle filter.

RGB	Red, Green, Blue Color Space
HSI	Hue, Saturation, Intensity Color Space
DCH	Discrete Color Histogram
UFCH	Uniform Fuzzy Color Histogram
FRH	Fuzzy Ratio Histogram
LBP	Local Binary Patterns

DCT	Discrete Cosine Transform
LPF	Low-Pass Filter
HVS	Human Visual System
MTF	Modulation Transformation Function
SSD	Sum of Square Differences
AOT	Articulated Object Tracking
SVD	Singular Value Decomposition
CFV	Characteristic Feature Vector
KDF	Kernel Density Function
OAM	Online Appearance Model
EM	Expectation Maximization
pdf	Probability density function
HMM	Hidden Markov Model
iid	Independent and identically-distributed
KF	Kalman Filter
Condensation	Conditional Density Estimation
ICC	Interactive Content Creator
PCA	Principal Component Analysis
NMF	Non-negative Matrix Factorization
VQ	Vector Quantization

1. INTRODUCTION

Tracking a moving object in a complicated scene is a difficult problem. If objects as well as the camera are moving, the resulting motion may get more complicated. As objects change pose, the surface geometry changes, causing drastic changes in surface illumination; and thus, the object appearance. Occluding objects and deformations in the object present further challenges. Particle-filter based approaches are employed to model the complicated tracking problem. Special object models that can descibe the object deformation and motion, such as a skeletal model for a human, simplify the problem. However, a general tracker that can locate and track general objects should not rely on such models.

A general-purpose object tracking algorithm is needed in many applications: There are several applications of object tracking including video compression, driver assistance, video post-editing. Some other applications are surveillance via intelligent security cameras, perceptual user interfaces that can make use of user's gestures or movements, or putting a 3d tracked real object in a virtual environment in augmented reality.

The special properties of the application field are usually exploited to come up with a more efficient method. A special object model, or simplifying prior assumptions are very common. Thus, a designer should determine some of the limitations before implementing a solution. There are several studies about tracking objects. Most are focused in different parts of the problem.

For constant camera applications, background difference methods are used. Some such methods are presented in [3], [4], [5]. Statistical appearance models help to distinguish foreground objects from the stationary background. Firstly, a set of statistical filters is applied to the background and the captured frames. Then, the foreground pixels are determined by comparing the responses. This is called background subtraction. In [5], color, motion and texture features were used.

After determining foreground pixels, blob tracking is used to process the information to reach a higher level of reasoning. In blob tracking, the foreground pixels are clustered in several blobs, and the detected blobs are tracked throughout the video. In blob tracking, an object is defined as a single blob (or a set of blobs) that may change shape as well as position during the video.

An improved multi-blob tracking method was developed in [3], whose main contribution is joining the separate steps (background subtraction and blob tracking) in a single process that is based on multi-object hypotheses and Bayesian probability theory.

Fixed camera techniques are usually multiple object trackers, because the cameras are located to busy places for tracking several people or vehicles. If the camera is not fixed, background subtraction is not possible, and the problem becomes considerably more difficult. Camera motion is a 3D transformation that affects the information processed in tracking as a whole. As the problem is more complex, most methods that involve camera motion are single object trackers.

It is possible to determine exact camera motion in 3D space and remove its effect from the video. However, this requires additional assumptions for object shape and motion. For example in [6], the object is assumed to have a rigid convex shape and no motion, as in an inanimate object like a building. Also, projection is assumed to be orthogonal, the focus of the camera lens being at infinity. In this case, if trajectories of several feature points are known on the video, the camera motion can be removed from these trajectories to reveal the exact 3D shape of the object.

In single object trackers, an important problem is occlusion, especially in a complex environment. The object being tracked may be partially or fully occluded for a short period of time. As we cannot model the background as a 3D collection of potential occluders, the occlusion is detected as the distortion of expected object appearance. Online appearance models, as presented in [7], present an approach to model the object in a feature space. In this technique, the change in object appearance in every frame is

represented as the combination of the stable object appearance, small random changes and uniform random values that are due to occlusion. If the occluded pixels are not detected, they cause the method to fail by either hiding the object, or distorting the object model.

If only a certain type of object is going to be tracked, such as a colored ball or a human body, a hard-coded object model can simplify the problem considerably. In [8], a 3D deformable face model is used. Before tracking, the model is registered to the human face, i.e. its parameters are adjusted to match the actual face. Then, the changes in the following frames are converted to movements and deformations of the simplified geometric face model. Articulated object tracking (AOT) is an approach for body tracking that involves a skeleton-like object model. In [9], two types of AOT were proposed: Decentralized AOT and hierarchical AOT. The decentralized method analyses the interaction between several object parts in a Bayesian conditional density propagation framework. The hierarchical approach improves the model further by grouping parts in higher level components and models the body in a hierarchical structure. For example, a human arm is a higher level component that consists of two rigid parts. Interpart relations are especially important for recovering the information lost by self-occlusion, such as a leg occluding the other leg, or an arm occluding the body.

If the objects have known behaviour patterns, prior assumptions can be made and application-dependent motion estimators can help the tracking. Kalman filter [10] and Condensation [2] motion estimation methods are two such examples. Some methods are supposed to run in real-time, and some do not have a time limitation.

In brief, object tracking is a very general problem, and in each approach, the methods are specialized for the particular field of application.

The application addressed in this thesis is the processing of movies for embedding extra information. In a movie, there may be multiple objects, occlusions, camera movement, and different types of objects moving through an arbitrary trajectory. How-

ever, object movements may be assumed to be smooth, and the scenes last only a few minutes. We develop a general tracking solution, which uses a combination of object color statistics and texture features with motion estimation. The object is defined by an ellipse window that is initially selected by the user. Color statistics are obtained by calculating object color histogram in the YCrCb space, with more resolution reserved for chroma components. In addition to the conventional discrete color histogram, a novel method, Uniform Fuzzy Color Histogram (UFCH) is proposed. The object texture is represented by lower frequency components of the object's discrete cosine transform (DCT) and local binary patterns (LBP). The general tracking procedure is based on constant velocity motion estimation by condensation particle filter.

In the application, we tracked the human head as an object, because it is appropriate due to its ellipse shape and movement patterns. However, the algorithm does not contain a structural assumption for human heads and can be adapted to a different types of objects.

1.1. Organisation of the Thesis

In Chapter 2, we describe the features used in tracking, the fitness measure, and the condensation-based tracking approach. Chapter 3 presents general framework of our tracking algorithm. In Chapter 4, we present the application which uses the object tracker: The Interactive Content Creator. Chapter 5 presents the results of our experiments and Chapter 6 concludes the thesis.

2. OBJECT TRACKING

The problem of object tracking is identifying the model parameters such as position or size of the object in each frame of the video. This is a complicated problem, since the object is a 3D object, and we observe its projection in a single image. Both the object and the camera are moving in time; and the illumination may change. The object may deform as well as move, and other objects may occlude it.

Our approach for the solution to the tracking problem is composed of several parts. In this chapter we are going to take a look at these components that should be considered in tracking objects in a digital video scene.

As a general component, most methods assume an object model. It may be a special model for a certain type of object, such as a human head, human body, or a vehicle. It may also be a general and flexible model that can represent many types of objects. The model is a function for matching features to determine where the object is. It may be a color distribution similarity, wavelet responses, template matching or any other model. In any case, the object changes its appearance throughout a scene (We can assume that this change is slow). Therefore, the model should be adaptable to changes in the object color, shape, or any other changing property.

The object model is used in conjunction with other models such as a motion model or an appearance model. Motion model defines a constraint in positional space-time, whereas the appearance model defines a constraint in the feature space-time. We are going to examine details in the following sections.

A window around the selection of features represents the placement of the object. The window can translate, scale, and in some cases, rotate. There may also be a more complex object window, e.g. a closed curve, connected blobs or a hierarchical skeleton of rectangles. In many tracking algorithms, a simple geometric window such as a rectangle or ellipse is used to simplify the process, even though the objects tracked are

quite complex. Such a simple window may also facilitate easier user interaction and output.

Another important constraint is the ending condition: When will the tracking stop? For example, if the scene changes, or the object permanently disappears, the tracking must automatically stop. There are also other problems like handling occlusion, and object birth and death. In [11], object detection is built into the particle filter. In this way, object birth and death probabilities can also be taken into account. A simpler approach would be to detect scene changes and to stop the tracking at the end of a scene.

Our approach consists of several sequential steps, which may be listed as:

1. *Feature extraction.* Features are the lowest level information that are extracted from the image. A raw video frame is a two dimensional matrix containing three dimensional (RGB) color vectors at each cell. Colors can be taken as a feature, they may be transformed to a different color space (YCrCb, HSI etc.), or a color histogram could be a better feature. There are other alternatives like transforming the image and extracting curve or edge information to use as feature vectors, or any combination of these.
2. *Feature selection.* The measurements taken from the image usually focus on a certain spatial area, not the whole image, or maybe an area on a different measurement space. A selection function is applied to the complete set of features on the image to obtain a weighted selection among those features. Different types of weighting functions may be used, e.g. a flat kernel, a gaussian kernel, a truncated gaussian kernel, thresholding a special function, or picking maxima of a function as important feature points.
3. *Appearance model.* The selection of features are to determine object movement, so they should be updated as the object moves from frame to frame. In the advancing frames, the feature points should be shifted to the new measured object position, by matching the nearby image features to the object model. Mostly used methods are mean-shift and gradient descent. As the points are spatially shifted

by matching the features, new measured object positions are obtained.

4. *Motion estimation.* As the object model is never completely accurate, and the image features are prone to error, the measured object position includes noise that is to be eliminated. This noise makes the measured trajectory inconsistent with an expected object movement. Noisy measurements can be corrected by the assistance of a prior assumption about possible object movement. A motion estimation algorithm can be applied, assuming constant velocity, constant acceleration or any other type of movement pattern, smoothing the trajectory by minimizing noise and extracting the hidden real object trajectory. Kalman Filter is mostly used for this purpose. Condensation [2] is another important motion estimation method.

2.1. Feature extraction and selection

The first step of the algorithm, the lowest level, is feature extraction. Initially, we have a series of RGB images. They have a certain resolution, frame rate and color resolution. They are quantized in three ways; spatially, temporally, and in color space. There are also boundaries of this information, starting and ending time, and the rectangular frame of view, to which the real 3D objects are projected. From this quantized, bounded and projected piece of information, a certain set of features are to be extracted. These features should have some properties to help track the object.

- The features should be spatially distinctive: The foreground object should not be confused with background elements
- The features should be temporally robust, less affected by changes in lighting and other conditions.

The invariances of used features are shown in Table 2.1.

Table 2.1. Table of invariances

Feature	Rotation	Intensity	Contrast
Color Histogram	Yes	No	No
LBP-VAR	Yes	Yes	No
DCT	No	No	No

2.1.1. Color Histogram

As we initially have RGB images, the first thing we can do is calculate the statistical color distribution of object pixels, the color histogram. We divide the 3D RGB color space into MxMxM equal cube-shaped bins, and count the number of pixels that fall in each bin, divided by total count. An RGB color histogram is defined by the following formula:

$$h_{R,G,B}(r,g,b) = Prob(R = r, G = g, B = b) \cong \frac{n_{r,g,b}}{N} \quad r,g,b \in \{0,1,...,M-1\} \quad (2.1)$$

The triplet (r,g,b) gives the location of a bin in 3D color space. Every continuous color value in the interval $[0,1]$ is quantized to an integer in range $0,1,...,M-1$, and R,G,B variables indicate quantized color values on the image. $Prob$ function gives the probability of a random pixel to fall in the bin at (r,g,b). It is computed by dividing $n_{r,g,b}$, the number of pixels that fall in that bin, by N, the total number of pixels on the image.

As a matter of fact, a histogram need not be calculated on an RGB image. It may first be transformed to a different color space. YCrCb or HSI have better properties than RGB. Hue and saturation of a color are less affected by lighting conditions, because the brightness is covered in the third coordinate, intensity. This brightness component is Y in YCrCb, and L in La*b*, which is a color-opponent color space. In a color-opponent color space, distances between colors are directly proportional to psychological human visual distinction.

For a general histogram the formula becomes:

$$h_{A,B,C}(a,b,c) = Prob(A=a, B=b, C=c) \cong \frac{n_{a,b,c}}{N} \quad a,b,c \in \{0,1,...,M-1\} \quad (2.2)$$

In this formula a single value (M) is used as bin count, but there may be different numbers of bins (e.g. M_A, M_B and M_C). A property of the histogram is that the sum of all probability values is equal to one:

$$\sum_{a,b,c=0}^{M-1} n_{a,b,c} = N \qquad \sum_{a,b,c=0}^{M-1} h_{A,B,C}(a,b,c) = 1 \quad (2.3)$$

The M^3 dimensional vector of these numbers is the histogram. This conventional definition of the color histogram bears some problems. In a discrete color histogram (DCH), a pixel falls into only a single bin. Consider the bins in a 4x4 2D histogram on the left in Figure 2.1. The points **X** and **Y** fall into the same bin, and the significant difference between them is neglected. Moreover, **Y** and **Z** are very close, but this information is lost as they fall in different bins.

Kernel density estimation can be used to improve the representation of a probability density. For example, Parzen window method is a nonparametric estimator, in which a window function is placed at every sample, and the density is considered to be the sum of these windows. By using this density function, a new probability value can be calculated for an unknown sample. In our application, we are not going to make estimations for new samples, but we are interested in comparing whole color density functions. Therefore, what we need is not a continuous kernel density estimation function, but a vector that represents the density, similar to the conventional histogram, but smoother.

The inherent problems of discrete color histograms were addressed in the context of image retrieval. In 2002, Han et al. derived the fuzzy color histogram (FCH) by modifying the mathematical definition of a histogram [12]. According to the total

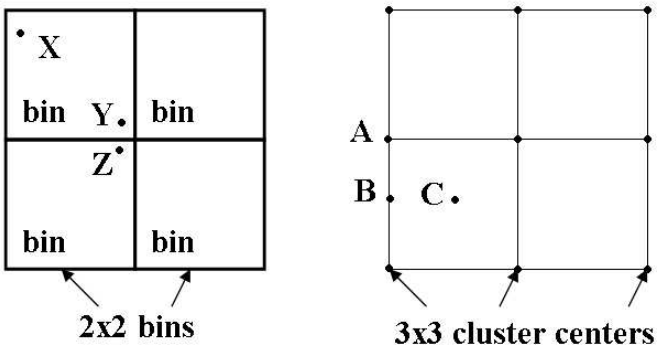


Figure 2.1. A conventional 2x2 histogram (left) and its corresponding 3x3 fuzzy histogram (right).

probability, a histogram can be defined as follows:

$$h_{A,B,C}(a,b,c) = \sum_{j=1}^{N} P_{a,b,c|j} P_j = \frac{1}{N} \sum_{j=1}^{N} P_{a,b,c|j} \quad (P_j = \frac{1}{N}) \tag{2.4}$$

P_j is the probability of selecting a pixel from the image, and is equal to $\frac{1}{N}$, where N is the number of pixels. Here, $P_{a,b,c|j}$ is the membership function that determines the behaviour of the histogram. In a conventional histogram, this function is either one or zero:

$$P_{a,b,c|j} = \begin{cases} 1 & \text{if the jth pixel falls in the bin (a, b, c)} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

In a fuzzy color histogram, $P_{a,b,c|j}$ is modified to be a fuzzy membership function that gives a real value in the interval $[0, 1]$ depending on the color difference between the pixel and the cluster center. In the original paper [12], the fuzzy c-means algorithm is applied to construct the fuzzy cluster centers. This is an unsupervised clustering method to find an optimum set of cluster centers that can represent certain data.

A recent paper [13] introduced triangular functions to extract fuzzy variables from image colors. Triangular function $\wedge(x)$ is a simple function in the shape of a triangle that is frequently used in signal processing (Figure 2.2 and Equation 2.6). They used HSI color space, and the color components were separately represented

by equidistant cluster centers that use a 1D triangular membership function. Then, these fuzzy variables were logically connected through fuzzy rules with premises and consequences. Our algorithm improves this idea by defining a 3D membership function based on 1D triangular functions that combines all the color components, unifying the fuzzy histogram in a single representation.

$$\wedge(x) = max(1 - |x|, 0) \tag{2.6}$$

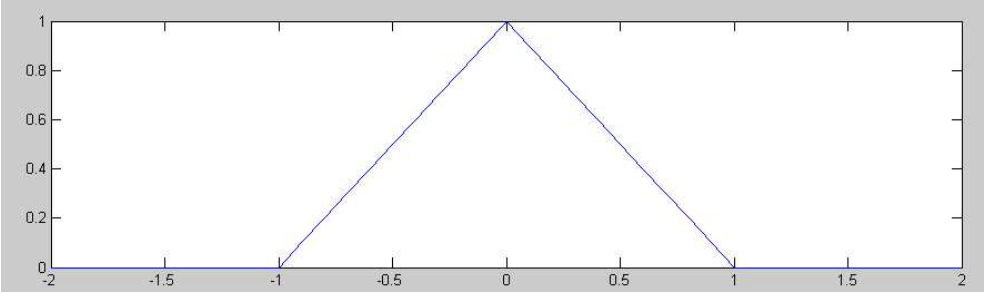


Figure 2.2. 1D triangular function that is used for fuzzy color representation.

The solution presented in this thesis is the Uniform Fuzzy Color Histogram (UFCH). In our method, the cluster centers directly correspond to the topological placement of the bins of a conventional DCH. The cluster centers of the UFCH are placed at the corner points that the bins intersect, as seen in Figure 2.1. Therefore, if M bins are allocated for a color axis in an DCH, the corresponding UFCH has $M + 1$ cluster centers for that color axis. The uniformity of this histogram is due to the equal distances between cluster centers in each of the color axes. The UFCH can be considered as a vector sampled from a continuous Parzen window estimation function, in which a triangular function is used.

In Figure 2.1, the point **A** belongs 100% to the cluster center at $(0, 1)$. **B** belongs 50% to cluster $(0, 1)$, 50% to cluster $(0, 0)$. The point **C** belongs 25% to each of the four clusters on the corners of its rectangular bin. The membership function of a cluster is defined as the multiplication of triangular functions in every color axis. In 2D, this

membership function is a peak shaped surface around a cluster center as shown in Figure 2.3.

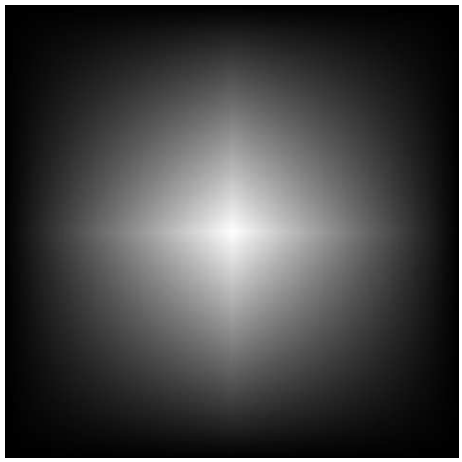


Figure 2.3. 2D fuzzy membership function as the product of two triangular functions.

In 3D, this membership function can be formulated as:

$$P_{a,b,c|j} = \wedge(da_j.M_A). \wedge (db_j.M_B). \wedge (dc_j.M_C) \tag{2.7}$$

where da, db and dc are the color components of the distance of the jth pixel from the cluster center:

$$color_j - center_{a,b,c} = [da, db, dc] \tag{2.8}$$

M_A, M_B and M_C are the number of bins in different color axes of the corresponding DCH. For the example in the Figure 2.1, these would be two for both axes. As every color value is in the range $[0, 1]$; width, height and depth of a bin are calculated to be the fractions $1/M_A$, $1/M_B$ and $1/M_C$ To scale the triangular function for A axis, its argument, the distance component da is divided into the bin width $1/M_A$, which yields $da.M_A$. The same procedure is followed for B and C axes. The membership is defined to be the product of three triangular functions.

In a 2D color space, a pixel is a member of a maximum of four clusters that are at the corners of a rectangular bin. In 3D, this is a maximum of eight clusters at

the corners of a cuboid, or 2^d clusters for higher dimensions, where d is the number of dimensions. Moreover, the sum of these 2^d functions is always equal to one. It is obvious for 1D, in the case of evenly spaced triangular functions. In 2D, if the functions sum up to one as shown in Equation 2.9, where x denotes $da_j.M_A$ and y is $db_j.M_B$. It can be proved that the sum is equal to one for any number of dimensions.

$$\sum P_{a,b|j} = xy+x(1-y)+y(1-x)+(1-x)(1-y) = xy+x-xy+y-xy+1+xy-x-y = 1 \tag{2.9}$$

If the triangular functions in UFCH are switched to rectangular functions as defined, UFCH becomes a regular DCH:

$$\sqcap(x) = \begin{cases} 1 & \text{if } 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

$$P_{a,b,c|j} = \sqcap(da_j.M_A).\sqcap(db_j.M_B).\sqcap(dc_j.M_C) \tag{2.11}$$

This equation is equivalent to (2.5). This formulation opens a new possibility. We can form special membership functions that can combine any number of discrete or continuous dimensions. Consider a special color space, where component A is discrete and the components B, C are continuous. We can combine them by a membership function as follows:

$$P_{a,b,c|j} = \sqcap(da_j.M_A).\wedge(db_j.M_B).\wedge(dc_j.M_C) \tag{2.12}$$

At the expense of losing the symmetrical properties of UFCH, it can be modified to a non-uniform fuzzy color histogram, in which cluster centers are not equidistant. In this case, the mesh defined by the cluster centers do not correspond to a regular grid structure, but a linear or non-linear transformation of a grid. For example, in RGB or YCrCb color space, cluster centers may be shifted to different positions, so that their distances are equal in a color-opponent color space such as La*b*. In this way, the histogram would behave as a color-opponent histogram without the computational cost

involved in color conversion. However, if we are to use a similar membership function, it would require the application of different transformations in different parts of the color space.

In the 2D version of Equation 2.7, rectangular color coordinates are transformed to square coordinates by scaling the triangular functions. This can be generalized to a transformation of any quad to a square. If the grid cell topology of the histogram is preserved, an image pixel would correspond to a quad in the mesh of cluster centers. In this case, the placement of the color inside the quad can be transformed to a placement in a square. Similarly, the placement of a color inside a distorted cuboid can be transformed to a placement in a cube as in Figure 2.4.

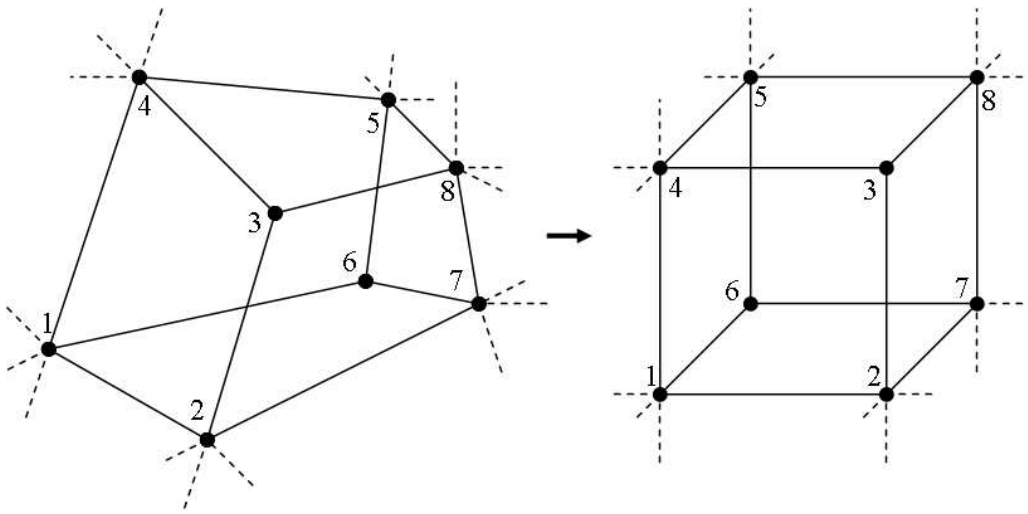


Figure 2.4. A distorted cuboid and a cube. Both constitute cells in the grid cell topology of a histogram.

Let a_i, b_i and c_i be color coordinates of the corners 1 to 8 of the distorted cuboid. If there is a 4x4 matrix **A** that transforms the corners of the distorted cuboid to the corners of a regular cube, we can also transform every color inside the cuboid to the

colors in the cube. The formula is given in homogeneous coordinates:

$$\mathbf{A} \times \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 & b_8 \\ b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 \\ c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 & b_8 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.13}$$

If we can calculate a rectified coordinate for a color value, then we can use the same membership function in (2.7) that is the multiplication of triangular functions. A small modification will be using the rectified color coordinates instead of $da.M_A$ etc. as the argument to the triangular functions.

If the cluster centers of a UFCH are shifted to different locations without affecting the topological structure, such as to simulate a color conversion function, we get the cluster centers of a non-uniform fuzzy histogram. In this case, the 3D geometric shapes that take the cluster centers as corners are distorted cuboids that lose their rectangular shape. To calculate membership of a 3D color, we must firstly find the distorted cuboid it belongs to, and apply the necessary transformation to rectify it to cube coordinates. Only then we can apply triangular functions to three color coordinates, and multiply them to obtain membership of that color to the 8 clusters around that cuboid. We used the uniform fuzzy color histogram in our experiments.

Another question is how to compare two histograms. The Bhattacharyya measure as presented in [14], gives a geometrical interpretation for this comparison. This measure is based on the fact stated above, that the sum of histogram values is 1. If we form a vector by taking square roots of all probability values in a histogram, this vector falls on the surface of the unit hyper-sphere. This hyper-sphere belongs to the $a.b.c$ dimensional space in which histogram vectors are defined. In this way, each histogram corresponds to a vector on the unit hyper-sphere, and the Bhattacharyya measure gives the cosine of the angle between two vectors these histograms correspond to. If the histograms are more similar, the cosine is nearer to 1.

Bhattacharyya coefficient is defined by the following formula:

$$BC(h_{A,B,C}, q_{A,B,C}) = \sum_{a,b,c=0}^{M-1} \sqrt{h_{A,B,C}(a,b,c) * q_{A,B,C}(a,b,c)} \tag{2.14}$$

In this equation, BC gives the similarity measure between two color histograms h and q. This measure is frequently used in histogram based tracking methods as in [15] and [16].

In our application, the object is initially marked by the user, so a histogram of the object region can be calculated, then compared to other frames to track the object. However, the background may contain some object colors, and the object region may have some background color.

There are ways of representing the color histogram relative to the object: If the object color distribution is not taken as absolute, but relative to the background, a larger region around the object, or the whole image, we can improve distinction of the object. In [17], Swain and Ballard proposed a method called Histogram Backprojection. This method is based on a *ratio histogram*, that is calculated by dividing each value in the object histogram into the corresponding value in the histogram of the entire image, and trimming the ratio if it is larger than 1. This calculation gives a strengthened distinction of the object from the background. In the following equation, h is the object histogram and q is the general image histogram. The fraction gives r, the ratio histogram:

$$r_{A,B,C}(a,b,c) = \min\left(\frac{h_{A,B,C}(a,b,c)}{q_{A,B,C}(a,b,c)}, 1\right) \quad a,b,c \in \{0,1,...,M-1\} \tag{2.15}$$

The histograms h and q may be any type of histogram (DCH or UFCH).

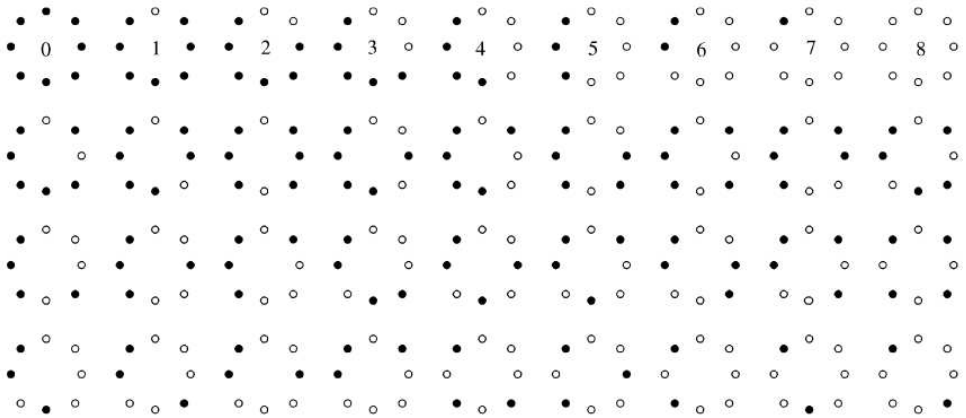


Figure 2.5. All possible combinations of 8-point LBP. [1]

2.1.2. Local Binary Patterns

Local Binary Patterns (LBP) are a new feature set for texture distinction proposed in 2002, [1]. Consider equidistant pixels on a circle around a point. For each pixel, a binary value is calculated: 0 if the pixel is darker than the center pixel, 1 if it is not. The values around the circle are taken as a cycle, in which shifted variations are not distinguished. All possible LBP values for 8 pixels are given in the Figure 2.5.

Among these several combinations, only the ones on the first line are considered and others are not counted. These LBPs that include less than three 0-1 transitions are defined to be "uniform", and others are discarded.

This feature is intensity invariant, because the values are relative; contrast invariant, because the magnitude of the difference is not taken into account; rotation invariant, because the values are on a circle, and there is no starting/ending point. In [1], at most 24 pixel circles are used.

A related feature, VAR, is also defined, being the statistical variance among these pixels on the circle. VAR is only related to image contrast, being approximately independent from LBP. Therefore, joint distribution of LBP and VAR happens to be a robust and distinctive feature for texture classification. In a very recent study, [5], LBP is used for tracking multiple objects on video.

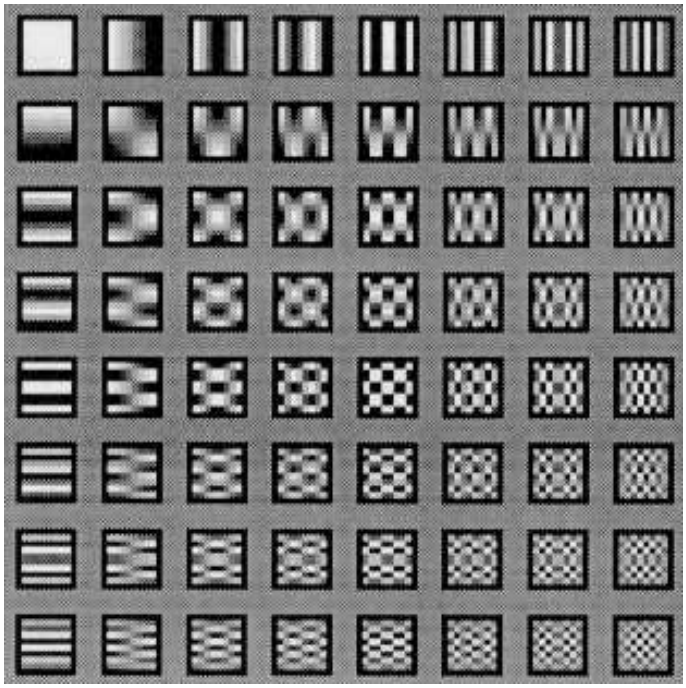


Figure 2.6. Basis functions of an 8x8 DCT.

As LBP is a discrete component and VAR is a continuous component, these can be joined by constructing a special membership function similar to (2.12).

2.1.3. Discrete Cosine Transform

Various image transformations can be used to amplify structural properties of the image, such as edge filters or DCT. Discrete Fourier Transform is a function that transforms the image from the spatial domain to the frequency domain. The greyscale image is considered as a weighted sum of horizontal and vertical cosine functions with different frequencies. These are called basis functions, and are shown in Figure 2.6.

If the image is an NxM matrix of greyscale color values, DCT is also an NxM matrix of response values. Every value gives the image's response to a particular cosine basis function. DCT of an image is calculated via the following equation:

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(i).\Lambda(j).cos\left[\frac{\pi.u}{2.N}(2i+1)\right] cos\left[\frac{\pi.v}{2.M}(2j+1)\right].f(i,j) \tag{2.16}$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } \xi = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.17}$$

In these equations, (i, j) defines the spatial coordinates whereas (u, v) define coordinates in the frequency domain. $F(u, v)$ is the response of the image to the basis function at that frequency pair. Low frequency components of a 2D DCT give high level structural information of an image. By using DCT, additive noise that is uniformly distributed on the image can be localized to higher frequency components of the DCT. Then, after eliminating these components, inverse DCT $F^{-1}(u, v)$ gives an image similar to the original. Eliminating higher frequencies is also called a Low-Pass Filter (LPF) and it results in a blurring effect.

In some applications, the videos are compressed and already include the structural information that can help tracking. In [18], the transform coefficients in MPEG videos were used for tracking purposes. In these videos, there are single I-frames followed by a sequence of P-frames. An I-frame includes a DCT, and defines the picture. In the following P and B frames, changes are conveyed differentially by motion vectors. However, as the purpose is different, not all motion vectors correspond to object motion.

2.1.4. Other features

The features discussed so far are low-level features, which disregard the 3D structures in the images. It may also be possible to represent higher level structures, by extracting corners, edges, intersections of lines and representing planar regions. Color and Texture features of different regions can be extracted separately.

If the feature points are known to be part of a rigid body, with additional assumptions of projection, the 3D shape can be extracted. In [6], the matched feature points were processed by a linear operation to get real 3D shape of the object being tracked.

The extracted features of the image object may be subjected to a feature selection

algorithm. Feature selection methods are used to discriminate between good features and bad features.

To select feature points, the study in [19] uses a Gabor function. Peaks of this function indicate important points on the image. After being selected, these points are matched with points on other frames. The simplest approach would be block matching, directly comparing pixels around these points. A better result could be obtained by multi-level features with different frequencies and orientations, such as wavelets.

In [20], Shi and Tomasi developed a method to determine which feature points are more appropriate for tracking. A window around a feature point is compared to translations and affine transformations of the same window on the following frames. The difference, dissimilarity of translated and affine tranformed windows from the first frame is computed. If the window contents change very significantly (above a threshold), that is a bad feature point and omitted.

In [6], Tomasi and Kanade presented an algorithm (referred as KLT transform) to track a 3D object from its known feature points in 2D video. The 2D frames are modeled as isometric projections of the rigid object onto the camera plane. And the movement is considered not to be the movement of the object, but the camera. Then the problem becomes similar to multiview geometry, in which every frame is a different viewing angle to a static rigid object. Isometricity and rigid body assumptions make a linear solution possible.

In KLT [6], all the measurements of feature points are collected in a single matrix, as image coordinates over all video frames. If the object is an orthogonally projected 3D rigid body, this matrix must be of rank three. If it has more rank, this is due to error. In this case, singular value decomposition (SVD) is applied, and only the first three singular values are considered. Row and column vectors corresponding to other singular values are eliminated, similar to a selection of features.

In [21], an occlusion resistant tracking method was developed based on the KLT

transform. After initial selection, the method eliminates some of the tracked feature points, whose motion coordinates are outside a statistical variance. When the object is partially occluded, lost feature points are detected and their positions are estimated until they are detected to appear again. If they were not eliminated, bad points would give wrong information and mislead the tracker.

An alternative to KLT was developed in [22], based on deformable surface model. 2D image is regarded as a 3D deformable surface, taking intensity as the Z component. Then, by only considering movement along Z axis, a physical motion equation is formed. In this equation, the values of the generalized displacement vector (also called characteristic feature vector, or CFV) are exploited to find good feature points.

As in [23], features can be weighted to optimise the information for human visual system. In this study, the DCT components are weighted to favor lower frequencies that our visual system is more sensitive to. The modulation transformation function (MTF) that gives the corresponding weights, is based on coefficients that are determined by actual psychological experiments.

2.2. Appearance modeling in time

The appearance of the object is derived from relevant feature values from the video frame. For the initial frame, object measurements are made by feature extraction and selection. As the object changes its appearance, its shape and color in time, the measurements that define the object must be adapted to match the updated appearance. New measurements may be directly adopted, the old measurements can be blended with new ones, or a special iterative process can be implemented for updating the model, such as mean-shift.

In the feature space of tracked images, an appearance model should be able to discriminate object pixels from background pixels. K-means is a clustering algorithm that partitions a set of data into several clusters (also called partitions). Let's assume we have n samples to be distributed to k partitions ($n >> k$). In the beginning, every

data point can be assigned randomly to the partitions, or a heuristic can be used. K-means is an iterative algorithm that gradually converges to a better clustering. In every step, partition centroids are calculated by the following formula:

$$\mu_i = \frac{\sum_{x_j \in S_i} x_j}{|S_i|} \tag{2.18}$$

S_i denotes the ith cluster, and x_j is the jth sample. Every x_j is assigned to the partition with the closest centroid.

$$v_{i,j} = \begin{cases} 1 & i = \arg\min_i |x_j - \mu_i|^2 \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

$v_{i,j}$ is the membership function of jth sample x_j: It is one if x_j belongs to cluster S_i. After reclustering, the centroids are recalculated and the other steps are repeated until convergence. K-means always converges after finite iterations to a point where clustering does not further change. Throughout these iterations, every cluster is updated to consist of less varied samples. K-means minimizes total intra-cluster variance. This total error can be calculated by the following formula:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{2.20}$$

Mean shift is a more general algorithm; K-means is a special case. It allows soft clustering as well as hard clustering. In hard clustering such as K-means, a sample may belong to only a single partition. In soft clustering, the membership function is expressed in terms of a percentage that gradually changes from one cluster to another.

In [24], the mean shift algorithm is defined, explained and analyzed. This is a method in which, each data point is shifted to the mean of its neighborhood. In mean shift, a kernel density function (KDF) is selected to determine the neighborhood size and shape. The algorithm is an iterative process that divide the dataset into clusters, and data points are approximately shifted to cluster centers by following the ascending direction, which is articulated as "mode seeking". A theoretical relation, "shadowness"

between KDFs is defined: Gradient descent direction of the "shadow" KDF is equal to the mean-shift direction of the KDF itself. It is also proved that the only kernels that are their own "shadows" are the gaussian kernel and truncated gaussian kernel.

In [25], Comaniciu et al. presented the theoretical grounds of a mean shift based nonparametric method for clustering a complex multimodal color space. In [15], they implemented a successful object tracking algorithm, which uses histogram similarity as the matching feature and mean-shift iterative process to follow the object. They frame the object being tracked in an elliptical window that has several degrees of freedom, allowing scaling as well as translation in both axes. For comparing the object model with nearby object candidates in adjacent frames, color histogram similarity is calculated via Bhattacharyya coefficient. If this coefficient turns out to be large, this means two discrete distributions (the color histograms) have small distance in between, i.e. they are similar. Depending on this similarity function, the mean-shift algorithm is ran with isotropic kernel density function. The object model is thus shifted to the maximum similarity point, and the window is updated to the new position.

In their paper [16] Polat and Ozden present their object tracking algorithm based on mean-shift. They used a rectangular window, and the object is modeled by its color histogram. To determine object pixels, motion is considered in addition to color information. Probability of a pixel is calculated by taking into account both motion and histogram similarity by Bhattacharyya coefficient. The motion is detected by color changes of a pixel in recent frames. The weight image is updated by mean-shift, and the rectangular window is moved to the centroid of the new weight image.

In [7], online appearance model (OAM) framework was developed. They use wavelet responses as image features. The features in single frames are assumed to be a mixture of three types of sources: A stable image structure that defines the exact feature combination that defines the object, a feature change that accumulates through adjacent frames, and a uniform error function that defines occlusion. Every object pixel has a stable component, wandering component and a lost component respectively. From the features in recent frames, they use expectation maximization (EM) to extract the

three components' weights that define the appearance model parameters. For example, if the object changes its appearance, the stable component drops and the wandering component jumps; then slowly the stable component increases and wandering decreases. When an occlusion occurs, the stable component falls and the lost component jumps.

2.3. Motion estimation

If we know that the object follows a certain motion pattern, e.g. constant velocity or constant acceleration, we can use this information to minimize the error in measurements. The knowledge of a certain motion pattern changes the probability density function (*pdf*) of the trajectory of the tracked object. To calculate posterior *pdf*, Bayesian filters are used.

There are different types of Bayesian filters, but in general they all try to obtain a final object state, given the initial state and a series of measurements:

$$p(\mathbf{x}_t|\mathbf{x}_0, \mathbf{z}_{1:t}) \tag{2.21}$$

In this equation, $\mathbf{x}_t$ gives the object state at time t, and $\mathbf{z}_t$ designates the measured object state at time t. The Bayesian filters are based on the following assumptions:

1. The current state only depends on the previous one, i.e. the state changes as a Markov process.

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, ..., \mathbf{x}_0) = p(\mathbf{x}_t|\mathbf{x}_{t-1}) \tag{2.22}$$

2. The current measurement only depends on the current state, i.e. it is the observed state of a Hidden Markov Model (HMM).

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{x}_{t-1}, ..., \mathbf{x}_0) = p(\mathbf{z}_t|\mathbf{x}_t) \tag{2.23}$$

In [26], the stages in Bayesian filtering are explained in detail. The calculation

of the final *pdf* can be considered as a recursive process composed of several steps. Obtaining the *pdf* at $\mathbf{x}_t$ in terms of the *pdf* at the previous state $\mathbf{x}_{t-1}$ is called the prediction stage, which is formulated by using the Chapman-Kolmogorov equation:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})\ p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1})\ d\mathbf{x}_{t-1} \tag{2.24}$$

This *pdf* is only a prediction, because it does not consider the current measurement $\mathbf{z}_t$. A second stage is necessary to obtain the conditional *pdf* $p(\mathbf{x}_t|\mathbf{z}_{1:t})$. The second stage is called correction, and it is based on the Bayes' rule:

$$p(\mathbf{x}_t|\mathbf{z}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{x}_t)\ p(\mathbf{x}_t|\mathbf{z}_{1:t-1})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1})} \tag{2.25}$$

In the correction equation, the denominator is a constant value that only depends on the measurement model $p(\mathbf{z}_t|\mathbf{x}_t)$:

$$p(\mathbf{z}_t|\mathbf{z}_{1:t-1}) = \int p(\mathbf{z}_t|\mathbf{x}_t)\ p(\mathbf{x}_t|\mathbf{z}_{1:t-1})\ d\mathbf{x}_t \tag{2.26}$$

In the correction step, the current observation $\mathbf{z}_t$ is used to modify the *pdf*. Now that we have $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, the next step is the prediction of $\mathbf{x}_{t+1}$, and the process continues until all the observations are processed.

2.3.1. Kalman filter

Kalman Filter (KF) is the most well-known Bayesian filter. It was introduced by R. E. Kalman in 1960, [10]. It is a recursive filter to determine the position of an object with noisy movement model and noisy measurement conditions.

To apply the KF, two models are formed. The first is the process model, that relates actual object variables between two adjacent frames, through a linear function plus a gaussian error:

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + w_t \tag{2.27}$$

The object state $\mathbf{x}_t$ may include all or some of the object variables that indicate its position, velocity and acceleration. The linear relation defined by matrix A may be a constant velocity model, constant acceleration model, or a completely different movement model. The noise factor w_t is considered because in actual applications, movement is not deterministic, the object never perfectly follows the movement model, and it may cumulatively deviate from it. This recursive function covers a very general stochastic motion model. If the noise is assumed to be gaussian with zero mean and a certain standard deviation, the solution is quite simple.

The second model is the measurement model, that relates actual variables at an instant to their measured values, again through a linear function plus a gaussian error.

$$\mathbf{z}_t = H\mathbf{x}_t + v_t \tag{2.28}$$

The matrix H, the measurement matrix, usually limits the measurement to only some of the process variables. In the simplest case, only the object position is included as a measurement. The velocity may also be included, if there is a radar that directly measures object velocity. v_t is a noise factor similar to w_t. It is also assumed to be a zero mean gaussian distribution to simplify the solution. In a typical example, process model could be a constant velocity movement with some small error, and the measurement model could be the measuring of the object position with a relatively high error.

In the beginning, the object state is known as a gaussian probability distribution. The first step is prediction, in which the process model is applied to the current distribution (at t=0) to obtain the predicted distribution at t=1. Since both the state and the observation variables are gaussian random variables, one only needs to update the appropriate mean and variance parameters at each time instant. The second step is correction, in which we make a measurement at t=1, and find the updated probability distribution of the object state in condition with this measurement. After correction, another prediction is made for t=2, and is corrected by a measurement at t=2. The method continues iteratively until the end.

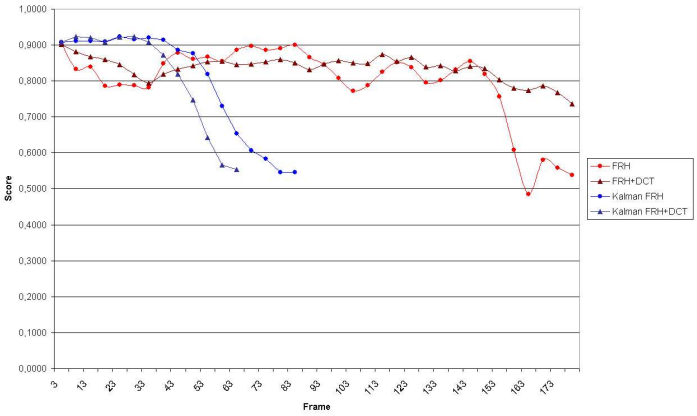


Figure 2.7. A simple 1D application of the Kalman filter with the constant velocity model. Green line is the actual object motion, blue line is the measured positions. Red smooth line is the estimated motion path obtained by applying Kalman Filter.

At first, we have the measured object states, a randomly jumping function that roughly follows the object trajectory. After applying the filter, we get a smoother function that gives the line of estimated movement. How smooth the function will get depends on the given noise parameters w_t and v_t. An example is shown in Figure 2.7

In a 2004 study, [27] Nguyen and Smeulders developed a simple and powerful object tracking method based on template matching. They used the KF for motion estimation. Appearance is modeled by photometric feature vectors that are defined by transforming the color values to make the variable independent of luminance. Measurements are made by fitting a rotating rectangular template to the current frame. Motion model of the KF is only a gaussian process noise, for covering the changes caused by object rotation and possible changes in illumination conditions as well as object movement. In this constant position model, the velocity is unpredictable and modeled by noise. The measurement error is calculated not by sum of square distances, but by the Mahalanobis distance with a scaling matrix. The measurement model is altered de-

pending on the error, so that the tracking is unaffected by short periods of occlusion. Outlier measurements are determined by using Huber's function and eliminated. A template is fitted to the new frame, by iteratively shifting it along the gradient descent direction in its nearby positions in the template space defined by rotation, scaling and translation.

A typical application of the Kalman Filter is given by Melo et al. in [4]. They used KF as a means to find highway lane curves by tracking car movements through a static camera. As the camera is static, a dynamic background difference method is implemented that is unaffected by slow changes like the day-night difference. The foreground pixels are clustered in connected sets that are called blobs. KF process model is constant acceleration movement, including velocity and position, whereas the measurement model is only the position of the car, both containing the noise term. How many cars that a blob contains is determined, depending on increasing blob size and decreasing blob count. After the cars are tracked and several trajectories are formed, highway lanes are obtained by clustering these trajectory curves by an algorithm similar to k-means, and the cluster mean curves give the highway lanes.

2.3.2. Conditional Density Estimation

Condensation (Conditional Density Estimation) is a particle filter developed by Isard and Blake [2] that combines Kalman filtering ideas with higher level image modeling and Bayesian sampling of a more general distribution. In KF, the estimation is based on probabilities obtained using two models, measurement model and process model, both usually assumed to be gaussian probability distributions. Calculations are based on the parameters of these distributions, namely mean and covariance. In contrast, condensation involves a sample set that represent the distribution, and every calculation is a statistical approximation based on this sample set.

A step in Kalman filter is shown in Figure 2.8. An iteration in Kalman filter consists of prediction and correction. The prediction is made by directly applying the process model. In the figure, deterministic drift and the stochastic diffusion correspond

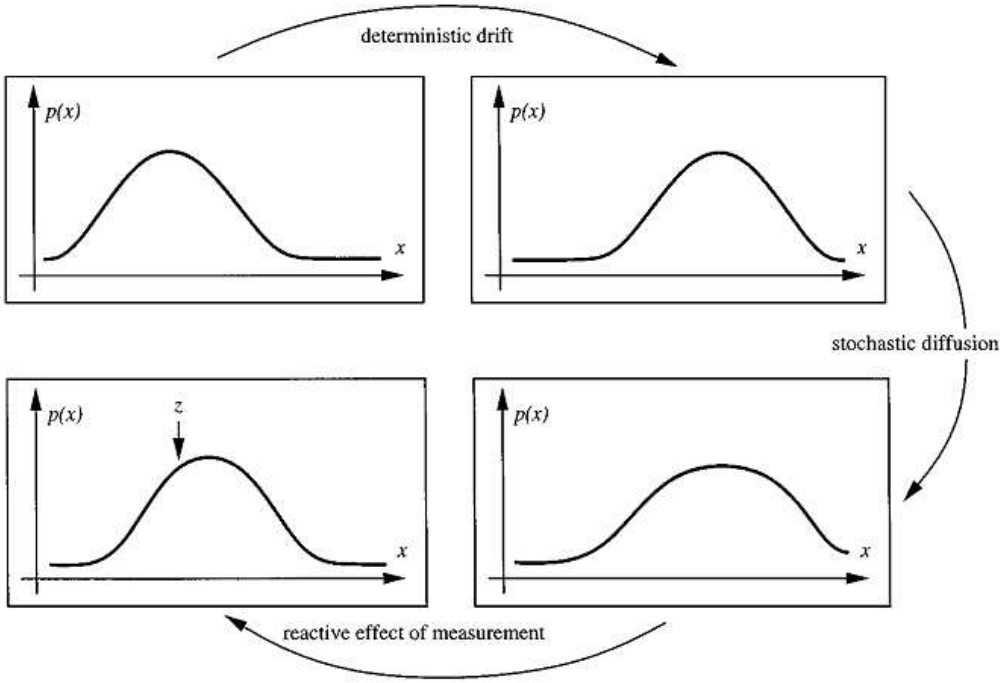


Figure 2.8. Processes in an iteration of Kalman filter. [2]

to the terms of process model $A\mathbf{x}_{t-1}$ and w_t respectively. The reactive effect of measurement corresponds to the correction step. In the correction step, measurement z is considered, and the probability distribution is updated by adding this knowledge. Usually, the measurement is close to the prediction, so the standard deviation decreases, resulting in a narrower gaussian distribution, whose mean is closer to the measured value.

This method is based on a gaussian model that consists of a mean and a standard deviation (or covariance for more dimensions). So, it is only possible to model simple distributions, whereas, the Condensation method can model very complex probability distributions as shown in Figure 2.9.

In our problem, we are trying to track the position of the center of the ellipse. Therefore, our state variable x refers to the (x,y) coordinates of the 2D position and velocity of the object.

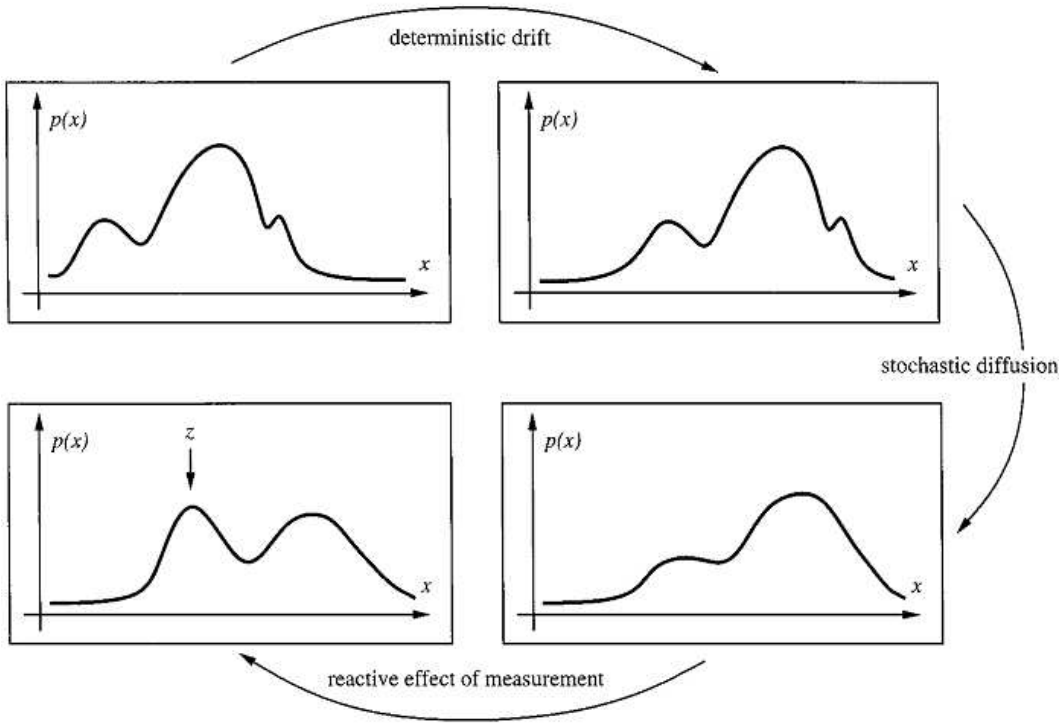


Figure 2.9. Processes in an iteration of Condensation. [2]

In Condensation, process and measurement models are defined by nonlinear functions f_t and h_t:

$$\mathbf{x}_t = f_t(\mathbf{x}_{t-1}, w_t) \tag{2.29}$$

$$\mathbf{z}_t = h_t(\mathbf{x}_t, v_t) \tag{2.30}$$

The measured object state z_t is not an explicit object state given as in Kalman Filter, but it is hidden in the weights given to the samples in the set. We do not have a particular measured object state, but the likelihoods calculated from the current samples.

The terms w_t and v_t are independent and identically-distributed (*iid*) noise sequences as in the Kalman filter. If these equations are compared to equations (2.27)

and (2.28), it is clear that Kalman filter is a special case where f_t and h_t are linear functions with gaussian noise.

The steps in the Condensation method are shown in Figure 2.10. In the beginning, we have an actual sample set. The probability distribution is not modeled by a mean and covariance, but it is simulated by a sample set. We do not have single probability values for every possible object state, but we can use the sample set for approximation. In this sense, Condensation is an example of a general category of non-parametric methods. Such motion estimators that are based on a sample set are also called particle filters. The sample set is denoted as in the following equation:

$$\mathbf{S}_t = \{(\mathbf{x}_t^n, \pi_t^n) : n = 1, 2, ..., N\} \tag{2.31}$$

The sample set includes a weight π_t^n for each sample $\mathbf{x}_t^n$, but initially, all the weights are equal to $1/N$.

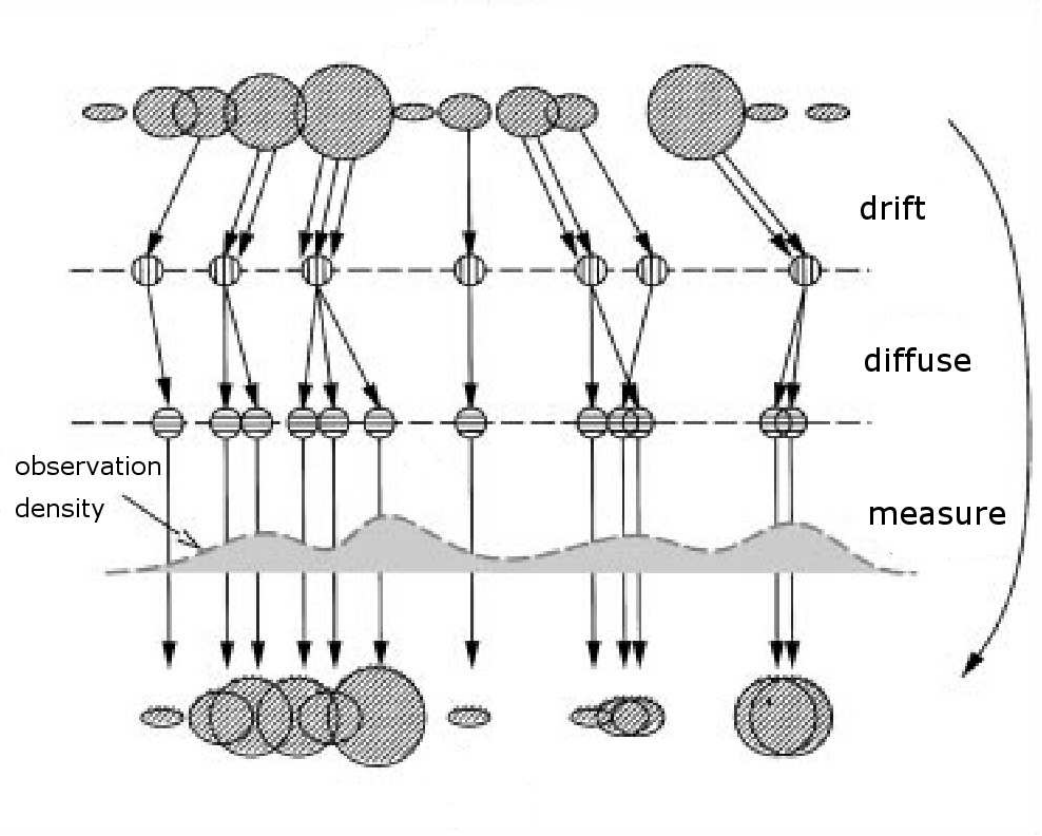


Figure 2.10. Steps of an iteration of Condensation. [2]

The steps of the Condensation algorithm are explained below.

1. *Drift* is a deterministic transformation that we apply to every sample in our set. This is similar to the term $A\mathbf{x}_{t-1}$ in the Kalman process model, except that it is applied to actual object state samples. There is no step for updating an estimated covariance.
2. *Diffuse* is the stochastic step that corresponds to the noise term w_t. A noise term is added to every sample in the sample set. *Drift* and *diffuse* steps together correspond to applying the process function f_t on every sample in the set:

$$\mathbf{x}_{t-1}^n \rightarrow f_t(\mathbf{x}_{t-1}^n) = \mathbf{x}_t'^n \tag{2.32}$$

   The samples $\mathbf{x}_t'^n$ are temporary and they constitute the predicted sample set. They must be weighted to obtain the corrected sample set that contains the samples $\mathbf{x}_t^n$. As the samples are shifted to new positions simulating the effect of the state transition *pdf*, the obtained sample set approximates the predicted *pdf* $p(x_t|z_{1:t-1})$ as calculated in the Chapman-Kolmogorov equation (2.24).
3. *Measure* is the effect of measurement on the distribution. This step differs the most from the Kalman filter. In the Kalman filter, a measurement is a particular measured object state. This measured object state is assumed to be the result of a random gaussian process. In contrast, Condensation measurements are made in actual samples. The sample weights are updated to get posterior probabilities of samples $\mathbf{x}_t^n$ in the condition of the measurement. The posterior probability can be derived from equations (2.23) and (2.25):

$$p(\mathbf{x}_t^n|\mathbf{z}_t) = p(\mathbf{x}_t^n|\mathbf{z}_{1:t}) = k.p(\mathbf{z}_t|\mathbf{x}_t^n).p(\mathbf{x}_t^n|\mathbf{z}_{1:t-1}) \tag{2.33}$$

   where k denotes the constant denominator in (2.25). The last multiplier is the prior probability, the predicted *pdf* that is currently represented by the sample set. To derive the posterior probability, this predicted distribution is to be weighted

by the conditional probability $p(\mathbf{z}_t|\mathbf{x}_t^n)$, in other words, π_t^n.

$$\pi_t^n = p(\mathbf{z}_t|\mathbf{x}_t^n) \tag{2.34}$$

The weight of a sample is calculated by the multiplication of independent probabilities obtained from different types of features. The color histogram, DCT and LBP are assumed to be independent variables. d_i are the distances from the object model obtained from each of the features, calculated by using SSD and Bhattacharyya comparison.

$$\pi_t^n = \prod_{i=1}^{3} 1 - d_i \tag{2.35}$$

In the correction step of the KF, the corrected *pdf* $p(\mathbf{x}_t|\mathbf{z}_t)$ is derived as a whole. But in Condensation, corrected *pdf* is derived by weighting all samples in a set, in which every weight π_t^n corresponds to the Bayesian multiplier $p(\mathbf{z}_t|\mathbf{x}_t^n)$. Instead of correcting the general object state *pdf* from a general measurement, single predicted object state samples are separately measured and weighted. The samples are updated in the *drift* and *diffuse* steps, and the weights are updated in the measure step. Thus, we get a new sample set $\mathbf{S}_t$:

$$\mathbf{S}_{t-1} = \{(\mathbf{x}_{t-1}^n, 1/N) : n = 1, 2, ..., N\} \rightarrow \mathbf{S}_t = \{(\mathbf{x}_t'^n, \pi_t^n) : n = 1, 2, ..., N\} \tag{2.36}$$

4. As a final step before the next iteration, the sample set is updated by these weights. These weights are treated as partial probabilities to select their corresponding samples, and samples are selected one by one from the existing set. Therefore, we get a statistically equivalent sample set, in which every weight is equal to $1/N$.

$$\mathbf{S}_t = \{(\mathbf{x}_t'^n, \pi_t^n) : n = 1, 2, ..., N\} \approx \mathbf{S}_t = \{(\mathbf{x}_t^n, 1/N) : n = 1, 2, ..., N\} \tag{2.37}$$

The obtained sample set includes duplicates of higher weighted samples, but this is not a problem, because they will get separated in the diffuse step of the next iteration.

Instead of updating gaussian parameters, in every iteration, the sample set itself is updated by a a process simulation, then weighted according to measurement probabilities at the estimated position, and finally a new sample set is created by weighted selection. In [2], they used a curve based object model. Equal distance normals are picked on the object curve, and the samples are taken along these one-dimensional normal lines.

Another particle filter was presented in [11] that incorporates the number of objects in the process model itself. This particle filter involves an object model based on color histogram similarity, instead of a curve model. The process model is a composition of discrete and continuous variables, the earlier being the number of objects in the scene and the latter being the positions and velocities of each object. A probability matrix determines probabilities of object birth and death, as well as estimated positions of current objects. The algorithm thus joins detection and tracking in a single method. This joint implementation of detection and tracking minimizes the problem of coalescence, the confusion caused by overlapping objects.

3. THE COLOR AND TEXTURE TRACKER

In the previous chapter, we examined three types of features.

1. Four types of histograms for extracting color statistics: Conventional discrete color histogram (DCH), uniform fuzzy color histogram (UFCH) and the ratio histograms derived from both of them.
2. Local binary patterns for extracting texture features.
3. Discrete cosine transform for extracting object texture.

In this chapter, we define a tracking method that can be programmed to use any subset of these features. As the method is based on object color statistics and texture, it is called the Color and Texture tracker. The tracker uses a Condensation-based particle filter. The motion is modelled as a constant velocity model with a linear transformation plus gaussian noise, similar to the Kalman filter. Selected variances of the gaussian noise are 1.0 for position and 0.3 for velocity components.

The steps of the process are shown in Figure 3.3. The algorithm starts with the initial object rectangle marked by the user. The object window is an ellipse inside a rectangle. Pixels in the ellipse are considered as object pixels, whereas pixels outside the ellipse are considered as background pixels. From the initial window, an object model is formed by its features.

Object features are obtained from three sources:

- Color statistics: YCrCb color histogram is evaluated. For DCH, 4x8x8 bins are allocated for each color component respectively. If UFCH is used, there are 5x9x9 cluster centers. Less resolution is given to the Y component, because chroma is more important than light intensity. Optionally, the ratio histogram can be calculated by dividing elements of inside/object histogram by outside/background histogram. This may allow to further discriminate the object from its background.

- Object texture by DCT: The object is resampled to 16x16, outside pixels are filled with an average color, and the lower components of its DCT are taken as the object appearance model.
- Object texture by LBP: The object is converted to greyscale and 8-point LBP is calculated. Joint LBP-VAR histogram with 10x16 bins is obtained.

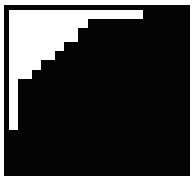


Figure 3.1. The filter for selecting lower frequency components of DCT.

As the tracking progresses, this initial object model is to be altered to adapt to changing object appearance.

After the initial selection of the object window and forming of the object model, a new position for the object is to be determined on the next frame. For the new frame, firstly a sample set of possible new center positions and velocities are generated by condensation motion estimation algorithm as in [2]. Sample velocities are initialized with a zero mean gaussian distribution with variance 10. Samples are translated to positions around the user-selected object center, as if they were at the center in the previous frame and scattered with their current velocities in one frame's time.

After generating the sample set, for each sample, color statistics and texture measurements are calculated, by color histogram, DCT and LBP. Optionally, some of these features may not be used. In this case, the measurements for an unused feature are assumed to yield the best match. Every sample in the sample set is a possible movement of the object window. Similarity of a sample to the measurements defined by the current object model defines the likelihood of the object to have moved to the object window defined by that sample, and is referred to as the weight or confidence value.

Object features are compared to the object model by the following methods:



Figure 3.2. Example tracking results for motion estimator that runs on color statistics and texture measurements, using Condensation with constant velocity.

- Color statistics: Comparison of color histograms is made by calculating the Bhattacharyya coefficient. The coefficient gives the similarity of two histograms in the interval (0,1]. As a geometric interpretation, the output is the cosine of the angle between two histograms in N dimensional space, where N is the histogram dimension, in this case 4x8x8.
- Object texture by DCT: DCT matrices are compared by calculating sum of squared differences (SSD).
- Object texture by LBP: LBP-VAR histograms are compared by Bhattacharyya coefficient, similar to the color histograms.

The probabilities for each sample are obtained by comparing object features at that sample to that of the current object model. Then, the probabilities derived from color, DCT and LBP are combined in a single confidence value, via multiplication of independent probabilities. The generated sample set is then weighted by the confidence values, obtaining a new sample set. The object window is shifted to the centroid of this new sample set, which is considered as the tracked position of the object. At this new position, a new object model is calculated for all three features, and the current object model is updated by blending it with the new model with 30% proportion. The

intent is not to forget previous object features while adapting to changing appearance.

To sum up, the steps of the process for each frame are:

1. A new sample set is generated.
2. Weights are calculated by comparing every sample to current object model.
3. The sample set is updated by these weights,
4. Object window is shifted to the center of the new sample set.
5. Finally, the object model is updated by the new object window.

Positions are measured and corrected for every frame until the scene ends. A tracking example is given in Figure 3.2. In this example, motion estimator with color and texture is used. Condensation algorithm uses constant velocity model. The rectangle and the size of the ellipse is manually determined in the first frame. As observed in different frames, the algorithm is able to track the head even when there are in-depth rotations and complex backgrounds.

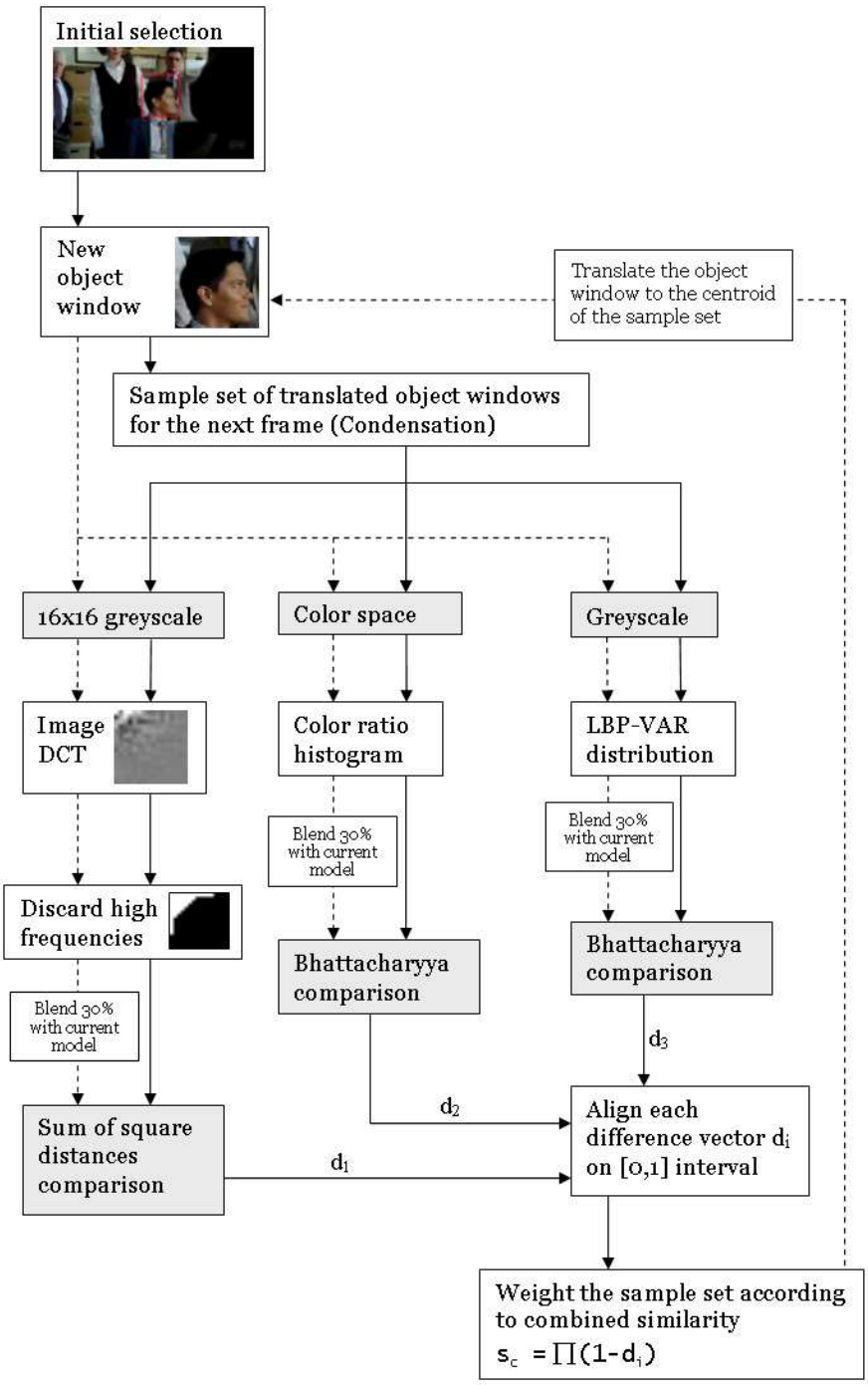


Figure 3.3. The processes in our tracking method.

4. INTERACTIVE CONTENT CREATOR

Interactive Content Creator (ICC) (Figure 4.1) is an application used for embedding interactivity information into video files. If one is watching a video with embedded interactivity information, he can do a number of things. For instance, if the user moves the cursor over objects inside the video, tooltips appear. The user can click on video objects to load web pages about them. In an advertisement of a product, you can click to buy the product. In a music channel, the listeners can click an area on their screens to select the next music video.

In ICC, the user loads a video and defines interactive actions that are connected to events. These events are of three types. Either the action happens spontaneously, it is triggered by a mouse click on a region, or it is triggered when the mouse is over a region. These regions are defined as sets of rectangles on the video frames. These rectangles are created by the user. As the object moves, the rectangle (or multiple rectangles) of the object should also move. Every rectangle has a history of changes through the movie frames. In a frame, a rectangle can appear or dissappear on a frame at some coordinates, and it can move to some other coordinates.

4.1. Object Tracking Issues in ICC

In the original version, all these changes in rectangle coordinates are marked on the video by the user manually, depending on the object movements inside the video. Currently, color and texture-based object tracking method is implemented in ICC. This method is used to improve the efficiency of this program.

Firstly, the scene transitions are detected while loading the video. Scene transitions are frames where object tracking starts and ends. The program automatically determines the frames where the scene changes instantaneously. Detecting more complex transitions such as fade in, fade out, dissolving and wiping is an unsolved issue.

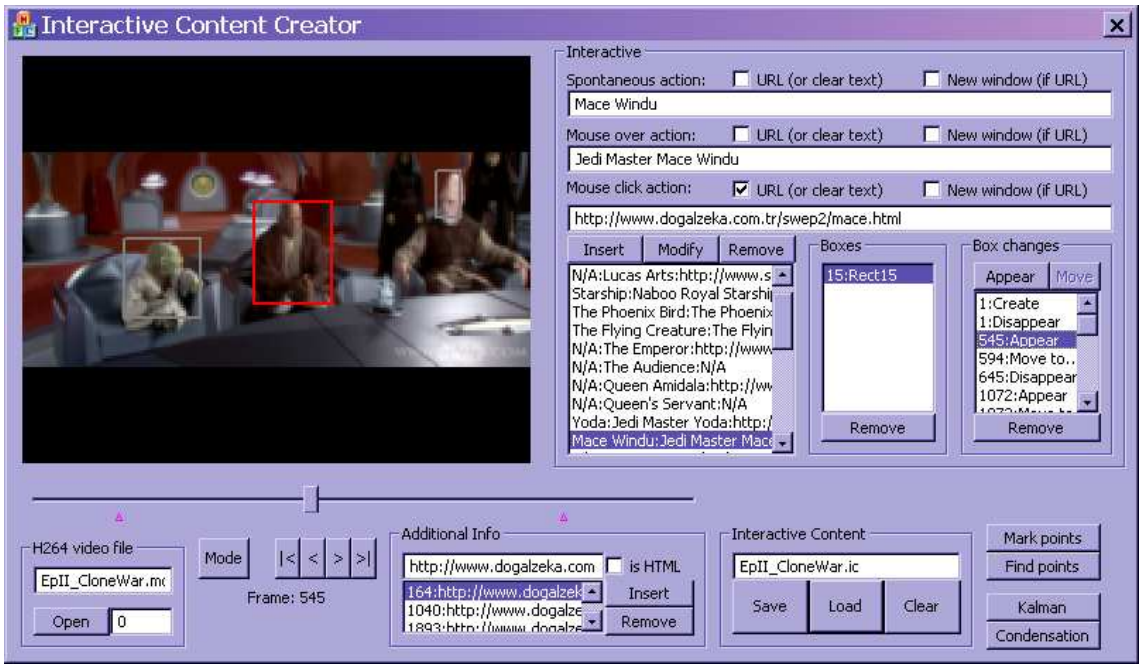


Figure 4.1. Interactivity Content Creator is an application used for embedding interactivity information into video files. [28]

- At a scene beginning, the user marks the object. Then the program follows the object throughout the scene. However, there may be in-scene partial or full occlusions. Unseen objects can also be partly or temporarily tracked. An object appearance model is necessary for this task.
- Usually, there are several objects in a scene. These objects may overlap. The algorithm must have a multiple object mechanism to avoid confusing similar overlapping objects with each other.
- Some objects may not appear at the beginning, but in the middle of the scene. The frames these objects appear can be detected, to request the user to manually mark a rectangle around the new object at these particular frames. This task requires a former model of the object. Maybe the algorithm may at least detect objects already known from preceding scenes.

5. EXPERIMENTS

Two videos from actual movies were used in the experiments. For each video, a first set of four experiments was conducted to compare tracking results of the four types of color histograms:

1. Conventional discrete color histogram (DCH)
2. Uniform fuzzy color histogram (UFCH)
3. Ratio histogram by using DCH
4. Ratio histogram by using UFCH

The purpose of the first stage is to compare the performances of these histograms under different conditions and to find the most suitable histogram calculation method for this video to use in the following experiments. Then, the second set of experiments is conducted to compare the performance of color histogram with other types of features, namely DCT and LBP:

1. Color Histogram
2. Discrete Cosine Transform
3. Local Binary Patterns

Color histogram method that is used in these experiments is the one that gave best results in the first stage. In this second stage, the compared features extract inherently different types of information from the videos. Therefore we expect most varying results. Our purpose is to find which types of features are better object trackers in the given conditions. In the last stage, color histogram is combined with the other two types of features to see if the tracking performance improves:

1. Color Histogram + DCT
2. Color Histogram + LBP
3. Color Histogram + DCT + LBP

The features are joined by simply multiplying their likelihood values.

In the videos, ground truth images are manually created, by defining object region as the head of the person. Ground truth images are only marked once every 5 frames, for example in frames 3,8,13,18,... etc. At these ground truth-enabled frames, a tracking score is calculated by comparing the object window of the tracking method to the true object pixels. The object window is an ellipse inside a rectangle. To score a tracking method at a certain frame, the number of true object pixels inside the ellipse is added to the number of non-object pixels outside the ellipse, and the sum is divided by the total number of pixels in the rectangle. This gives a number in the interval [0,1]. The nearer it is to 1, the better. Example scores at frame 48 for Video 1 are shown in Figure 5.1.



(a) DCT (b) Color (c) Combined (d) Ground truth

Figure 5.1. An example frame of Video 1 in three methods using DCT, color statistics and combination of these two. Scores are 0.77, 0.83 and 0.9 respectively, and calculated by comparing pixels inside and outside the ellipse to the ground truth image (d).

For each video, an initial object window is selected once, and every experiment started from exactly the same initial window. Scores are calculated at five frame intervals for each experiment. Typically, the tracking score starts near 0.9, the score is never exactly 1.0, since the object shape is not a perfect ellipse. A sudden decrease in the score means that the method is failing and losing the object, whereas an increase means that the method is catching-up on the object. In addition, a fourth group of experiments were conducted on the first video to compare our particle filter motion estimation to a different method that uses Kalman Filter.

Table 5.1. Single feature results for two videos

	Video 1	Video 2
Color Histogram	0,3809	0,3449
Fuzzy Hist.	0,4142	0,2945
Ratio Histogram	0,2092	0,3852
Fuzzy Ratio Hist.	0,2022	0,3798
DCT	0,3666	0,3688
LBP	0,2913	0,3822

Video 1 is an indoor sequence with 180 frames, in which the actor in the middle turns around, significantly changing both lighting conditions and object appearance. The background is complex, but it is relatively stationary, because there are no sudden movements and the camera motion is slow. Video 2 is an outdoor sequence with 120 frames, sharing all the challenges mentioned with Video 1. In addition, Video 2 has a dynamic background that is both due to constant camera motion and dynamic objects such as the helicopter and a walking person.

The other videos were selected to determine the limitations of our algorithm in additional experiments. Video 3 is the "foreman" video, which is used in many image processing experiments. It is a high resolution video, and the object is making small random movements, but it constantly changes its shape. Video 4 is the coastguard, an example of an object with a different shape. Video 5 is an example tracking for a smaller object, lower resolution.

Overall results of the experiments are listed in Tables 5.1, 5.2 and 5.3. In these tables, the error values are shown. These errors are calculated as the average value of one minus score values during tracking.

5.1. Experiments using Video 1

Example frames from two tracking experiments of Video 1 are shown in Figure 5.2 and 5.3 at the end of the chapter. The first set of experiments showed that the fuzzy

Table 5.2. Combined results for Video 1

	Video 1
FRH	0,2022
FRH+DCT	0,1643
FRH+LBP	0,2559
FRH+DCT+LBP	0,1462

Table 5.3. Combined results for Video 2

	Video 2
FH	0,2945
FH+DCT	0,2055
FH+LBP	0,3562
FH+DCT+LBP	0,2634

ratio histogram performed better than all the other histograms in this sequence. The stationary background allowed color ratios to help tracking, and the fuzzy histogram better adapted the smooth changes in color conditions.

If we look at Figure 5.4, we can see the histograms behaved very differently from each other. From the first frames, we can see the significant advantage of the ratio histograms over ordinary ones. At frame 50, ratio histograms are parallelly following the objects, whereas Fuzzy histogram is losing it. The discrete histogram makes another peak, but falls again. Between frames 50 and 100, the head turns around changing illumination drastically, and the fuzzy ratio histogram (FRH) seems to adapt better to this smooth change (Figure 5.2). After frame 100, the head does not change orientation, and both ratio histograms track the object similarly.

The next stage is shown in Figure 5.5, where the results of FRH is compared to DCT and LBP. Apparently, color is the best feature, and the information extracted in LBP and DCT is not sufficient to track the object. There is no stable texture. For example, as the head turns around in frame 60-70, the texture becomes completely different. In this case, DCT starts a steady fall and loses the object. LBP seems to

follow an inconsistent trajectory around the object, easily being misled.

We can see the results of the last stage in Figure 5.6. Surprisingly, combined results appear to be more stable than either of its components. In the first 50 frames, the triple combination proves to be even better than double combinations. However, between 50-100 FRH+LBP seems to be slightly better, until frame 110 when it suddenly loses the object (Figure 5.3).

In the fourth set of experiments (Figure 5.7), the same features were used in a different tracking approach to compare the particle filter to the Kalman Filter. This method uses exactly the same feature functions, but as there is no particle filter, measurements on particles cannot be combined, a global measured state is necessary. To calculate the measured state in a particular frame, we start at the corrected object window position. This window is recursivly shifted in the direction it becomes similar to the object model. This corresponds to the gradient descent direction in a different space, a space defined by the comparison function between the object window and the object model. Thus, the object window is shifted to a local maximum in this comparison space. This position that gives a local maximum of similarity to the object model is selected as the measured object state, and it is used in the correction phase of the Kalman Filter. The results show that the particle filter is more successful than Kalman Filter in following the object.

5.2. Experiments using Video 2

Example frames from Video 2 are shown in Figure 5.8. In this video, the tracked person is walking, so the object is quickly moving up and down, resulting in the fluctuating behaviour of the score. Results from the first stage are shown in Figure 5.9. Unlike the first video, the ratio histograms are unsuccessful, due to the movements in the background. As ratio histogram represents the object relative to its background, an unexpected change in the background distorts the object representation. While ratio histograms lose the object, the fuzzy and disrete histograms cannot track the object closely, because of its fast oscillating movement. As they only track the color and do

not consider the appearance of the face, they shift down to the person's neck, which has similar colors. Even then, the fuzzy histogram seems to track for a longer time, due to its increased resolution to represent colors.

Fuzzy histogram is used in the second stage. As seen on Figure 5.10, the color histogram is a better tracker than DCT and LBP on their own. They both lose the object in a short period of time. This is probably because the face is continuously changing direction, and the texture modelled by DCT and LBP fails to adapt to this change. Color statistics is a more stable feature.

In the last stage, we are testing different combinations of fuzzy color histogram with DCT and LBP. Color+DCT seems to be the most stable tracker, following the object until frame 400 (Figure 5.8). Color+LBP wanders around the object, but cannot closely track it. This shows that the DCT of this object is a better feature than LBP for this video. Color+DCT+LBP is secondary, relatively close to the object but unstable due to the effect of LBP.

5.3. Additional experiments

The other three videos were used in these experiments to see the effectiveness of tracking in different types of objects. Video 1 demonstrated rotation in depth, changing object shape. Video 2 demonstrated a fast object on a complex background. Video 3 (Figure 5.12) also demonstrates rotation in depth in a larger object, with more resolution. In Video 4 (Figure 5.14), the selected object does not have an elliptical shape, but there is a stationary background. In Video 5 (Figure 5.14), the object window is selected to be smaller, requiring the tracking algorithm to extract the features from less information, and less resolution. It is a shorter segment with 50 frames. A single tracking method was used for the last three videos: Fuzzy Histogram with DCT.

In the experiment with Video 3 (Figure 5.13), we see that the method tracks the object, but slowly loses it due to sudden rotations and changes in lighting conditions. In Video 4 (Figure 5.15), due to the different object shape, the maximum score is

0,75. However, it is able to track the object, because of the stationary background. In Video 5 (Figure 5.16), due to the smaller object window, the effect of object speed is amplified, resulting in a fluctuating behaviour. The tracker loses the object near to the end.



(a) Frame 2

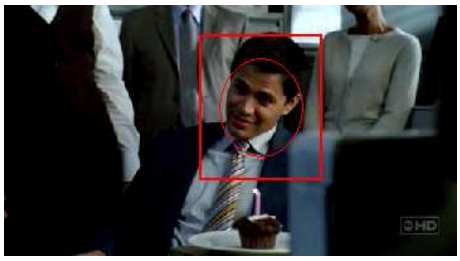
(b) Frame 28



(c) Frame 52



(d) Frame 90



(e) Frame 103



(f) Frame 118



(g) Frame 140



(h) Frame 158



(i) Frame 163

Figure 5.2. Example frames from tracking with Fuzzy Ratio Histogram in Video 1.



(a) Frame 2



(b) Frame 28



(c) Frame 52



(d) Frame 90



(e) Frame 103



(f) Frame 118



(g) Frame 140



(h) Frame 158



(i) Frame 163

Figure 5.3. Example frames from tracking with Fuzzy Ratio Histogram combined with DCT and LBP in Video 1.

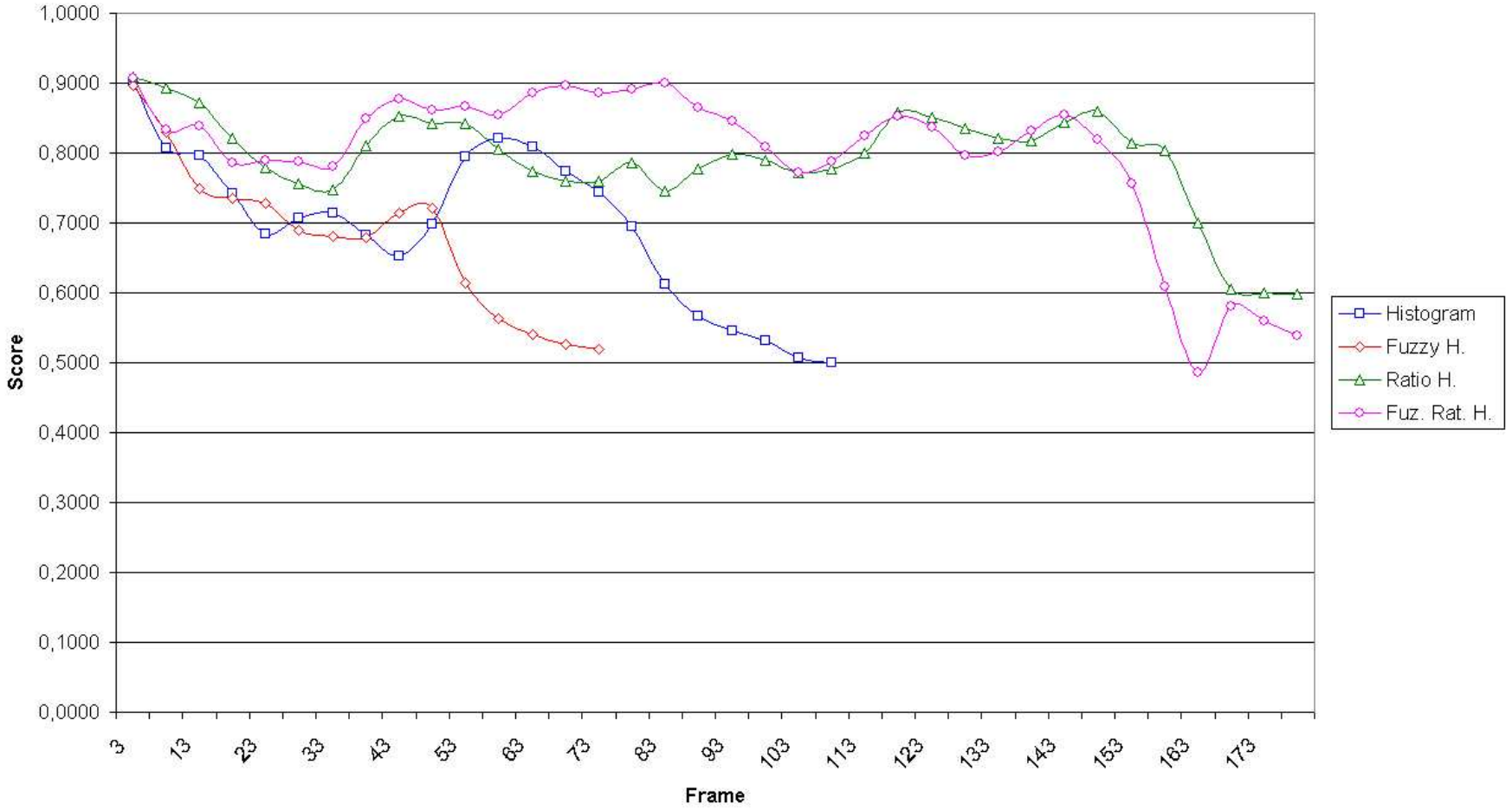


Figure 5.4. Different histogram algorithms' performances on Video 1.

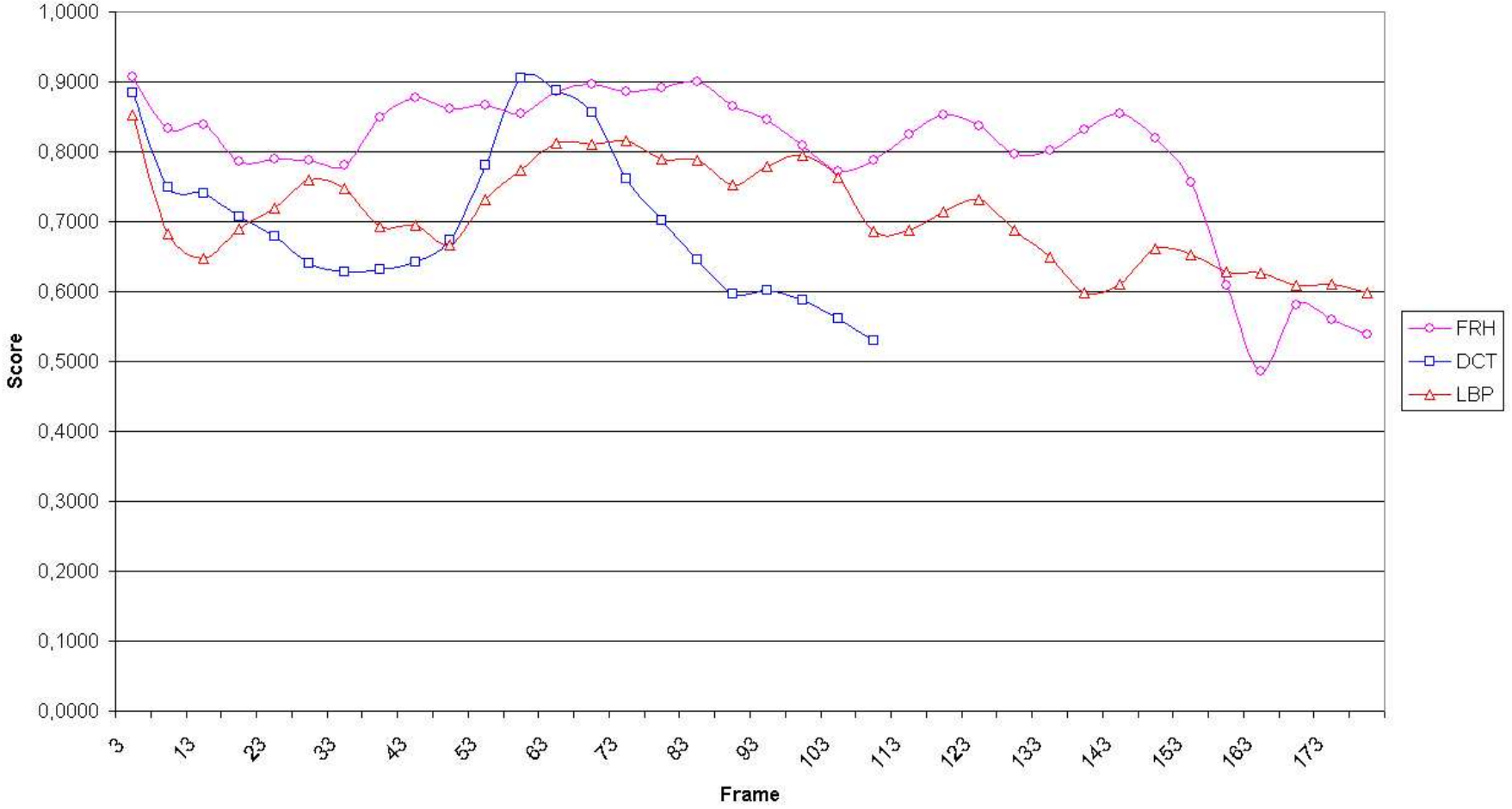


Figure 5.5. Fuzzy Ratio Histogram, LBP and DCT separate performances in Video 1.

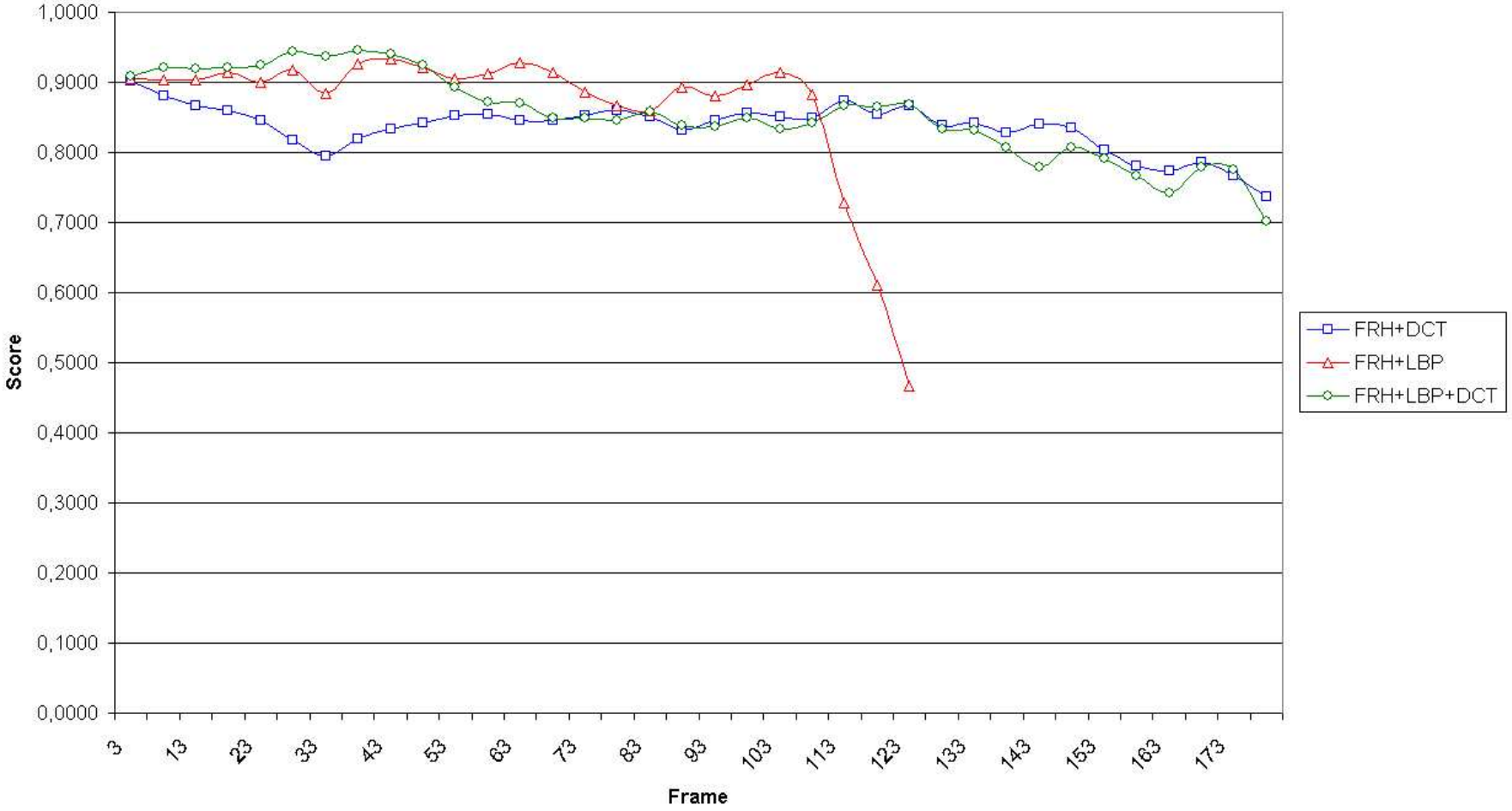


Figure 5.6. Combined performance of Fuzzy Ratio Histogram with LBP and DCT in Video 1.

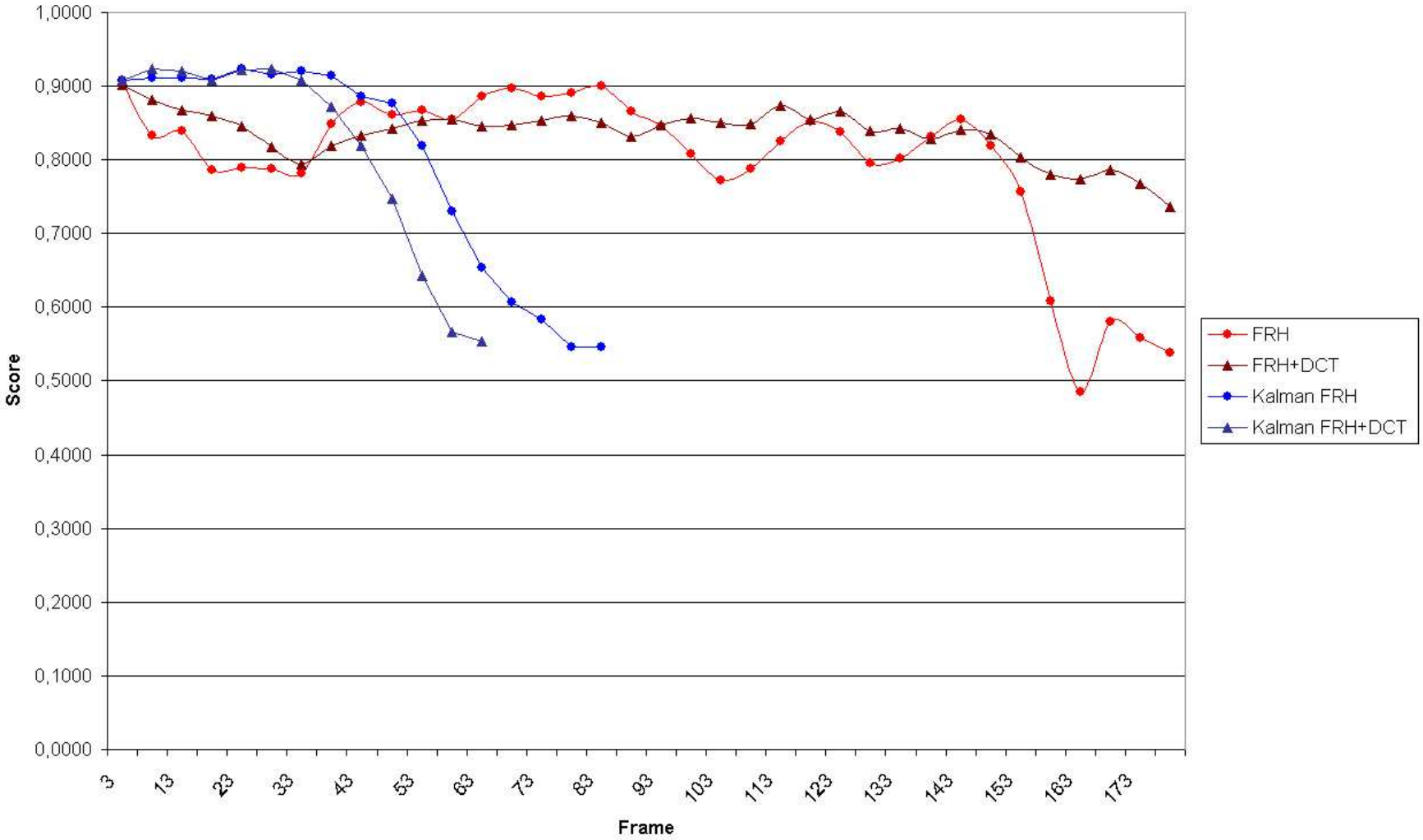


Figure 5.7. Comparison of our particle filter motion estimation with a Kalman Filter-based approach on Video 1.



(a) Frame 263

(b) Frame 278



(c) Frame 298



(d) Frame 263



(e) Frame 313



(f) Frame 338



(g) Frame 368



(h) Frame 388



(i) Frame 403



(j) Frame 418

Figure 5.8. Example frames from tracking with Fuzzy Histogram combined with DCT in Video 2.

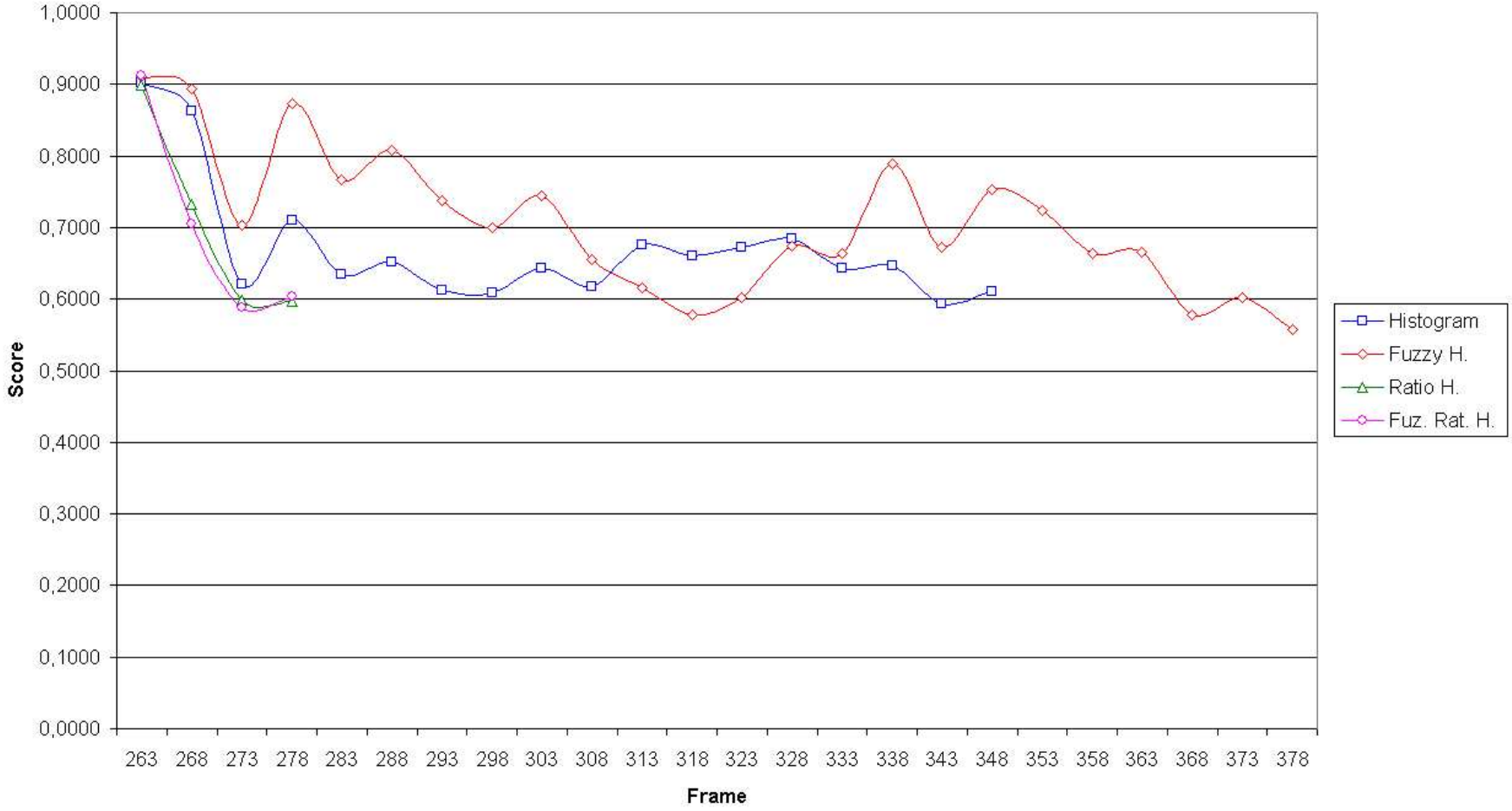


Figure 5.9. Different histogram algorithms' performances on Video 2.

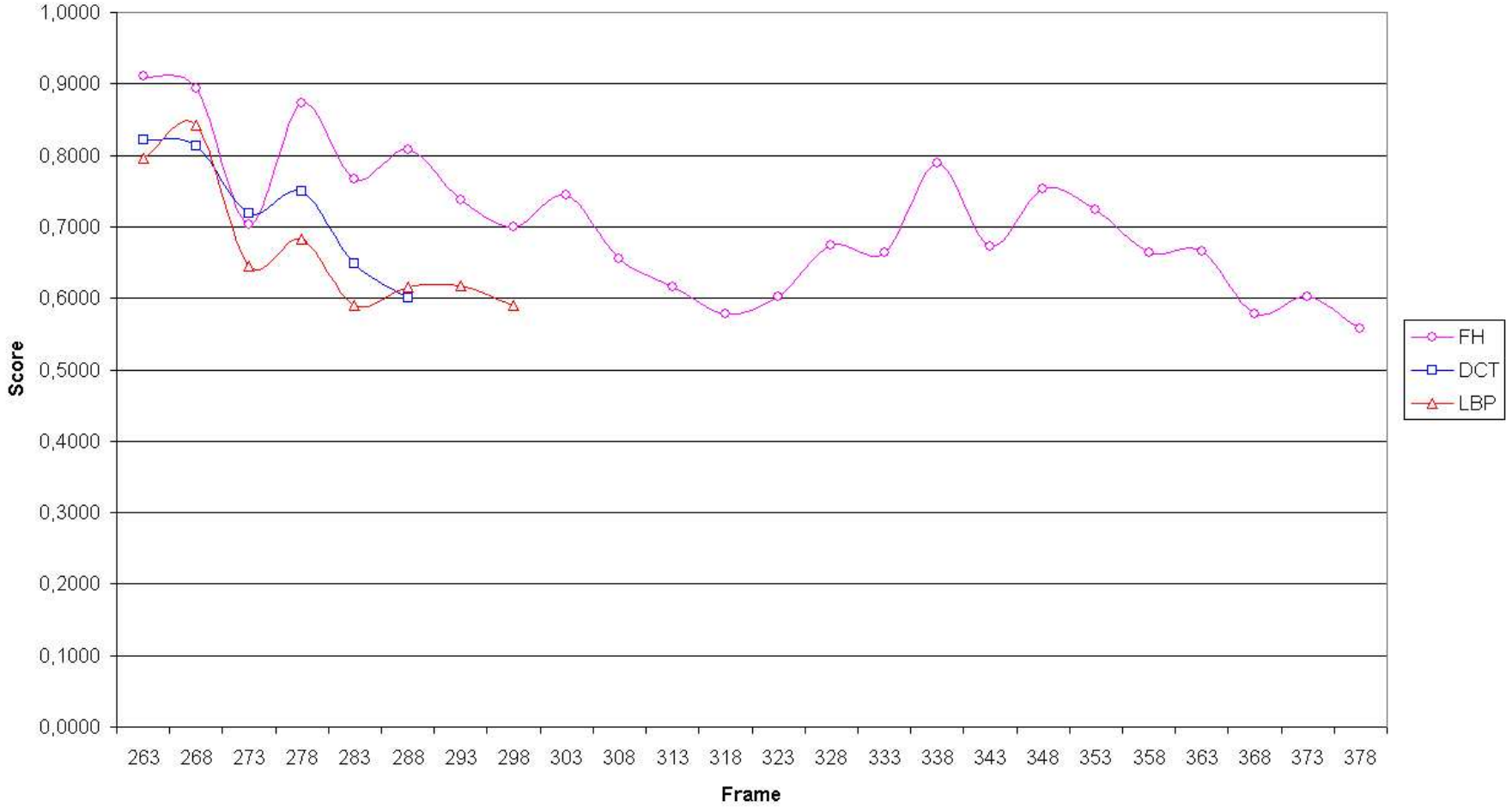


Figure 5.10. Fuzzy Histogram, LBP and DCT separate performances in Video 2.

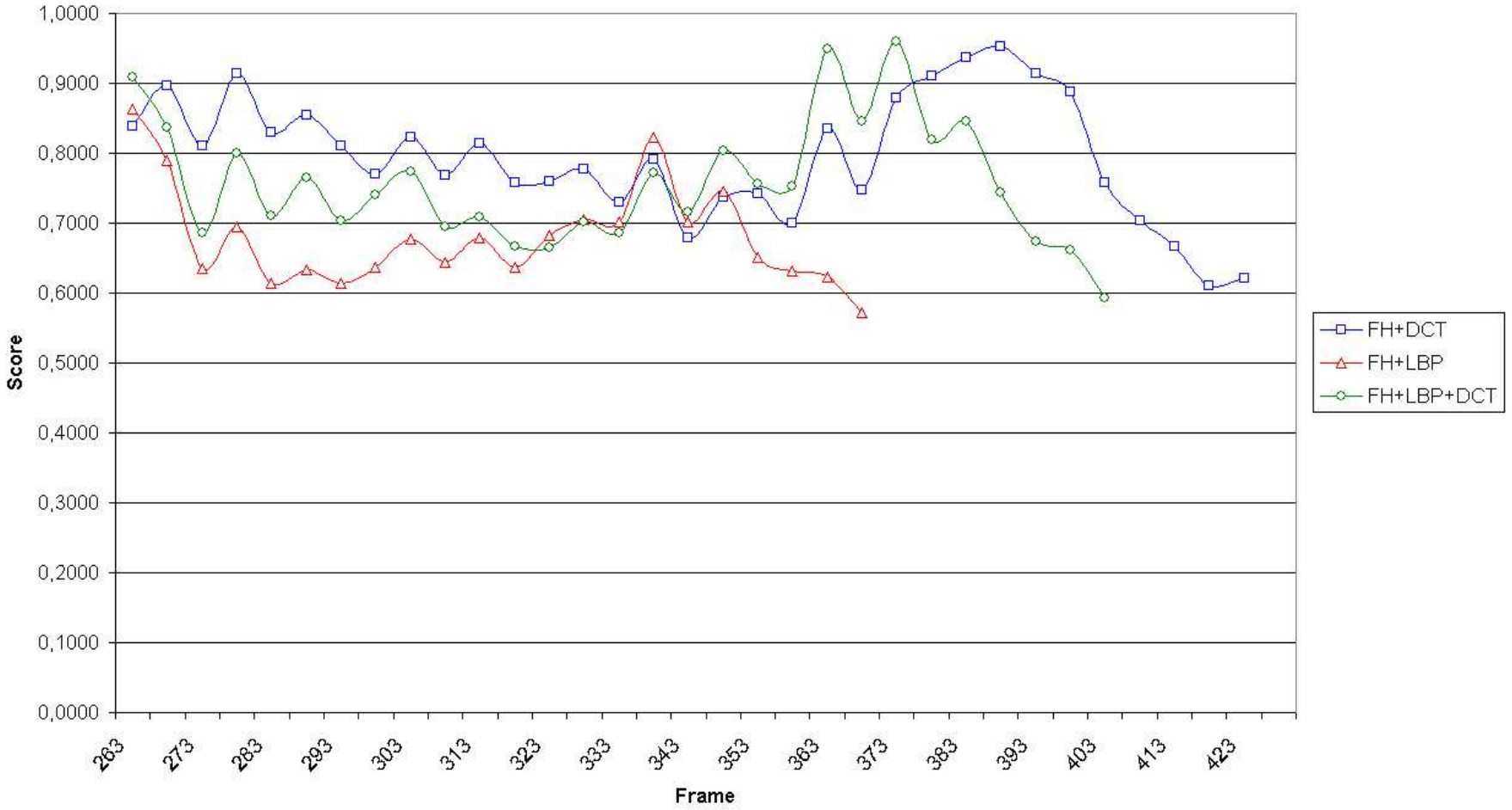
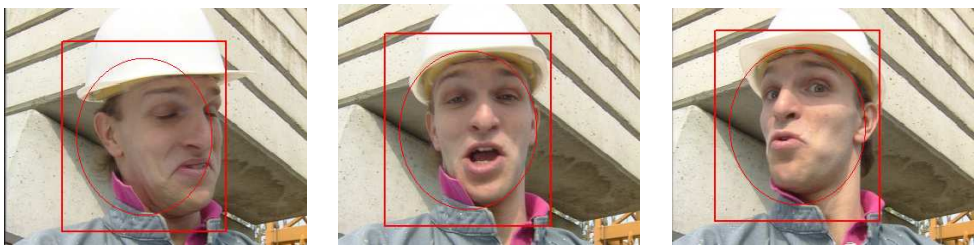


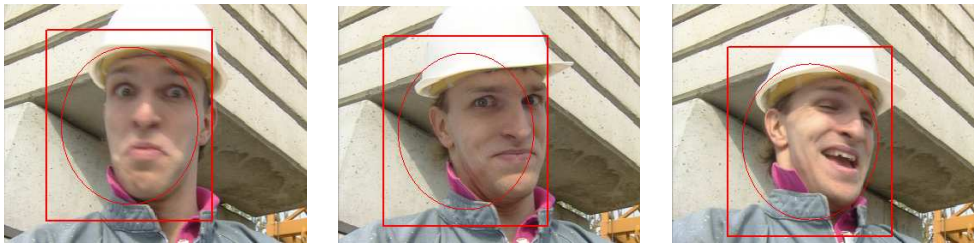
Figure 5.11. Combined performance of Fuzzy Histogram with LBP and DCT in Video 2.



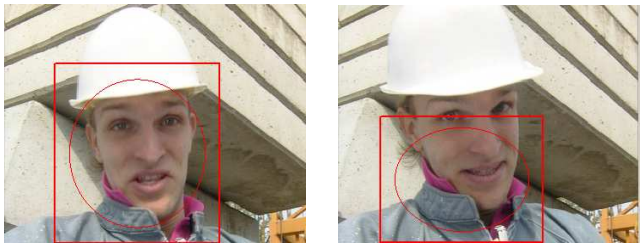
(a) Frame 21 (b) Frame 37 (c) Frame 67



(d) Frame 74 (e) Frame 82 (f) Frame 88



(g) Frame 93 (h) Frame 101 (i) Frame 138



(j) Frame 144 (k) Frame 150

Figure 5.12. Example frames from Fuzzy Histogram + DCT tracking of Video 3 (Foreman).

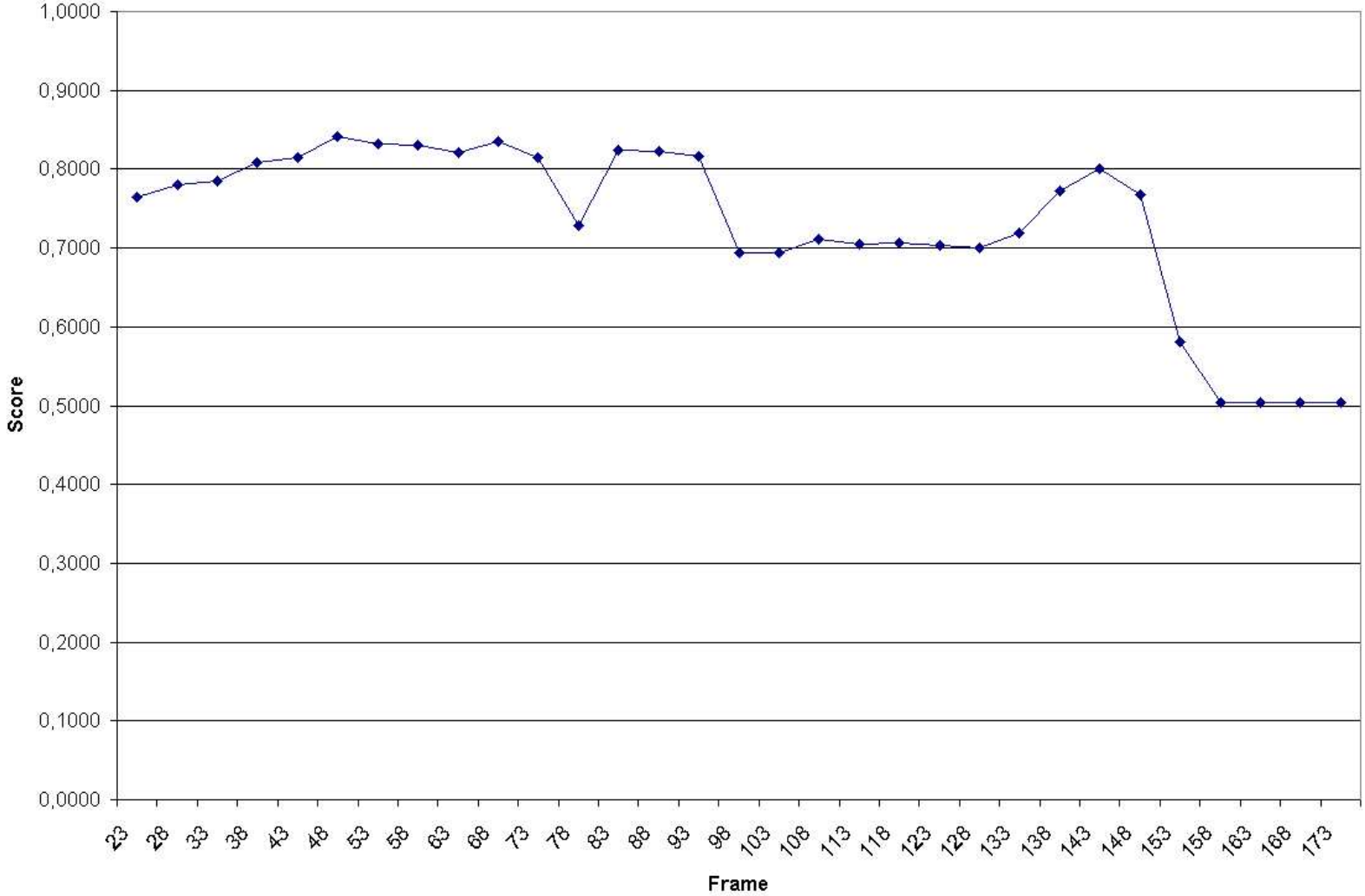
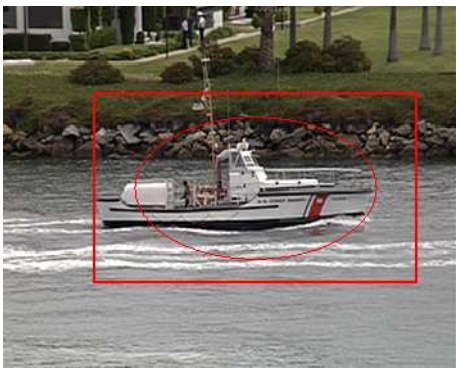


Figure 5.13. Fuzzy Histogram with DCT tracking in Video 3 (Foreman).



(a) Frame 98



(b) Frame 151



(c) Frame 226



(d) Frame 697



(e) Frame 723



(f) Frame 744

Figure 5.14. Example frames from Fuzzy Histogram + DCT tracking of Video 4 and 5.

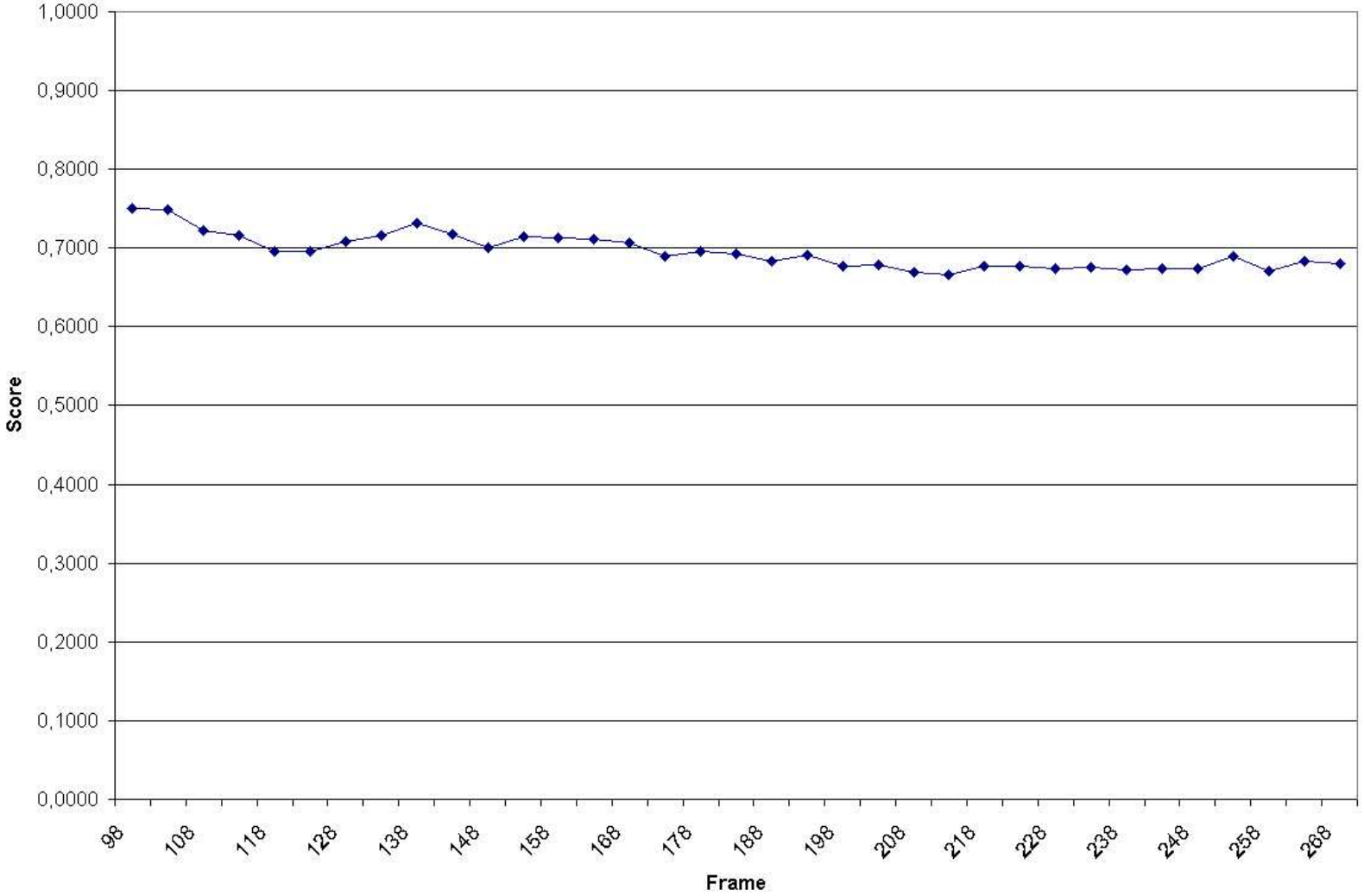


Figure 5.15. Fuzzy Histogram with DCT tracking in Video 4 (Coastguard).

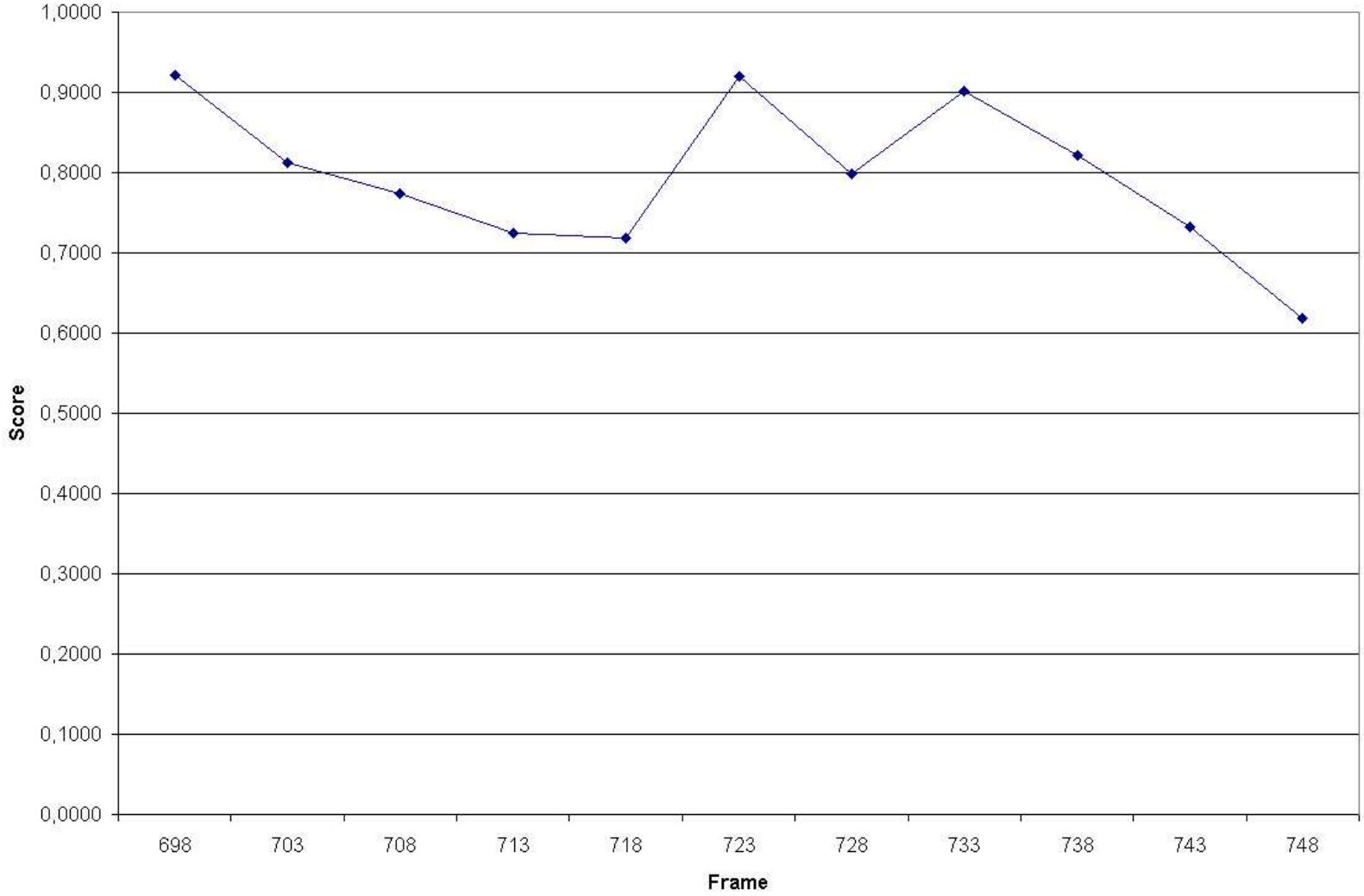


Figure 5.16. Fuzzy Histogram with DCT tracking in Video 5.

6. CONCLUSION

In this study, an object tracking method is developed for an application that enables users to embed interactivity to videos. The method is a general object tracker based on a translating ellipse-shaped object window, whose motion is estimated by condensation particle filter. The improvement introduced by the method is the joining of three different types of object features, namely, YCrCb color statistics, DCT and LBP representation of the object texture statistics. Due to different representations used, each performs best under different circumstances.

Uniform Fuzzy Color Histogram (UFCH) is a novel method for calculating color statistics presented in this thesis. The histogram comparison experiments show that it represents color features better than a conventional discrete histogram. Feature comparison experiments show that color histogram is a more stable and representative feature for use with object tracking than texture (DCT, LBP), if each feature is used alone.

The combination experiments are conducted to find out if joining color with texture features improved its representative power. The experiments show that it is possible to obtain a more stable tracker by combining color with texture. However, this kind of combined implementation may amplify weaknesses of these features as well as strengths. In the combined tracker, three features are directly joined by multiplication rule for independent probabilities. Thus, all have the same significance and same weight on the output. There must be a mechanism that evaluates the efficiency of every feature and decides which features are more desirable for the moment. If feature changes can be considered as a motion in the feature space, there could be an appearance model similar to motion estimation, that does not predict object movement, but changes in features. A dynamical feature estimation mechanism would help to reduce the temporary negative effect of weak features and increase overall efficiency.

Unsupervised learning algorithms can be implemented to improve the tracking by

determining best combination of different features for a given input. Principal Component Analysis (PCA) is an algorithm to find a linear transformation that represents the given samples in least number of components. PCA can be used to transform the feature space and use only the meaningful feature components. Vector Quantization (VQ) is another method that can represent nonlinear functions in a feature space, but with a limited resolution. In [29], another method, Non-negative Matrix Factorization (NMF) is demonstrated. Contrary to PCA and VQ that learn holistically, NMF is a factorization that focuses on the parts of the object features. These unsupervised learning algorithms can be used to learn the types of features that best discriminate the object from its background.

REFERENCES

1. T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, 2002.

2. M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

3. M. Isard and J. Maccormick, "Bramble: a bayesian multiple-blob tracker," in *Proceedings of the IEEE Computer Vision, ICCV '01.*, vol. 2, 2001, pp. 34–41 vol.2.

4. A. B. A. S.-V. J. Melo, J.; Naftel, "Detection and classification of highway lanes using vehicle motion trajectories," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 7, no. 2, pp. 188–200, June 2006.

5. V. Takala and M. Pietikainen, "Multi-object tracking using color, texture and motion," in *Proceedings of the IEEE Computer Vision and Pattern Recognition, CVPR '07*, June 2007, pp. 1–7.

6. C. Tomasi and T. Kanade, "Shape and motion from image streams: A factorization method," CMU, Tech. Rep. CMU-CS-91-132, April 1991. [Online]. Available: ftp://reports.adm.cs.cmu.edu/usr/anon/1991/CMU-CS-91-132.ps

7. A. Jepson, D. Fleet, and T. El-Maraghi, "Robust online appearance models for visual tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 10, pp. 1296–1311, Oct. 2003.

8. F. Dornaika and F. Davoine, "On appearance based face and facial action tracking," vol. 16, no. 9, pp. 1107–1124, September 2006.

9. W. Qu and D. Schonfeld, "Real-time decentralized articulated motion analysis and object tracking from videos," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2129–2138, 2007.

10. R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

11. J. Czyz, B. Ristic, and B. Macq, "A particle filter for joint detection and tracking of color objects," *Image Vision Comput.*, vol. 25, no. 8, pp. 1271–1281, 2007.

12. J. Han and K.-K. Ma, "Fuzzy color histogram and its use in color image retrieval," *Image Processing, IEEE Transactions on*, vol. 11, no. 8, pp. 944–952, Aug 2002.

13. I. El-Feghi, H. Aboasha, M. Sid-Ahmed, and M. Ahmadi, "Content-based image retrieval based on efficient fuzzy color signature," in *Proceedings of the IEEE Systems, Man and Cybernetics, ISIC '07.*, Oct. 2007, pp. 1118–1124.

14. A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, no. 35, pp. 99–110, 1943.

15. D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–575, 2003.

16. M. Polat, E.; Ozden, "A nonparametric adaptive tracking algorithm based on multiple feature distributions," *Multimedia, IEEE Transactions on*, vol. 8, no. 6, pp. 1156–1163, Dec. 2006.

17. M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vision*, vol. 7, no. 1, pp. 11–32, November 1991. [Online]. Available: http://portal.acm.org/citation.cfm?id=134841

18. L. Dong and S. Schwartz, "Dct-based object tracking in compressed video," in

Proceedings of the IEEE Acoustics, Speech and Signal Processing, ICASSP '06., vol. 2, 14-19 May 2006, pp. II–II.

19. C. He, Y. Zheng, and S. Ahalt, "Object tracking using the gabor wavelet transform and the golden section algorithm," *Multimedia, IEEE Transactions on*, vol. 4, no. 4, pp. 528–538, Dec 2002.

20. C. Jianbo Shi; Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Vision and Pattern Recognition, CVPR '94*, 1994, pp. 593–600.

21. E. Loutas, K. Diamantaras, and I. Pitas, "Occlusion resistant object tracking," in *Proceedings of the International Conference on Image Processing, ICIP '01.*, vol. 2, 7-10 Oct 2001, pp. 65–68 vol.2.

22. M. Krinidis, N. Nikolaidis, and I. Pitas, "2-d feature-point selection and tracking using 3-d physics-based deformable surfaces," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, no. 7, pp. 876–888, July 2007.

23. B. Chitprasert and K. Rao, "Human visual weighted progressive image transmission," *Communications, IEEE Transactions on*, vol. 38, no. 7, pp. 1040–1044, Jul 1990.

24. Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, 1995.

25. D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, May 2002.

26. S. Maskell and N. Gordon, "A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking," *Target Tracking: Algorithms and Applications (Ref. No. 2001/174), IEE*, vol. Workshop, pp. 2/1–2/15 vol.2, Oct. 2001.

27. A. Nguyen, H.T.; Smeulders, "Fast occluded object tracking by a robust appear-

ance filter," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 8, pp. 1099–1104, Aug. 2004.

28. DoğalZeka. [Online]. Available: http://www.dogalzeka.com.tr

29. D. D. Lee and S. H. Seung, "Learning the parts of objects by nonnegative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.