TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

RANJEETH SHETTY

INTELLIGENT SUMMARIZATION OF SPORTS VIDEOS USING AUTOMATIC SALIENCY DETECTION

Master of Science thesis

Examiners:
Prof. Moncef Gabbouj
Dr. Iftikhar Ahmad
Examiner and topic approved by the Faculty Council of the Faculty of Computing and Electrical Engineering on 14th January 2015.

ABSTRACT

RANJEETH SHETTY: Intelligent Summarization of Sports Videos Using Automatic Saliency Detection.
Tampere University of technology
Master of Science Thesis, 52 pages
July 2015
Master's Degree Programme in Information Technology
Major: Multimedia

Examiners:

Professor Moncef Gabbouj, Department of Signal Processing, Tampere University of Technology.

Dr. Iftikhar Ahmad, Department of Signal Processing, Tampere University of Technology.

Nokia supervisor:

Dr. Igor Curcio, Principal Scientist, Nokia Technologies Tampere.

Keywords: Sports summarization, Saliency detection, Object detection, Motion detection, Background subtraction, Video remixing

The aim of this thesis is to present an efficient and intelligent way of creating sports summary videos by automatically identifying the highlights or salient events from one or multiple video footage using computer vision techniques and combining them to form a video summary of the game.

The thesis presents a twofold solution

- Identification of salient parts from single or multiple video footage of a certain sports event
- Remixing of video by extracting and merging various segments, with effects (such as slow replay) and mixing audio.

This project involves applying methods of machine learning and computer vision to identify regions of interest in the video frames and detect action areas and scoring attempts. These methods were developed for the sport of basketball. However, the methods may be tweaked or enhanced for other sports such as football, hockey etc.
For creating summary videos, various video processing techniques have been experimented to add certain visual effects to improve the quality of summary videos.
The goal has been to deliver a fully automated, fast and robust system that could work with large high definition video files.

PREFACE

This thesis was done as a part of a project in Nokia Technologies, Tampere, Finland.

In this regard, I am glad to have the pleasure of working with Dr. Igor Curcio from Nokia Technologies and appreciate the support and guidance offered by him. I thank my colleagues in Nokia Technologies in particular Sujeet Mate and Francesco Cricri for all their help.

I am grateful to Prof. Moncef Gabbouj and Dr. Iftikhar Ahmad from the Department of Signal Processing, Tampere University of Technology for their support. I thank my friend and former colleague Muhammad Adeel Waris at Tampere University of Technology for his valuable ideas, suggestions and interesting discussions.

Last but not the least I thank my family for their moral support.

Tampere, 15.7.2015

Ranjeeth Shetty

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ROI	Region of Interest
LBP	Local Binary Patterns
MB-LBP	Multi-Block Local Binary Patterns
RGB	Red, Green and Blue
AdaBoost	Adaptive Boosting
GMM	Gaussian Mixture Model
RDP	Ramer-Douglas-Peucker algorithm
OpenCV	Open source Computer Vision Library
FFmpeg	Fast forward mpeg – an open source media processing library
PTS	Presentation Timestamps
TP	True Positives
FP	False Positives
GT	Ground Truth

1. INTRODUCTION

Video recording of sports games have improved a lot in the past few years. Today most games are often recorded using multiple cameras that provide different view angles of the game. Figure 1 shows an example of camera setup using multiple cameras to record a single game. These views are mixed often adding some effects like slow playback for replay to provide a rich viewing experience for the viewer.

In order to create great media content for the viewers, it is important to understand how well these games are recorded, how multiple camera footages are mixed, manipulation of background audio etc. There is an art in the way these events are recorded and mixed to create an entertaining video for the user. The cameraman must pay attention to the key areas in the field or the playing area and most importantly must not miss any significant event during the game such as a score or a scoring attempt.

The creation of media content is also influenced by where and how it is played or shared. For example a recording of game may be broadcasted on TV or the recording showing only the top highlights of the game may be posted as an Internet media on the web page of the respective sports team.

Users watching recorded videos of sports, most of the times just want to see the important bits or the highlights. Manually creating a media content with the highlights is time consuming and tedious as it requires a person to go through the entire recording to choose the best bits. This work becomes more cumbersome when multiple cameras are involved as this would include tasks of combining different view angles for each highlight event. Thus there is a need for a solution that could automatically and efficiently detect salient events in a game, combine different camera angles of each of these salient events and make it visually appealing to the viewer.

There has been quite a lot of development in the field of computer vision in the past few years. Many techniques of computer vision are now applied in sports for game and player analysis.

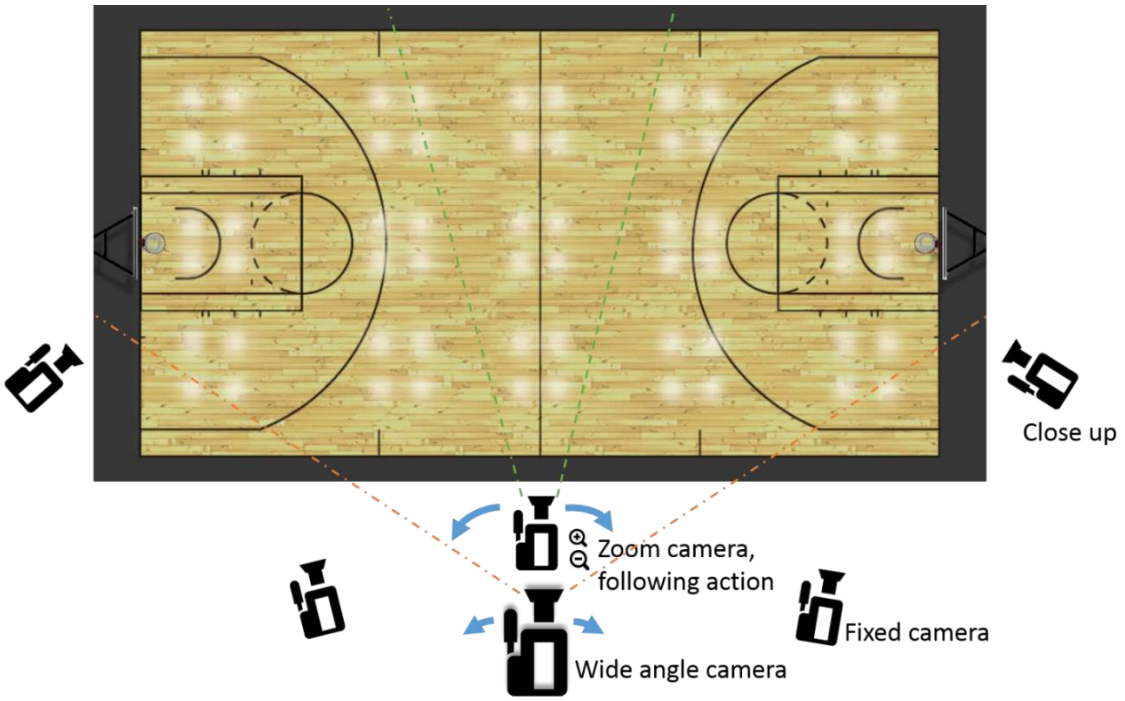

Figure 1. *Illustration of an example setup of various cameras and their coverage in a typical basketball game recording*

1.1. Computer Vision in sports

In the past recent years, there has been a growing need to understand the semantics in sports games. Use of technology in analyzing player movements and understanding the action on a sports field has been growing in the past few years. Most of the systems today make use of certain tracking devices worn by players or certain markers with sensors placed around the play area. These trackers or markers are electronic devices that communicate with the cameras or cameramen.

Other technologies such as the goal line technology popularly used in soccer helps game referees to make accurate decisions that are often misjudged by mere human perception.

The primary challenges in these techniques is to make it cost effective and ease of installation and use. It is not convenient to setup markers and sensors around the playing field or to force players to wear certain recording or communication devices without affecting their natural style of playing. Placing a sensor in the game ball also poses a tricky problem of not altering the physical properties of the ball. Sports recorders and broadcasters are now looking for simple and yet effective solutions to get semantic information from a sports game. The big question here is - Can we get sufficient important data only from a video capture just as a human would without relying on external aids of markers and sensors? With advances in various computer vision algorithms and techniques the goal for the future is to analyze everything from captured video. This kind of solution is obviously more attractive to broadcasting and game recording companies as they don't need

to setup extra equipment, or influence the authorities to change the match ball or players' outfits.

One of the starting points in using cameras to accurately detect a salient event is identifying regions of interest and identifying objects. In the next steps these regions and motion of identified object/s in the regions are to be analysed. In case of sports like soccer or basketball, these objects would be the ball and the goal post or the basket hoop. The region of interest would be the region around the goal post or the basket hoop. Object recognition can be done with supervised machine learning methods using classifiers that are trained with images of these objects.

1.2. Thesis Outline and scope

In this thesis work, the algorithms and techniques have been developed focusing mainly on the sport of basketball. However these techniques may be tweaked and fine-tuned for use in other team sports such as soccer, hockey etc.

One of the goals in this project work has been to figure out ways to determine salient events and salient regions automatically by using computer vision algorithms and techniques. The other goal of this project work was to create a system that uses the previously detected salient timestamps and automatically clip and merge salient segments of video clips from multiple camera feeds and create a summarization video comprising of the most salient events in a typical basketball game.

The rest of the thesis is organized as follows. In the next chapter, the basic principles and algorithms used in the thesis are discussed. Chapter 3 explains the overall framework and gives an overview of various modules of the developed system. In chapter 4 the methods and implementation of the saliency detection module are explained. Chapter 4 explains the implementation of the video editing and remixing module. In chapter 5, the experimental setup and results of testing the methods are presented. Finally in the last chapter, the conclusion from the results are derived and possible future directions are presented.

2. THEORETICAL BACKGROUND

In this chapter, the reader is presented with the technical and theoretical background information which is necessary in order to understand the solutions and methods described later.

2.1. Object detection and recognition

Object detection is the task of finding the position of an object if present in a given image whereas Object recognition is the task of identifying which object is shown in the given image. There are various techniques for object detection and recognition emerging from the field of computer vision and machine learning. A brief introduction to some of these methods are presented in the following sections. Two main approaches to identifying objects in an image are feature extraction techniques and cascade based supervised machine learning. In the following sections two widely used features in object detection and recognition namely Local binary patterns and Haar-like features are explained.

2.1.1. LBP Local Binary patterns

Local Binary Pattern is a feature descriptor operator using which two-dimensional surface textures can be described by two complementary measures: local spatial patterns and grayscale contrast. It is a simple yet efficient operator that works by labelling the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number [1].

Local Binary Patterns (LBP), was first proposed by Ojala et al. [2]. It has been proved to be robust against illumination variations and effective for capturing the underlying textural information of an image [2][3]. Since the development of LBP, its many variants have been proposed in the literature such as Extended-LBP [2][4], Improved [5], MB-LBP [3], CS-LBP [11] etc.

The original LBP operator involves thresholding the 3x3 neighborhood of each pixel with the center value and considering the result as a binary number. The histogram of these 2^8 = 256 different labels are then used as a texture descriptor. This operator used jointly with a simple local contrast measure provided very good performance in unsupervised texture segmentation [6].

The LBP operator was extended to use neighborhoods of different sizes [2]. In the following, the notation (P, R) will be used for pixel neighborhoods which means P sampling points on a circle of radius of R. Figure 2 illustrates an example of circular neighborhoods for LBP with P=8, R=1 and P=8, R=2.

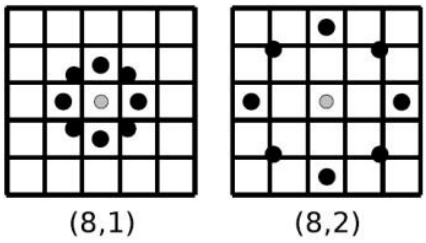

Figure 2. *Circular neighborhoods for LBP with P=8, R=1 and P=8, R=2. The pixel values are bilinearly interpolated whenever the sampling point is not the center of a pixel [2]*

The value of the LBP code of a pixel (x_c,y_c) is given by:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)\, 2^p \tag{2.1}$$

$$s(x) = \begin{cases} 1, if\ x \geq 0\,; \\ 0, otherwise. \end{cases} \tag{2.2}$$

where g_p is the intensity *p*th sampling point and g_c is the intensity of the center point (x_c, y_c).

An example of LBP operation is shown in the Figure 3.

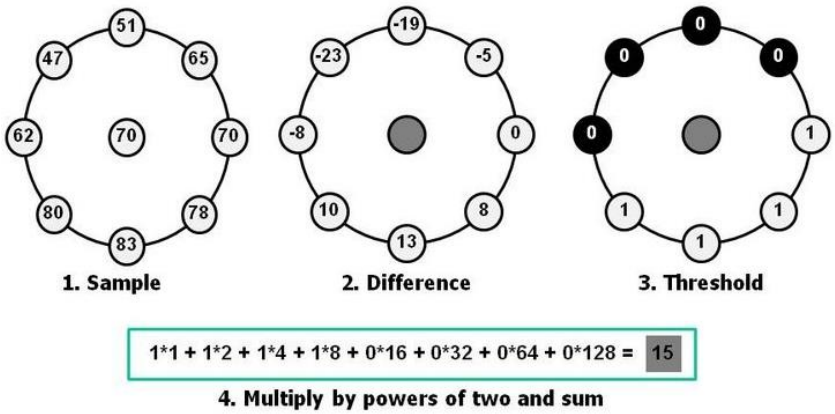

Figure 3. *Example illustrating the LBP operation over 8 neighbours*

MB-LBP (Multi-Block LBP) extends the idea of basic LBP and encodes rectangular regions by comparing the average intensity value of the central rectangle with average intensity values of neighboring rectangles as illustrated in Figure 4.

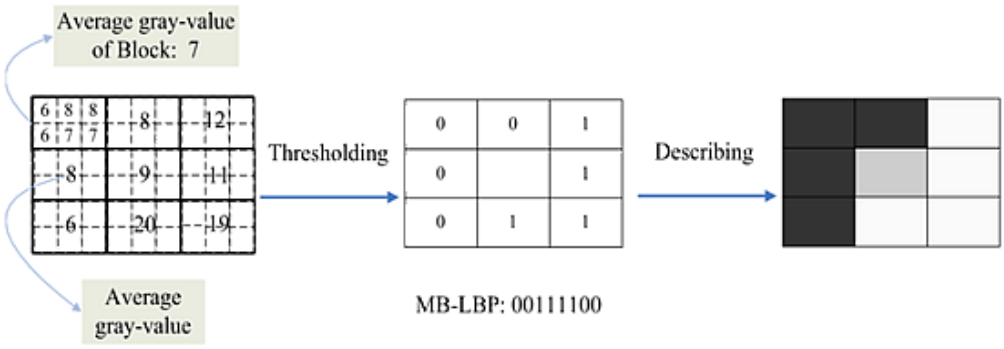

Figure 4. *Illustration of the Multi-Block LBP [7]*

We can get 256 kinds of resulting patterns which can be used as MB-LBP features in later stages of object detection. Diverse image structures such as edges, lines, spots, flat areas and corners can be detected using these binary patterns [7]. The MB-LBP feature set is far smaller compared to Haar-like features (about 1/20th) which is discussed in the next section. This smaller set of feature gives a computational advantage. However, most of these features are redundant and therefore, an approach called boosting is used for selecting the most discriminating features for classification. The boosting approach for feature selection and efficient classification is briefly explained later.

An important property of LBP is its computational simplicity, which makes it possible to analyze images in challenging real-time settings. Due to its discriminative power and computational simplicity, LBP texture operator has become a popular approach in various applications.

2.1.2. Haar-like features

Haar-like features their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector [7].

As an alternative approach to the computationally expensive feature calculation using the usual pixel intensities (i.e., the RGB values), Papageorgiou et al. [9] discussed working with a feature set based on Haar wavelets. Viola and Jones [7] adapted the idea of using Haar wavelets and developed the so-called Haar-like features. In this method adjacent rectangular regions are considered at a specific location in a detection window, the pixel intensities in each region are summed up and the difference between these sums is calculated. This difference is then used to categorize subsections of an image. Figure 5 shows different types of Haar-like feature.

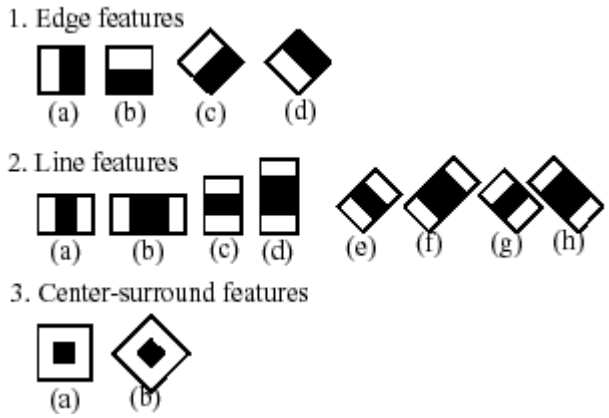

Figure 5. *Different types of Haar-like features*

The downside of a Haar-like feature is that it is only a weak classifier and a large number of Haar-like features are necessary to describe an object with sufficient accuracy. In the Viola–Jones object detection framework [7][9], the Haar-like features are therefore organized in something called a classifier cascade to form a strong classifier. The next section discusses cascaded classifiers and boosting.

2.1.3. Boosting and Cascade Classifiers

Boosting [12] is a general and provably effective method of producing a very accurate prediction rule by combining rough and moderately inaccurate rules of thumb.

The boosting technique was used by the Viola & Jones object detection framework [8] for forming a strong classifier by combining weak classifiers. The linear combination of weak classifiers is given by

$$H(X) = \sum_{i=0}^{n} w_i h_i(X) \tag{2.3}$$

where, $h_i(X)$ is the output of the *i*th classifier and w_i are the weights assigned to each weak classifier that are updated iteratively using the boosting approach.

AdaBoost [13] is an effective boosting algorithm to improve classification accuracies by combining a voted ensemble of weak learners. At each round, AdaBoost chooses the weak learners with lowest error, increases the weights of wrongly classified training samples and decreases the weights of those correctly classified training samples. The error rates for remaining classifiers are evaluated again over the entire training set. Thus the algorithm emphasizes more on samples that are more difficult to classify. A strong classifier is then formed as per the equation (2.3). Figure 6 explains the AdaBoost algorithm.

Several boosting techniques similar to AdaBoost have been proposed in the literature e.g., Gentle AdaBoost, Real Adaboost, Discrete Adaboost, etc. [14]. Most of these techniques differ only in their weighting strategies and are equally effective for this purpose.

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : X \to \{-1, +1\}$ with error

$$\epsilon_t = \Pr_{i \sim D_t} \left[h_t(x_i) \neq y_i \right].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$.
- Update:

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Figure 6. *The boosting algorithm AdaBoost [12].*

2.1.4. Cascade architecture

In an image there are much more non-object windows than object windows and therefore the average processing time of the windows is heavily influenced by the processing time of the non-object windows. In order to dedicate less time in processing non-object windows a cascade classifier architecture is proposed [7].

The term cascading refers to the process of creating a classifier architecture that consists of several simpler classifiers (stages) that are applied subsequently to a region of interest until at some stage the candidate is rejected or all the stages are passed. A simple illustration of the cascade architecture is shown in Figure 7.

Stages in the cascade are constructed by training classifiers using a boosting algorithm like AdaBoost and then adjusting the threshold to minimize false negatives [7]. Many non-object windows are very different to object windows and hence easy to classify them as non-objects. Most windows will be classified as non-object windows by the first stages of the cascade. Simpler classifiers are used to reject the majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates.

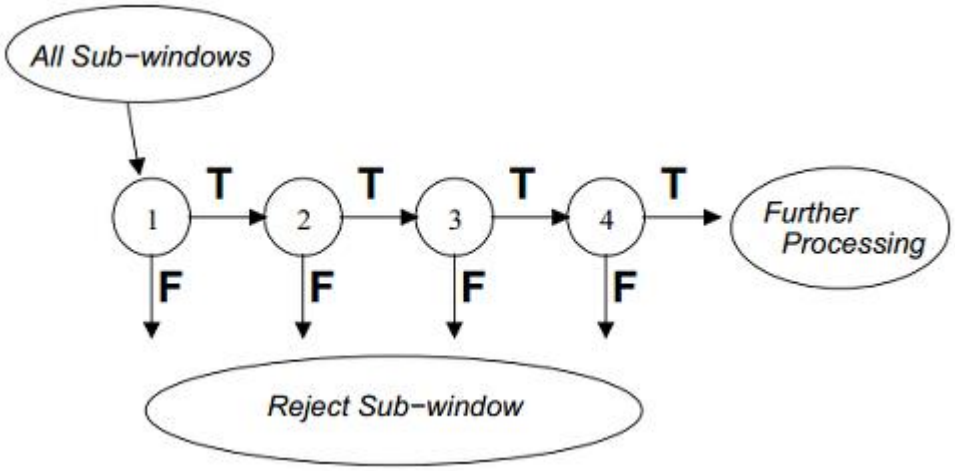

Figure 7. *Illustration of the cascade architecture used in object detection*

While training a cascade of classifiers, a target is selected for the minimum false positives rate and the maximum detection rate. Training of each stage is done by adding features until the target detection and false positives rates are met. Stages are added until the overall target for false positive and detection rate is met.

Cascading of boosted classifiers has been proven to not only give better detection accuracy, but also drastically decrease the computation time. For more details on object detection using boosted cascade classifiers, readers may refer to [7][9][12][13][14].

2.2. Motion analysis

In order to get semantic information from a video sequence such as detecting a salient event, in addition to object detection, it is important to track objects and analyze their motion and their interactions with other objects in the scene. There have been lot of methods proposed in the literature of computer vision for object tracking. In the following sections we will look at some of these methods.

2.2.1. Motion detection and analysis

One of the simplest ways to track an object is by getting a difference of intensity values of two consecutive frames in a video sequence. This method is very useful when there is a steady background and a distinct object in motion in foreground. A simple frame difference would result in motion contours as illustrated in the Figure 8. This motion contour can be then threshold to give a binary image describing a distinctive shape which may be analyzed to form a shape descriptor to detect and track the object.

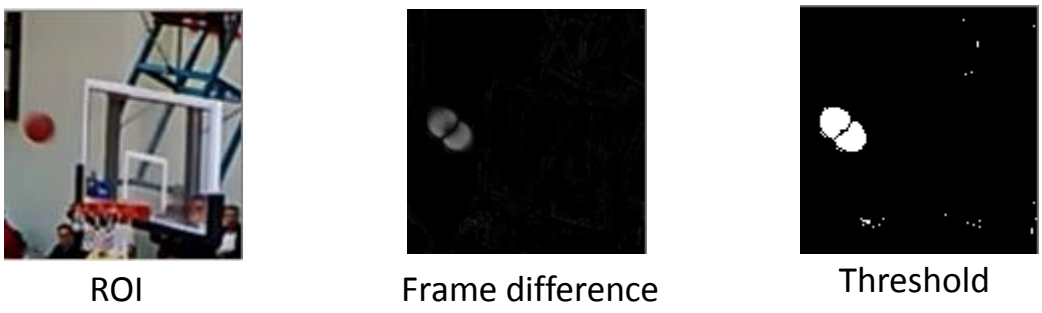

Figure 8. *An example showing the ROI, the respective frame difference between current and previous frames and its thresholded counterpart.*

Frame difference between two consecutive frames *k* and *k-1* at pixel (x, y) is calculated by

$$d_k(x,y) = |I_k(x,y) - I_{k-1}(x,y)| \tag{2.4}$$

where $I_k(x, y)$ is the intensity of pixel (x, y) in *k*th frame. Theoretically, if $d_k(x,y)$ is 0, it indicates that the pixel at (x, y) has not changed across frames and hence does not belong to the moving object. However, in reality there could be some noise that could generate a non-zero value for $d_k(x,y)$ even if there is no actual motion at that pixel. To overcome this problem we use thresholding. We set the difference to be 1 or 0 if it is higher or lower than the set threshold value respectively. The binarized frame difference $b_k(x,y)$ got by using threshold value *T* is given by

$$b_k(x,y) = \begin{cases} 1, & d_k(x,y) > T \\ 0, & d_k(x,y) \leq T \end{cases} \tag{2.5}$$

The motion contour resulting from the binarized frame difference is analyzed for size and shape in order to determine if the observed motion is occurring from the object of interest. This could be done with the help of a shape descriptor describing the shape features of the object of interest. Simple shape features like the area of the motion contour and the dimensions of the bounding box of the motion contour may be sufficient to accurately detect motion of the object of interest. Figure 8 shows an example of frame difference and thresholded difference.

2.2.2. Background subtraction by adaptive mixture modelling

In case of a static camera, analyzing motion from frame differences and motion contours is quite effective in segmenting foreground moving objects. However in some cases the background is not always stable as it may contain objects that are not perfectly steady. Frame differences with constant tiny movements in the background (for e.g., fluttering of leaves of a tree) or lighting changes in the scene would result in a lot of motion contours and make it difficult to detect the foreground object and its motion.

'Background subtraction' [22] is a technique used to effectively detect and identify an object against a changing background by minimizing or eliminating the motion contours resulting from motion of objects in background. This is done by creating a model of the background that contains cumulative motion over a certain number of consecutive frames. A standard method of adaptive background modelling is averaging the images over time, creating a background approximation which is similar to the current static scene except where relevant motion occurs due to foreground object. 'Background subtraction' can therefore be defined as a method of segmenting the foreground object by thresholding the error between an estimate of the image without moving foreground objects and the current image.

There have been various methods for background subtraction proposed in literature. A standard method of adaptive backgrounding is averaging the images over time, creating a background approximation which is similar to the current static scene except where motion occurs. This method faces challenges with scenes with many moving objects particularly if they move slowly. Changes in scene lighting can cause problems for many backgrounding methods. Ridder et al. [20] modelled each pixel with a Kalman Filter which made their system more robust to lighting changes in the scene. While this method does have a pixel-wise automatic threshold, it still recovers slowly and does not handle bimodal backgrounds well. Koller et al. [21] have successfully integrated this method in an automatic traffic monitoring application.

Stauffer & Grimson [22] proposed the use of Gaussian Mixture Model where instead of explicitly modelling the values of all the pixels as one particular type of distribution, the values of a particular pixel are modelled as a mixture of *K* Gaussian distributions (*K* is a small number from 3 to 5). Based on the persistence and the variance of each of the Gaussians of the mixture, the Gaussians corresponding to background colors are determined. Pixel values that do not fit the background distributions are considered foreground until there is a Gaussian that includes them with sufficient, consistent evidence supporting it.

Each pixel in the scene is modelled by a mixture of *K* Gaussian distributions. The probability that a certain pixel has a value of x_N at time *N* can be written as

$$p(x_N) = \sum_{k=1}^{K} w_k \, \eta(x_N; \theta_k) \tag{2.6}$$

where w_k is the weight parameter of the *k*th Gaussian component. $\eta(x; \theta_k)$ is the Normal distribution of *k*th component for *D* dimensions is represented by

$$\eta(x; \theta_k) = \eta(x; \mu_k, \Sigma_k) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_k|^{\frac{1}{2}}} \, e^{-\frac{1}{2}(x-\mu_k)^T / \Sigma_k (x-\mu_k)} \tag{2.7}$$

where μ_k is the mean and $\Sigma_k = \sigma_k^2 I$ is the covariance of the *k*th component. Here, the notation $(\mathrm{x} - \mu_k)^T$ indicates the transpose of $(\mathrm{x} - \mu_k)$. The complete Gaussian mixture model is parameterized by the mean vectors, covariance matrices and mixture weights from all component densities.

The first *B* distributions based on the fitness value w_k/σ_k are used as a model of the background of the scene where *B* is estimated as

$$B = \arg\min_b \left(\sum_{j=1}^{b} w_j > T \right) \tag{2.8}$$

The threshold *T* is the minimum fraction of the background model. In other words, it is the minimum prior probability that the background is in the scene. Background subtraction is performed by marking any pixel as foreground pixel that is more than 2.5 standard deviations away from any of the *B* distributions.

The first Gaussian component that matches the test value will be updated by the following update equations for maximum likelihood parameters as per the EM (Expectation Maximization) algorithm [24][25][27].

$$\begin{aligned}
\hat{w}_k^{N+1} &= (1-\alpha)\hat{w}_k^N + \alpha\hat{p}(\omega_k|\mathrm{x}_{N+1}) \\
\hat{\mu}_k^{N+1} &= (1-\alpha)\hat{\mu}_k^N + \rho\mathrm{x}_{N+1} \\
\hat{\Sigma}_k^{N+1} &= (1-\alpha)\hat{\Sigma}_k^N + \rho(\mathrm{x}_{N+1} - \hat{\mu}_k^{N+1})(\mathrm{x}_{N+1} - \hat{\mu}_k^{N+1})^T \\
\rho &= \alpha\eta\left(\mathrm{x}_{N+1}; \hat{\mu}_k^N, \hat{\Sigma}_k^N\right) \\
\hat{p}(\omega_k|\mathrm{x}_{N+1}) &= \begin{cases} 1 \text{ ; if } \omega_k \text{ is the first match Gaussian component} \\ 0 \text{ ; otherwise} \end{cases}
\end{aligned} \tag{2.9}$$

where ω_k is the *k*th Gaussian component and $\hat{w}_k^N$, $\hat{\mu}_k^N$ and $\hat{\Sigma}_k^N$ are the updated weight, mean and covariance respectively at *N*th iteration. $1/\alpha$ defines the time constant which determines change. There are several tutorial introductions to EM, including [26][23][28].

If none of the *K* distributions match that pixel value, the least probable component is replaced by a distribution with the current value as its mean, an initially high variance, and a low weight parameter.

This system seems to deal robustly with repetitive motions of scene elements, lighting changes, slow-moving objects, and introducing or removing objects from the scene.

The standard GMM update equations were extended in KaewTraKulPong and Bowden [15] to improve the speed of adaptation of the model. Zivkovic[16][17] improved the adaptive GMM background subtraction scheme by proposing an algorithm that can automatically select the needed number of components per pixel and in this way fully adapt to the observed scene. The processing time is reduced and the segmentation is slightly improved.

2.3. Shape recognition and analysis

In order to identify a segmented foreground object in a scene, there are various methods in literature involving complex structure analysis, shape descriptors, polygon approximation etc. In the following sections two significant and widely used methods for shape recognition are discussed.

2.3.1. Moment invariants

Moment invariants are one of the most popular and widely used contour-based shape descriptors. Using moments in shape recognition gained prominence when Hu [33] derived a set of invariants. These geometrical moment invariants have been then extended to larger sets by Wong & Siu [30] and to other forms by Dudani et al. [31] and Liao & Pawlak [32].

For a grayscale M x M image with pixel intensities $I(x,y)$, raw image moments $m_{p,q}$ of the order $(p+q)$ are defined as

$$m_{p,q} = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} x^p y^q I(x,y) \qquad \text{p,q=0,1,2,3...} \qquad (2.10)$$

The moments for $I(x,y)$ translated by an amount (a, b), are defined as,

$$\hat{m}_{pq} = \sum_x \sum_y (x+a)^p (y+b)^q I(x,y) \qquad (2.11)$$

Thus the central moments μ_{pq} can be computed from (2.11) by substituting $a = -\bar{x}$ and $b = -\bar{y}$ as,

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q I(x,y) \qquad (2.12)$$

where $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$

The central moments on applying scaling normalization change to

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{pq}^{\gamma}}, \qquad \gamma = \left[\frac{p+q}{2}\right] + 1 \tag{2.13}$$

Hu [33] defined seven values, computed by normalizing central moments through order three, that are invariant to object scaling, translation and rotation. In terms of the central moments, the seven Hu's moment invariants are given as

$$\begin{aligned}
M_1 &= (\eta_{20} + \eta_{02}) \\
M_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
M_3 &= (\eta_{30} - 3\eta_{02})^2 + (3\eta_{21} - \eta_{03})^2 \\
M_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
M_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
M_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
M_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{2.14}$$

Derivation of these results involves concepts beyond the scope of this discussion. Interested readers may refer to [33]. The seven Hu's moment invariants may be used as feature vectors to be used with a pattern classification system.

2.3.2. Polygon approximation

Sometimes the shape recognition problem can be simple as recognizing simple geometrical shapes. Polygon approximation is a technique where a digital boundary is approximated with arbitrary accuracy by a polygon. This technique is useful and very effective in object detection by recognizing simple shapes from a binary image representing the foreground mask of the object in scene.

First the contours are extracted from the mask image that gives the edges or the digital boundary of the shape. To identify shapes, the algorithm of Ramer-Douglas-Peucker (RDP) algorithm [35][36][37][38] is used. The algorithm works recursively by the method of "divide and conquer". The algorithm simply tries to reduce the number nodes or vertices in a polyline (n nodes). The node farthest from an imaginary line formed by two nodes is discarded if its distance from the line is below threshold, otherwise the farthest node is chosen as one of the points and the process is repeated recursively. An example of the RDP algorithm over a polyline is illustrated in Figure 9.

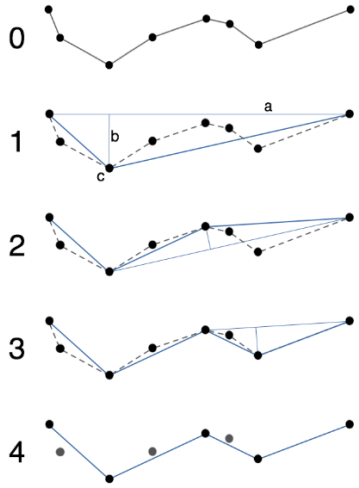

Figure 9. *Simplifying a piecewise linear curve with the Douglas-Peucker algorithm.*

The number of vertices remaining after running the RDP algorithm can give an approximation of the shape of the object. If the number of vertices are more than six than the object can be assumed to be of circular shape. This method quite simple and yet effective and is often used in cases where objects to be classified are of distinct geometric shapes. Interested readers may refer to [41] to read more about the RDP algorithm. Soendoro et al. [39] and Salhi et al. [40] have demonstrated the use of this method for recognizing traffic signs.

3. PROCESS PIPELINE AND FRAMEWORK

This chapter gives an overview of the whole framework of developed summary creation system. The summary creation process consists of two main stages:

1. Saliency detection
2. Video remixing

The first stage deals with detecting salient events from an input video file. The saliency detection module processes each frame of the video and applies machine learning techniques combined with image processing and computer vision algorithms to identify parts of the video where a scoring attempt occurs. The saliency detection module was implemented in C++ using OpenCV. OpenCV [35] [34] is an open source computer vision library that offers implementations for various machine learning and image processing algorithms that are useful in tasks related to object recognition and pattern classification. The output of the first stage is a list of salient timestamps, each representing a scoring attempt. Other information related to each salient event such as confidence value and detection of successful score is also output to the list.

The second stage comprises of the actual summary creation process. It takes as input the salient timestamps resulting from the previous stages and other video files that are simultaneous recordings of the game by various cameras. First these videos are time synchronized based on an audio alignment algorithm. Then relevant sections matching the salient timestamps are cut from each of the videos. These cut sections associated with a single salient event are processed with certain filters and then merged together into one continuous video stream to form a salient event segment. Each salient segment is applied some fading effects and then appended to a continuous video stream to form the summary video. Finally a background audio (e.g. some music) is filtered and added to the video stream to form the final summary video. This module was developed using Python and the open source library FFmpeg [34].

A detailed schematic diagram of the framework is shown in Figure 10. In the following chapters, the operations in each of these stages are explained in detail.

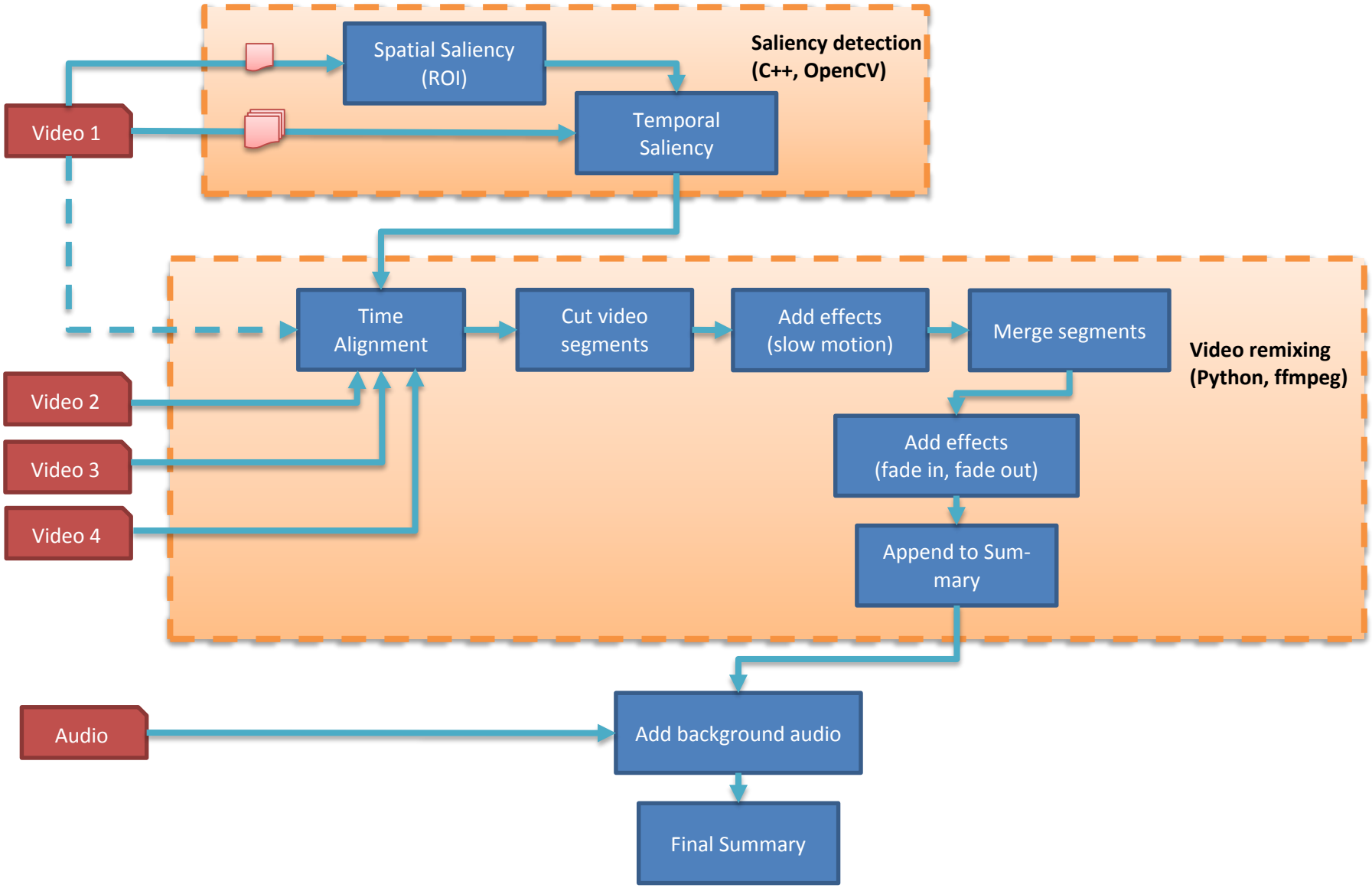

Figure 10. *Schematic diagram showing the process pipeline and various modules*

4. SALIENCY DETECTION

The term saliency in general means being most noticeable or important. In the context of this thesis, saliency in a video would naturally be the important section of the video that is most appealing to the viewer and hence in a basketball game would mean those important segments of the game that would appeal as highlights of the game or in other words parts that consists of action that should not be missed by the viewer.

To define saliency in the basketball game various highlights videos that were created by expert media professionals in sports recording and editing were observed and studied. Also valuable inputs from some experts and basketball fans were collected. The top events occurring in a basketball game that could be categorized as a salient event would be a successful scoring attempt and a missed scoring attempt. Other than a scoring attempt event, one may also consider events during the game that may be of interest to the viewer. For example, a bad foul committed by a player resulting in an important game changing situation, peculiar reactions of the coach or some strange and interesting behavior by the team mascot or someone in the audience. However detecting these kinds of saliencies are much more complicated and beyond the scope of this thesis project.

Thus for this project we considered defining the salient event as an event time frame where a scoring attempt, successful or failed, is made by either team. A salient segment could therefore be defined as a segment of the video containing frames a few seconds before and a few seconds after the occurrence of the salient event. Thus a salient segment would include few seconds of the play before the scoring attempt and the celebration after the score or the players' reactions in case of a failed attempt.

Saliency detection process can be broken down to two stages namely spatial saliency detection and temporal saliency detection.

- **Spatial salient areas detection (Detection of Region of Interest).** This involves in analyzing a single frame of video to identify a salient region of interest. The salient region of interest (ROI) is a window region in a frame that contains the object of interest which may be one of the basket hoops or the game ball.
- **Temporal salient segments detection (Detection of scoring attempt timestamps)** which is detection of the salient action event like a scoring attempt. This involves processing multiple consecutive frames for ROIs and analyzing the position and motion in and around these ROIs

Figure 11 shows a schematic work flow of the saliency detection process.

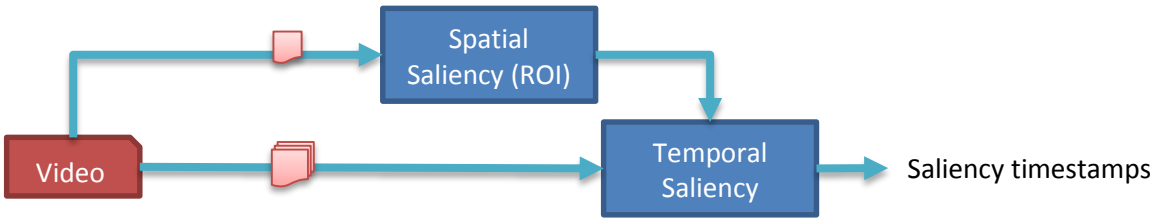

Figure 11. *Schematic work flow of saliency detection.*

In a basketball game video, the approach to detect a salient event defined by a scoring attempt (successful or failed) involves detecting video frames where the ball is seen close to one of basket hoops on either side. A rather simple and straightforward technique to achieve this is by using object detection techniques discussed earlier and detect the two objects of interest i.e., the ball and the basket, and then determine their relative position to estimate the validity of a scoring attempt. This method is a prior art [42] and was used as a starting point in this thesis project. This was later improved by using other techniques such as motion analysis and background subtraction that are discussed in detail in the following chapters.

4.1. Detecting objects of interest

In the basketball game the two primary objects of interest to be identified for saliency detection are the ball and the basket. Cascaded classifier based on LBP as discussed earlier are used for this purpose of object detection.

The first step is to train the classifiers. Over 1000 images obtained from various basketball videos and photos were used to train each of the following classifiers.

- Classifier for ball detection
- Classifier for left side basket detection
- Classifier for right sided basket detection

These classifiers are implemented using an open source library OpenCV. OpenCV supports both Haar [7] and LBP [3] features. LBP are several times faster than Haar features in both training and detection. The quality of detection in both LBP-based and Haar-based classifier is influenced by the quality of the training dataset and the training parameters used. It has been seen by experimentation that it is possible to train a LBP-based classifier that will provide almost the same quality as Haar-based one. For this reason LBP-based classifiers were chosen for this task.

These classifiers as mentioned earlier are known to work better when trained with larger dataset. As the left side basket and the right side basket are identical in shape and structure and appear mirrored with respect to each other, by vertical flipping of the images of the

left basket we could increase the dataset used to train the right side basket and vice versa, thereby increasing the overall probability of successful object detection.

Once these objects are detected in a given frame of video, their relative positions are analyzed. Based on the distance of the ball from the basket the decision on saliency is made. If the ball is found too close to the basket, the frame can be considered to be a part of an event where at least an attempt to score has been made and hence marked as a salient frame.

Figure 12 shows a simplified flow chart of the salience detection process using the prior technique.

The video input for object and saliency detection used in this method is one from a panning camera with a wide angle view and possibly zooming in and out during recording. This is very straightforward method where all possible objects i.e., the ball and the baskets are searched in every frame of the video. One may clearly see this as a performance bottle neck.

Out of the various improvement ideas that emerged through discussions some are presented. One approach in optimizing the ball detection is to use some kind of object tracking mechanism as the ball is most likely to be found near its previously detected region in the previous frame. Also since we are interested only when the ball is near one of the baskets, instead of running ball detection on the entire frame, only the region around a detected basket may be considered. Figure 13 shows a typical region of interest considered for ball detection around a detected basket region. It was seen by experimentation that though these optimization techniques improved the performance of detection to certain extent the overall quality of saliency detection was not very satisfactory. This has been mainly because of problems in accuracy of ball detection. It was noticed that the traditional object detection methods fail in case of detecting a ball against a detailed and noisy background. Also the players' heads were sometimes falsely detected as the ball.

In order to improve the saliency detection other methods were considered involving motion based analysis of the frames that focus mainly on the motion in the region of interest. These methods are discussed in the following sections.

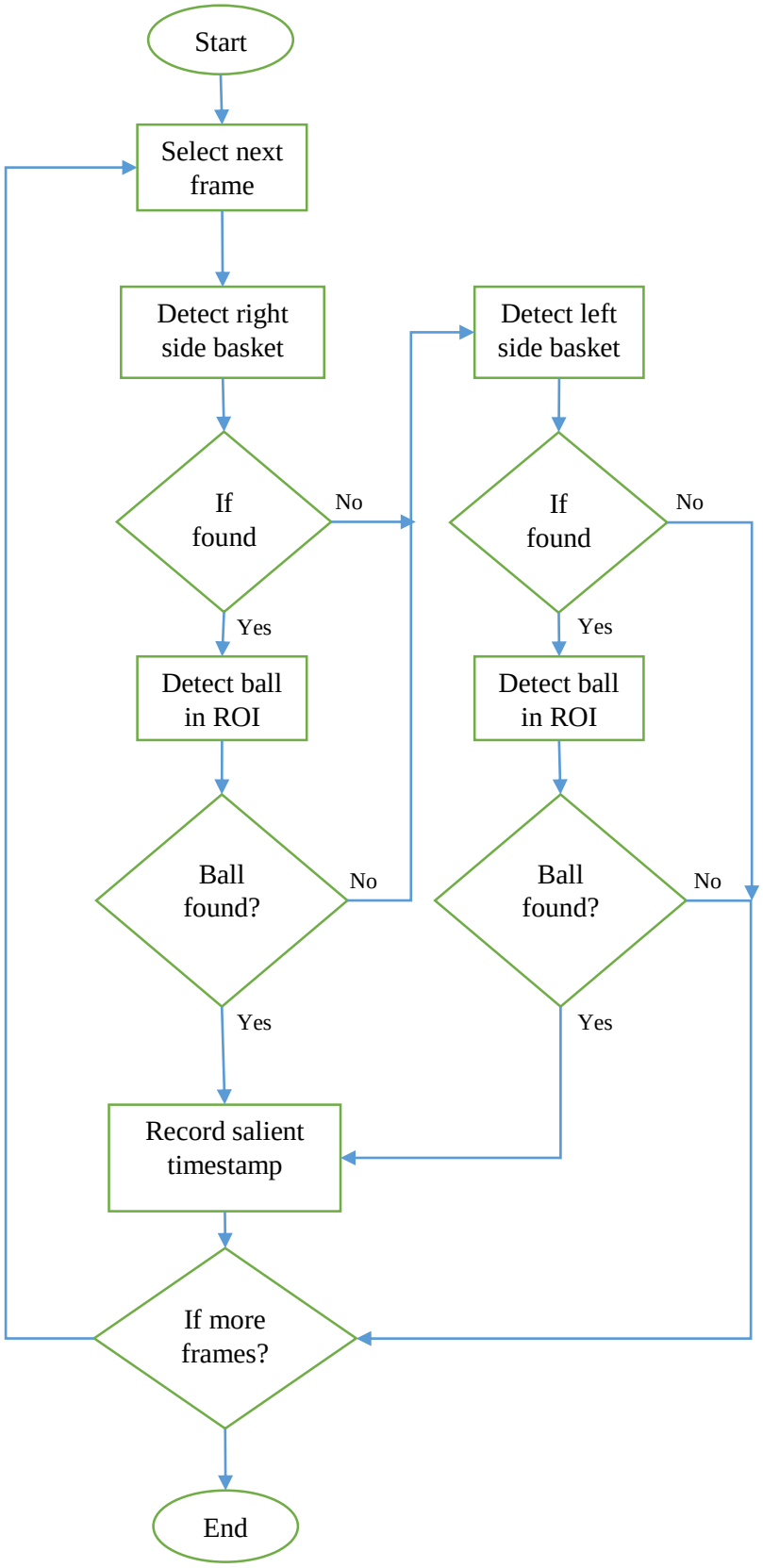

Figure 12. *Flow chart of the saliency detection process using prior art technique*

Figure 13. *Snapshot of an example salient event. The green box defines the Region of Interest (ROI) and the blue box indicates the detected ball in the ROI.*

4.2. Motion Analysis

As discussed earlier this approach of saliency detection involves analyzing motion in and around the basket. To do this first we need to define our region of interest. As shown in the Figure 13, the region of interest is chosen as a larger window surrounding the window identifying the basket.

The main limitation with this method is that the video input needs to be from a camera with a fixed view pointing towards one of the baskets. Although this may seem like a big limitation in the beginning, in real life situations where multiple cameras are being used to record games, this requirement does not seem so much of a hindrance. Moreover, the greater quality of saliency detection achieved using this method justifies this requirement.

Analyzing motion in the region of interest to detect saliency consists of the following four steps.

1. Calculate frame difference between current and previous frames.
2. Threshold frame difference to get motion contours.
3. Apply noise reduction techniques to filter out noise and enhance motion contours
4. Analyze the shape of the motion contour to determine saliency

The frame difference image is a binary image obtained by processing two consecutive frames. It may be calculated in one of the two following ways.

- Convert the ROI window to grayscale and get the absolute difference in pixel values
- Get absolute difference in pixel values of each of the red, green and blue channels in the RGB window frames and average these differences to form a grayscale frame difference image.

The next step is converting the grayscale frame difference image into binary to get the motion contours. This is done by thresholding. A threshold value is chosen by experimentation to get the most accurate motion contours.

Figure 8 shows an example of the frame difference image and its threshold counterpart.

There is always some noise present in the frame difference that represent irrelevant motion mostly arising from change in lighting, movement of crowd or slight camera shake etc. This noise appears as tiny unconnected white patches as can be seen in Figure 8.

4.3. Enhancement and Noise removal

As mostly the noise found in the frame difference is white noise, one way to eliminate or reduce them is by applying blur. Blurring is also useful in bridging gaps in image with broken contours. When applied before thresholding, blurring the frame difference image greatly reduces the noise and unwanted contours. Another set of widely used techniques effective in reducing this kind of noise is morphological transformations.

Morphological transformations are some simple operations based on the image shape normally performed on binary images. A small window (typically of size 3x3 or 5x5) is chosen as a structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation.

4.3.1. Erosion

The concept of erosion is to reduce the size of each contour by eroding away its boundaries (assuming the foreground is in white). The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero). As a result of this, all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises.

4.3.2. Dilation

The dilation process is just opposite of erosion, where the idea is to increase the boundary of the foreground object mask and bridge gaps between fragmented components. The operation is similar to that erosion, where the kernel slides through the image, but here, a

pixel element is set to 1 if at least one pixel under the kernel is 1. As a result it increases the white region in the image or size of foreground object.

For the purpose of noise removal, erosion is followed by dilation. As erosion removes white noises, it also shrinks the object. When the image is dilated after erosion the removed noise won't come back, but the object area increases to its approximate original size and shape.

An example of the operations of erosion and dilation applied on the foreground mask image of the ROI is shown in Figure 14.

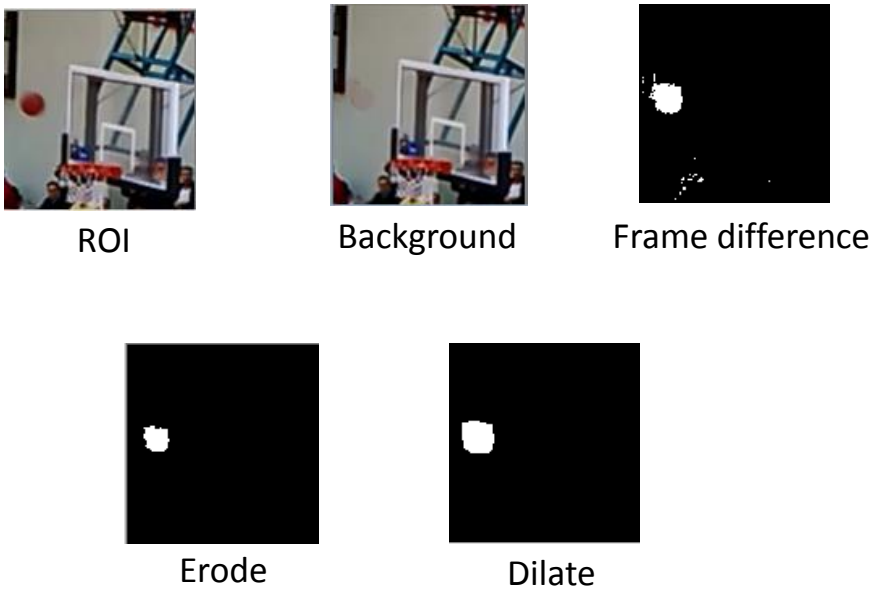

Figure 14. *For a sample frame of a scoring attempt where the ball is just entering the ROI, the modelled background image along with output of different stages of processing.*

4.4. Discussion and improvements

In spite of the operations including thresholding and noise removal, the motion contour appears as an irregular blob rather than having the distinct round shape as that of the ball. However, this is still very useful and can identify and classify most cases correctly. But sometimes an occlusion of region of interest for example by someone in the audience blocking the view of the camera or movement by some person seated right across the court behind the basket can produce a huge blob of motion contour making the situation difficult to guess. The background subtraction method discussed in the next section 4.5 seems to be more effective in this case.

In order to optimize and improve performance, following methods were considered. Instead of calculating frame difference for every frame, alternate pairs of frames may be taken into account. Since in a genuine scoring attempt the ball is expected to found in the

region of interest at least for a few frames of video, this method does not harm the quality of detection whereas the computation is reduced to half.

However, the main goal is to accurately detect motion due to the ball in the region of interest it is important to produce motion contour that can be easily identified as that occurring from the ball. In other words, the motion contour should be round in shape. One approach to get this kind of motion contour is to have one frame as a reference frame where no ball is present in the ROI and compare consecutive frames to this reference frame. The reference frame can be chosen to be the frame just before the first frame where the ball seems to be entering the ROI.

4.5. Background subtraction

The concept of having a single frame as a reference frame for each salient event detection works in the way that it identifies the ROI in the reference frame as the background and distinguishes the foreground object (not seen in the reference frame, but seen in other frames) against the reference background. Another approach based on the similar concept is the background subtraction method with background modelling.

Background modelling is a method of creating a Gaussian mixture model of the background by cumulatively processing certain number of consecutive frames. As explained in section 2.2.2 of chapter 2, this is very useful in eliminating small irrelevant motion in the scene.

The method of background subtraction is applied in solving the problem of ball detection in the ROI, where certain number of consecutive frames are added to create a background mixture model and then every new frame is compared to this background mixture to distinctly identify the presence of the foreground object i.e., the ball in the ROI. Figure 14 shows an example of the background model and the respective motion contour obtained using this background against a frame where the ball is present in the ROI.

Once we have the motion contour, the next step is to have the contour analyzed for its shape. But before that we need to get rid of unnecessary contours or noise from the residual image. The goal is to have a single large distinct round shaped contour if present and eliminate the rest as noise. Noise removal methods such as erosion and dilation when applied in a sequence seemed to be very effective in this case. Once the noise is removed the shape of the largest contour is analyzed. The next section describes methods of shape identification used in this project.

4.6. Shape identification

As we are trying to detect the ball, we are looking for a contour that is round in shape. One simple and computationally inexpensive way that used in earlier stages of this project

is to check if the bounding box of the contour is a square i.e., if the width and height of the bounding box of the contour are same or within close range.

This seemed quite effective in detection of the ball but also resulted in false positives in many cases as a square bounding box does not always indicate a circular contour shape.

As it was noticed, an effective shape recognition method was required to accurately detect the ball in the ROI and get rid of the classification errors. Two methods of shape recognition discussed in section 2.3 were considered. The first one using Hu's moment invariant seemed a bit too heavy as the task was to identify a simple circular shape. Therefore the simpler method of polygon approximation was chosen.

The polygon approximation method implemented as per the Douglas-Peucker algorithm [35][36][38] was used. This method is quite effective in identifying the circular shape by producing an approximate polygon with more than six vertices. For more accuracy in ball detection, if the shape is found to be circular after polygon approximation, the area of the contour is checked to be within certain threshold limits of the area of a circle that could be inscribed in the bounding rectangle of the contour.

$$A = \pi r^2 , \qquad r = \frac{w}{2} \tag{4.1}$$

where A is the area of the circle with radius r and w is the width of the bounding box of the contour. Area of contour A', (obtained by counting the white pixels) is compared to the area of the circle A. The ball detection is positive if $|A - A'| <$ threshold T.

4.7. Detecting salient events

This section explains the procedure for detecting the timestamps of salient events that is scoring attempts based on the ROI and ball detection method described earlier. The idea is to mark all those frames where the ball is seen very close to the basket and then analyze the timestamps of these frames and make a calculated guess of whether these belong to a genuine scoring attempt event.

A scoring attempt would comprise of frames where the ball is either entering the basket hoop or bouncing off the hoop or the basket wall. In all these cases the ball is expected to be in the ROI at least for a few seconds. In a typical video with a frame rate of 24 or 30 fps, one would expect the ball to be detected for a certain number of consecutive frames for at least 2 seconds. This is the underlying logic used in determining a scoring attempt.

As we are using still camera in this case the basket detection is run only for the first few frames. Once the basket is found with good confidence, the ROI is defined. Figure 15 show the flow chart for the ROI detection process. Then for each frame the ball detection algorithm as explained earlier is run on the ROI window. If the ball is detected, the

timestamp is recorded to a result file. Figure 16 shows the flow chart for ball detection. To improve performance and avoid false positives, an extra step is added to check if there are at least two consecutive positive ball detections within one second time gap. The second timestamp is then recorded.

Now, we can see that for a single scoring attempt there would be multiple consecutive positive ball detections and therefore many salient timestamps. We actually don't want all timestamps related to a single scoring attempt. If we assume that a typical scoring attempt would last for three seconds, then we can ignore any positive ball detections if a salient timestamps is already recorded within three seconds in the past. This works well as we would then have one salient timestamps recorded for one scoring attempt. This would also work well in cases where there are multiple attempts in very short duration, e.g. when the ball bounces off the wall and a repeated attempted to score is seen. Figure 17 illustrates the simplified code flow for the salient event detection process.

The output of the salient event detection process is the set of timestamps, where each timestamp corresponds to a detected salient event. As the input is video from fixed camera recording only one side of the playing court (pointing to one of the baskets) this process needs to be carried out twice, once for the left side basket and once for the right side basket using two fixed camera input videos corresponding to the respective sides.

Once the salient event timestamps are obtained for each side these are merged in chronological order into one list of salient timestamps. Before merging any time alignment difference between the two videos needs to be taken care of. The time offset between these two videos may be input manually or can be calculated using the audio-based alignment module as in [19].

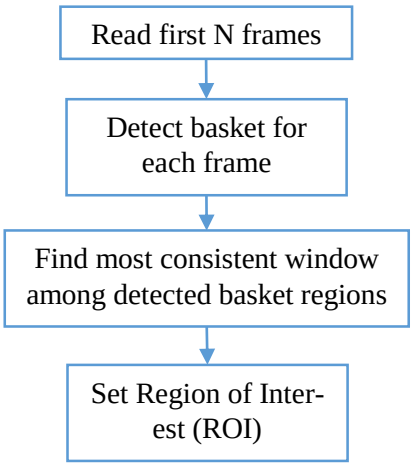

Figure 15. *Flow chart of the ROI detection process*

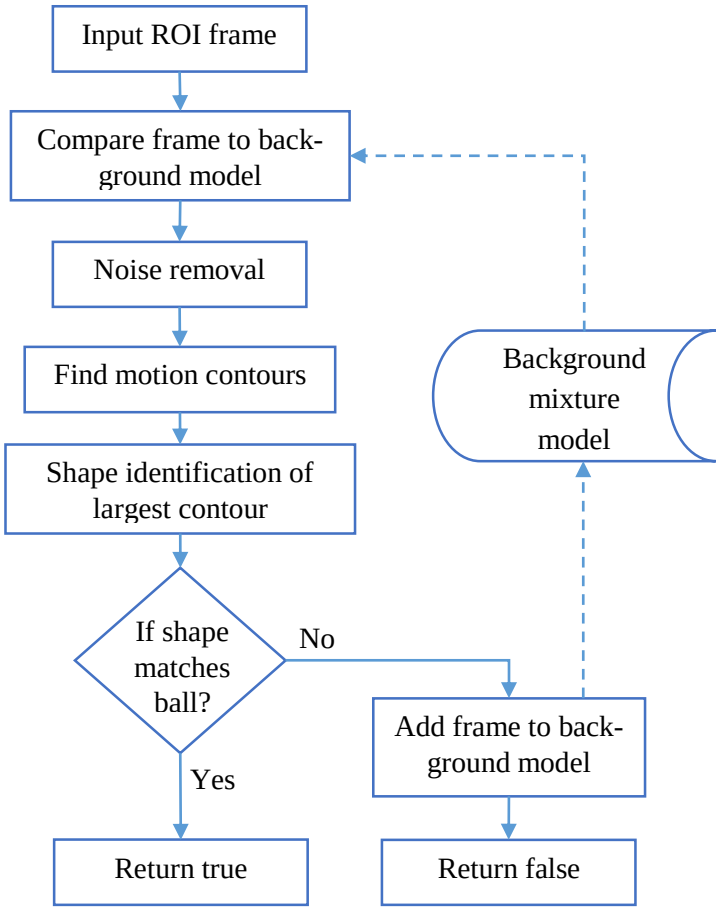

Figure 16. *Flow chart of the ball detection process*

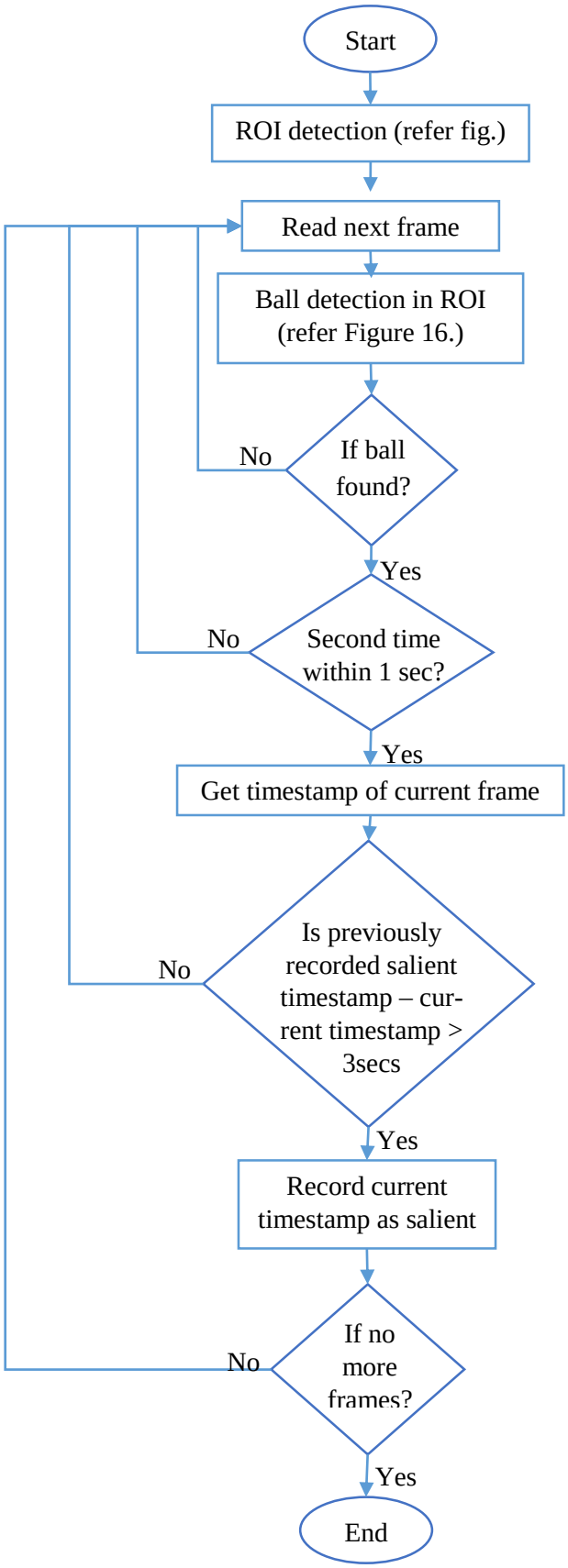

Figure 17. *Flow chart of the refined and complete saliency detection process*

4.8. Detecting successful attempts

Though scoring attempts are important highlights of the game, many fans may argue that the most important ones are the attempts that resulted in successful score. A successful attempt in this context means an event when an attempt to score results in the ball passing through the basket hoop. This however in the real game may not always result in a point being awarded to the scoring team as in rare certain cases there could be foul played, time run out or some other reasons related to the rules of the game. For the scope of this project we consider an event of the ball passing through the hoop as a successful attempt. The purpose of classifying the scoring attempts to successful attempts is that give a higher importance to these salient events. This classification is useful if the user wants to choose the top N number of highlights for customized summary of the game.

In order to classify an attempt as a successful attempt, motion in the net region of the basket needs to be analyzed. The ROI is divided into 9 blocks and the lowest middle block is assumed to contain the net of the basket. This block is now treated as the inner ROI. Figure 18 illustrates the estimation of inner ROI from the ROI.

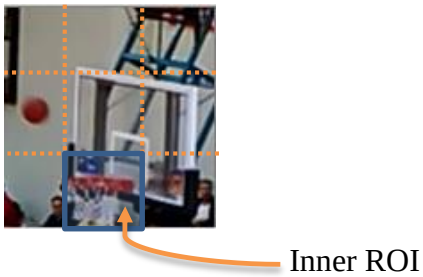

Figure 18. *The ROI is divided into 9 blocks and the bottom middle block is chosen as the inner ROI for detecting successful attempts.*

Similar motion analysis techniques used earlier are used again this time with this inner ROI to get motion contours. But now we are not interested in detecting any object, but just motion within that region. If motion contours are detected in the inner ROI consecutively for at least three frames during a scoring attempt event (as detected earlier), then it is classified as successful attempt.

4.9. Confidence values

In a typical object detection and recognition tasks, confidence values for each detection is used to indicate how strongly the detected result can be considered or in other words how confident is the object detection and recognition system about a particular result. In case of this project we use confidence values for each detected salient event to show how strongly the system recommends an event to be salient.

The logic used to calculate the confidence of a salient event is that the more the ball is seen in the ROI during a detected scoring attempt, the more confidence is assigned to that saliency.

These confidence values serves for the purpose of customized summary creation. Customized summary is where the creator of the summary can choose only a certain fixed number of the most salient events of the game to be the part of the summary while discarding the rest (that are less salient). The customized summary creation is described in detail later in the next chapter in section 5.6.

To calculate the confidence value for a salient event, the number of times when the ball is detected during a salient event is recorded. This number is then normalized over all the confidence values in the whole set of salient events. As a successful scoring attempt is assumed to be more salient than an unsuccessful one, the confidence value of every successful attempt is boosted by adding a fixed value of 1 to its normalized confidence score. Thus this gives us confidence values ranging from 0 -1 for unsuccessful attempts and values ranging from 1 -2 for successful attempts.

4.10. Common issues and counter measures

As discussed before, full or partial occlusions of the ROI makes it almost impossible to detect successful score by the method of motion analysis in ROI. The successful score detection is very challenging task as the inner ROI being analyzed may be occluded by the hand of a player attempting to score or block the attempt. Therefore, the motion seen in the inner ROI itself may not be sufficient in concluding a successful score.

The videos were analyzed to observe patterns that could be used to improve the detection results. One common observation was that the once the point is scored, there is no motion seen around the basket region i.e., the ball is not found in the ROI for at least a few seconds after the score. Another observation was that occlusion such as a players hand blocking the view of the inner ROI only lasts for a few frames, whereas when the ball passes through the net, motion contours appear to last for more number of consecutive frames, because of the movement of the net.

Taking these observations into account following guidelines were implemented for elimination of commonly seen false positives.

- When motion is detected in the inner ROI, check if there is motion in net region for at least certain (3) number of consecutive frames
- If ball is found again in the basket ROI in the immediate next few frames (3 seconds) after successful score detection, then eliminate the last detection as false positive.

These tweaks help to eliminate false positives to some extent. However further improvement in accuracy still remains a challenge.

5. VIDEO REMIXING

In this chapter, we will look at the process of mixing various camera feeds, applying filters for visual effects and mixing background audio to create the summary video containing the top highlights of the recorded game. The main goal is to have the process fully automated and yet create a beautiful and professional looking summary video. This process was implemented using Python script and an open source library called FFmpeg[34].

5.1. The art of summary creation

Sports video summarization is a creative process. The style in which a summary video is created depends on the individual creating the summary and the taste of the target audience. Therefore, there are no specifically defined rules for the process. Although, there are a few things that may be considered as guidelines while creating a high quality summary video. These guidelines are derived from the most commonly followed patterns by professional and creative media personal in today's world. In the following sections these guidelines are discussed.

5.1.1. Shot boundaries

In order to provide a good viewing experience, when mixing different camera view angles, one should take care of the timing of cutting the shot to another view angle. Viewers would not want to be distracted when focusing on an important action of the game. It is therefore a generally followed rule of thumb in sports recording to never cut a shot during a salient action. In other words never change the camera view when a basket is being scored, but rather change after the scoring attempt.

5.1.2. Using different camera angles

Choosing the right camera is important while mixing different views in the summary creation process. In order to decide which camera to use at what point, many professionally captured basketball recordings were studied and following guideline was formed.

- Choose a wide angle view showing few seconds before the scoring attempt till just the completion of the scoring attempt.
- Choose a zoomed-in or close-up view to show the replay of the action.
- Choose a third view different from the previous two to show the follow through a few seconds after the attempt. This third view could also be a close-up of players or coach on the bench or the audience.

Following this guideline, the method of mixing three camera views for a singles salient segment is described in detail later in section 5.3.

5.1.3. Action replay

An action replay of a salient event is considered important for most viewers. Most of the times the replay is shown at a slower speed allowing the viewer to get full grasp of the action. Making the whole replay segment play at constant slow speed could appear boring to some viewers. Therefore a replay with varying speed, while slowing it down only close to the salient event timestamp is visually more appealing and also adds a creative touch. The length and speed of the replay segment is purely subjective to the user. This process is discussed more in detail in section 5.4.1 and customizing of replay sections is discussed in later in section 5.6.

5.1.4. Length of each salient segment

As the salient segment is a segment of video containing the salient event, it includes the action from a few seconds before the salient event and a few seconds after it. The length of a salient segment comprising of a single game highlight is one of the most important things to consider while designing an automated summary creation system. If the length is too long it may be boring for some viewers and if it is too short it may have missed some important part of the action. Also while creating a salient segment since various clips corresponding to different cameras may be combined, the length of each of these clips are also something that needs to be paid attention to. After studying some video archive certain guidelines that were setup to be default configuration for this project are as follows.

- For the first clip, 7 seconds of playback before and 3 seconds after the salient event timestamp were added.
- For second clip, which is a slow replay, 3 seconds before and 1 second after the salient event timestamp were added.
- For third clip, 3 seconds of playback were added after the timestamp where the second clip ends.

These values were used only as a default configuration which the user could easily change for every new summary creation task. The clipping of videos by following the above guidelines is illustrated in Figure 17. The details of clipping and merging are explained in the section 5.3. The length of these clips could also be changed later in the customization process after the summary is created. Section 5.6 discusses this process in detail.

5.2. FFmpeg, video filters and filter graphs

In order to cut, clip and merge video segments, a software tool called FFmpeg was used. FFmpeg is a powerful, cross-platform and open source tool useful in editing video and audio files.

The FFmpeg project contains several libraries for multimedia processing. Some of them mainly used in this project are:

- *libavcodec* provides a decoding and encoding API, and all the supported codecs.
- *libavformat* provides a demuxing and muxing API, and all the supported muxers and de-muxers.
- *libswscale* provides a scaling and (raw pixel) format conversions API, with high speed/assembly optimized versions of several scaling routines.
- *libavfilter* provides an audio and video filtering API, and all the supported filters.

In addition to these FFmpeg comes with a command line tools such as:

- *ffmpeg* – a command line tool to edit and convert multimedia files between formats.
- *ffplay* – a simple media player based on FFmpeg libraries
- *ffprobe* – a multimedia stream analyzer
- *ffserver* – streaming server for live broadcasts

In this project we mainly use the command line tools ffmpeg and ffprobe along with python scripting. However the above mentioned libraries may also be directly used in C++ code. Details of such usage can be found in the official documentation page of FFmpeg (the "Libraries Documentation" section) [34].

The ffmpeg command line executable is called from a python script multiple times for various processing tasks such as cutting, merging, filtering etc. The standard output of each these executions of ffmpeg is parsed in the script. Interested readers may find the official documentation [34] of ffmpeg very useful.

The input to this script is a file containing the salient timestamps (output of saliency detection), recorded game video files and audio files (for background music).

FFmpeg comes with useful video filters provided by the library libavfilter. These filters can be combined to form a pipeline or graph of filters with which many kinds of video manipulation tasks can be achieved. An example of a filter graph is shown in Figure 19.

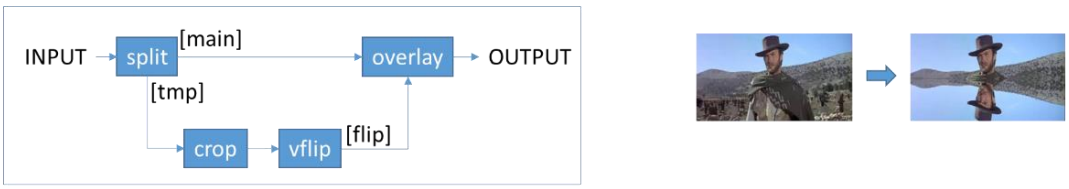

Figure 19. *An example of a complex filter graph and the ffmpeg command line arguments at the bottom.*

Some of the filters and sub-commands used with ffmpeg in this project are:

- *Split* - Clones an input into two intermediate outputs which may then be used as separate inputs to two different filter chains.
- *Overlay* - Overlay one video on top of another.
- *Scale* - Scale or resize the input video, using the libswscale library.
- *Trim* - Trim the input so that the output contains one continuous subpart of the input.
- *Concat* – Concatenate or merge two or more media files to play one after the other. This is a demuxer that demuxes the input files and adjusts the timestamps so that the first file starts at 0 and each next file starts where the previous one finishes. All files must have the same streams (same codecs, same time base, etc.).
- *SetPTS* – Set the presentation timestamps in video frames to alter the playback speed.
- *Fade* - Apply a fade-in/out effect to the input video.

For detailed usage of the above filters the reader may refer to the official FFmpeg documentation [34].

5.3. Cutting and merging salient segments

This section discusses the process of video editing to create the summary video that involves cutting relevant segments of videos from different camera feeds and merging them according to the salient event each of them represent. Before cutting the video clips, these videos have to be aligned as per real time. There are various ways of achieving temporal alignment. In this project, an audio based temporal alignment method was used based on [19].

The start and end timestamps around the salient event timestamps are determined for each input video for each salient event by following the guidelines explained earlier. These timestamps are adjusted for temporal alignment. Then these sections are cut accordingly using the ffmpeg tool to create intermediate video clips. Figure 20 shows views from three different cameras used in creation of a salient event segment. Figure 21 illustrates the process of creating a salient section from three video feeds. The second clip is edited to play in slower playback speed by using the setPTS filter in ffmpeg and overlaid in picture-in-picture format in bottom corner of the reused trimmed first clip using the overlay filter. The third clip contains action just after the salient event. These three clips are then merged so as to play one after the other using the concat filter. This intermediate merged clip now contains a sequence from three cameras showing a single salient event taking place. Once all the intermediate merged salient event clips are created, these clips are merged in to a full summary video using the same concat filter.

Optionally each of these salient clips may be filtered with some fade-in and fade-out effects at the beginning and end of each clip depicting the end of one and start of another highlight in the summary video.

(a)

(b)

(c)

Figure 20. *Example snapshots of three different views from three cameras chosen to make a salient event segment. (a) Wide-angle view that shows the action till completion of scoring attempt, (b) Close-up view showing a slowed down replay of the scoring attempt and (c) Third view showing the follow up after the scoring attempt.*

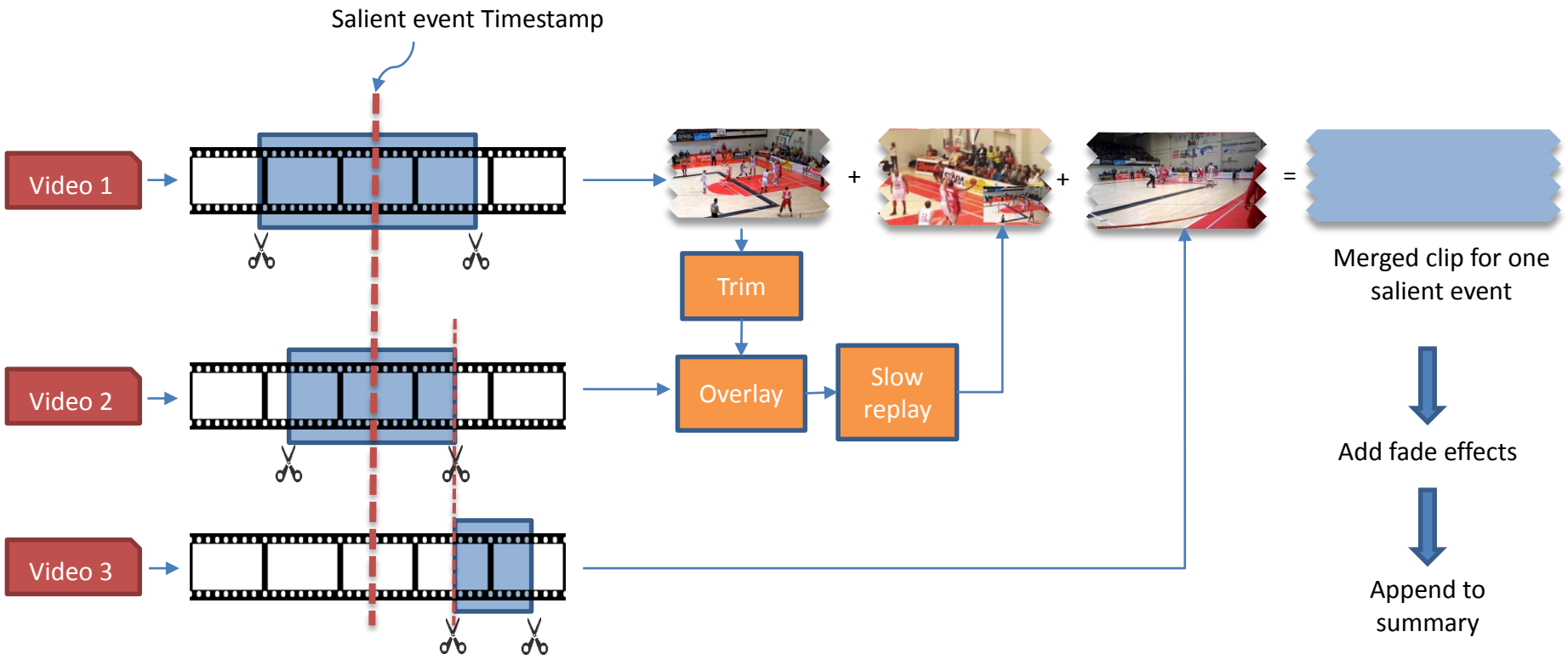

Figure 21. *Schematic diagram of the process of creating summary video by clipping and merging multiple video segments.*

5.4. Adding effects

In this section we discuss about various visual effects used to improve the overall quality of the summary video.

5.4.1. Slow replay

As discussed earlier, the slow replay is used on the camera with a close-up view of the action and played at slower speed immediately after showing the action in wide angle view. This is achieved by using the setPTS filter in the ffmpeg tool. SetPTS basically stands for setting presentation timestamps and its value is a division factor of playback speed. So for playing at half the speed, setPTS value should be 2.0. Readers may refer the ffmpeg documentation [34] for further details on usage of this filter. A further improvement on this would be to have a playback with variable speed, with the video slowing down just at the occurrence of the salient event. This can be achieved by trimming the segment and applying different setPTS filters for each and merging them with concat filter. Figure shows the complex filter graph for creating a varying speed replay segment.

5.4.2. Overlay

While showing the close-up replay of the action, the wide angle view is overlaid in the bottom corner. These two views are temporally synchronised giving the viewer a good viewing experience of the salient game situation. The overlay filter of ffmpeg is used for this purpose. An example frame of the overlaid clip is shown in Figure 22.

Figure 22. *Example frame of an overlaid clip for replay section of a salient event segment.*

5.4.3. Fade effects

Fade effects are used at points of transition between salient events. At the end of each salient segment a fade out filter is applied and a fade-in filter is applied at the beginning

of each salient segment. Depending on the users choice, the fade filters may designed be such that they fade to black or fade to white. The ffmpeg fade filter allows the user to specify the type (in or out), duration, start and end frames, number of frames and color (default black). Though applying the fading effect itself is entirely optional, it has been seen that they improve the visual quality of the video as it gives the viewer an indication of time lapse between two events.

5.5. Mixing audio

Audio plays an important role in the summary video. Some users may prefer to have the unaltered recorded original game audio comprising of the commentary and the crowd noise, but some others may prefer to have a background music added to it. In this project, adding background audio was experimented with. The challenge however was an effective playback of audio during the slow replay sections. In these sections, the original audio needed to be muted and only background music was played. Also the volume of the background music was adjusted such that it is lower when combined with original recorded game audio and higher when played alone in slow replay sections.

5.6. Customizing

As discussed earlier there are various parameters that can be configured before the automated summary creation. However sometimes it would be necessary to edit the summary video after it has been created without having the whole process repeated.

Some of the parameters that could be changed in order to customize the summary are - the length of wide angle playback(L1), length of slow replay if present(L2), length of third camera view(L3). Here the replay section may be totally omitted by having the length L2 as 0.

In order to allow customization after the summary creation, an initial summary is created with maximum acceptable values for L1, L2 and L3. The start and end timestamps of each of these three sections are recorded on to a file in xml format.

The custom configuration encoded in XML format is used in one of the two different ways. One in which the xml is embedded into am mp4 container of the final summary video as metadata. This metadata is parsed and read by a custom video player, which first reads the start and end time of each sections and plays them, jumping to various timeframes accordingly. The confidence values of each salient event is also encoded in to the metadata. This would also give the user a choice to view the certain top number of highlights of the game.

The custom configuration information may also be used to edit and re-encode the summary video for it to be played in any standard video player.

An illustration of the usage of customization is shown in Figure 23.

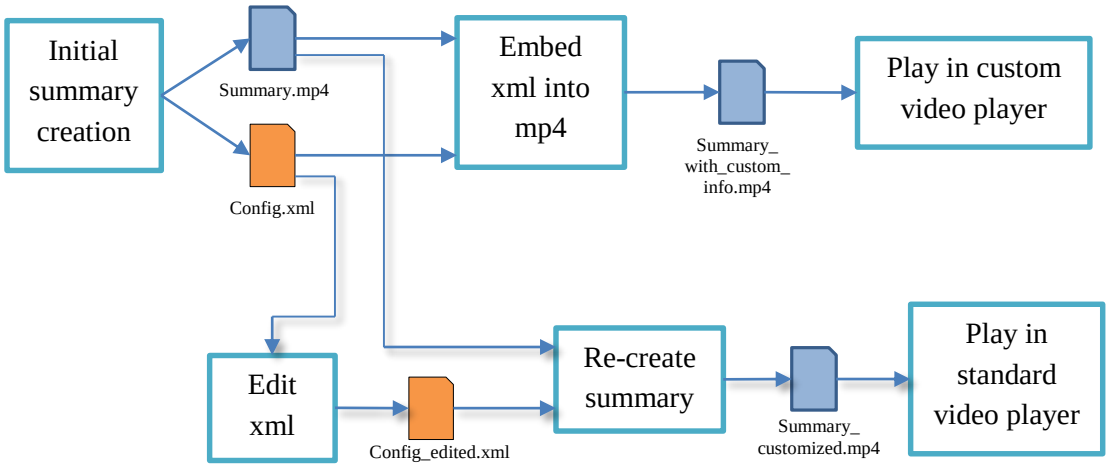

Figure 23. *Illustration of how the customized information stored in a separate xml file can be used in two different ways*

6. EXPERIMENTAL SETUP AND RESULTS

This section describes the dataset and the setup used to collect the dataset, results obtained using various methods and their performance comparison.

6.1. Experimental setup

An actual professional game of basketball was recorded with four different cameras simultaneously. The first camera was setup for wide angle view with fixed zoom covering almost the play area with a little manual panning required to follow the action. The second camera was setup with a close-up view, zoomed in to closely follow the game. The third camera mounted at ground level to capture the game from a different perspective. The fourth camera was un-manned static camera with a fixed view of the right side basket, which was used for automatic saliency detection. In an actual scenario a similar camera would be needed to be setup for the left side basket as well, but in this case as this was just to test the proposed method and prove the concept, only one side was recorded and used.

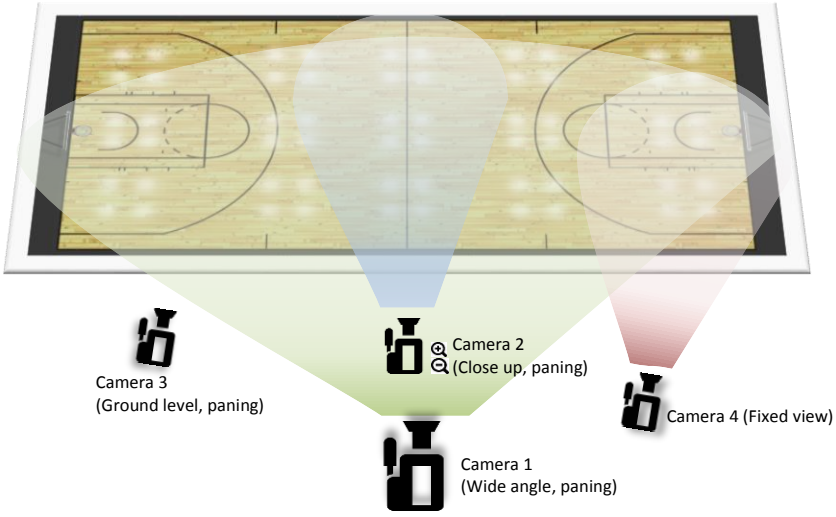

Figure 24. *Camera setup of the game recording for data collection*

The camera set up of the recording event is as illustrated in the Figure 24. The entire recording length was about forty minutes. Even though the dataset set contains recordings of a single game, the videos contained different kinds of scoring attempts, occlusions, lighting changes etc. that provided all kinds of challenges in the detection process and therefore were considered sufficient for proving the concept. Each of these recorded video files were input to the system for automatic saliency detection and summary creation and

experiments were conducted using the prior art method, with motion analysis, background subtraction method and finally background subtraction with effective shape recognition. In the next sections the saliency detection results of each of these methods are compared to the ground truth data and their performances are compared to one another. The metrics for performance measurement are explained in the following section.

6.2. Performance metrics

To define the performance of saliency detection task, commonly used terms in pattern classification used to measure the performance of a classification system are used. These terms are defined below as they apply to this case.

- *True Positives (TP)* refers to the number of events/ timestamps accurately detected as positive salient events.
- *False Positives (FP)* refers to the number of events/ timestamps wrongly detected as salient event.
- *Ground Truth (GT)* refers to the total number of salient timestamps present in the dataset that are identified manually.

Given the above terms, two widely used performance metrics, *Recall* and *Precision* can be derived that represent the overall performance of a given system or classification method and are used for performance comparison. Recall (also known as sensitivity) is defined as the fraction of accurately retrieved salient instances (TP) out of the total number of salient instances (GT). Precision (also known as positive predictive value) is the fraction of accurately retrieved salient instances (TP) out of the total number of instances retrieved (TP+FP).

$$Recall = TP/GT \ , \ Precision = \ TP \ / \ (TP + FP) \tag{6.1}$$

The idea behind a good classification system would be to increase the true positives while reducing the false positives, thereby increasing the recall and precision.

6.3. Results

The experiments were carried out with four methods for saliency detection, with each designed to be an improvement over the prior. Therefore these methods may be treated as different stages in achieving the best performing saliency detection method. The ground truth for this dataset is obtained by manual supervision. Results from each of these methods are compared to the ground truth data and the detection results (salient timestamps) are plotted across the timeline along with ground truth for comparison. As the ground truth is obtained manually, when matching the detected timestamps to the ground truth timestamps, a threshold of up to three seconds is kept as it is subjective to

the user as to what exact moment in time during the scoring attempt could be considered as the most salient one.

The first method is implemented as per the prior art with the trained classifiers for the baskets and the ball with basket detection and ball detection run over each frame as explained in chapter 4.

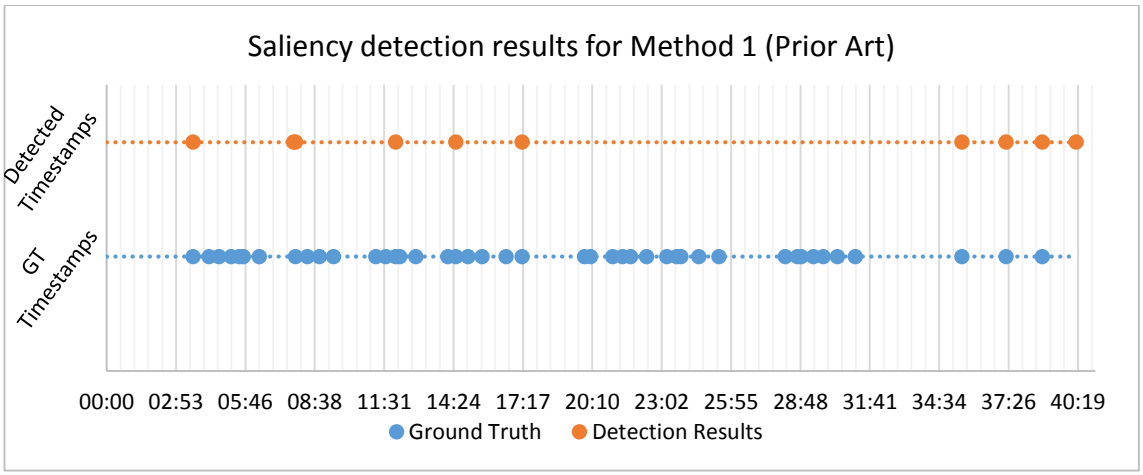

Figure 25. *Salient timestamps detected with the prior art method compared with ground truth.*

The detection results of this method as shown in Figure 25. The overall results from this method contained very few detections compared to the ground truth. Out of 45 ground truth timestamps only 10 were detected with 8 true positives and 2 false positives, thereby giving a recall of 17% and precision of 80%.

The second method applies the techniques of motion analysis in the ROI from a static camera view for detection of ball in the ROI to identify a scoring attempt. This involves frame differencing, thresholding, noise reduction, followed by basic shape recognition as explained in the motion analysis section 4.2 of chapter 4.

The results from the second method are plotted in Figure 26. As it can be seen from this figure, the overall number of detections are much more than the prior method. There were 61 instances detected which contained 45 true positives i.e., all of the ground truth, thereby resulting in a recall of 100%. However the 16 false positives bring down the precision to 73.77%. Most of the false positives observed in this case were due to occlusions of the ROI by audience blocking the view and tiny movements of personal behind the basket region across the line of sight of the camera. The next two methods were developed as an improvement over the prior method with the intention of reducing the false positives to get a good precision score.

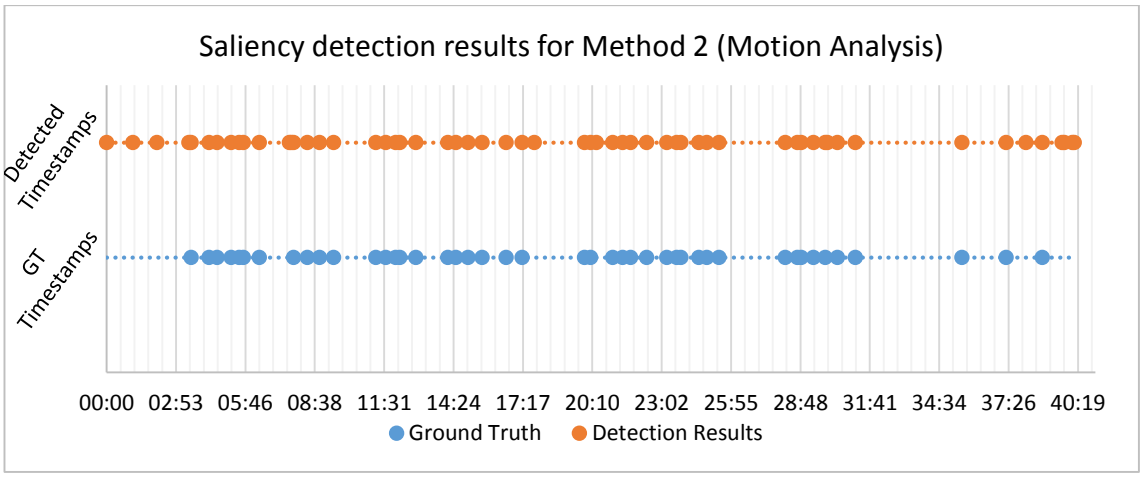

Figure 26. *Salient timestamps detected with the second method using motion analysis compared with ground truth*

The background subtraction technique was applied in the third method to improve ball detection in ROI thereby increasing the accuracy of saliency detection. The background subtraction method using the Adaptive Gaussian Mixture Model seem to be effective in accurate ball detection and reduced the number false positives compared to the prior method. Figure 27 shows the results obtained from this method. The false positives are reduced by applying the background subtraction technique. The detection results include 55 instances with 45 of all ground truth as true positives and 10 false positives, maintaining the recall of 100% and giving the precision of 81.82%.

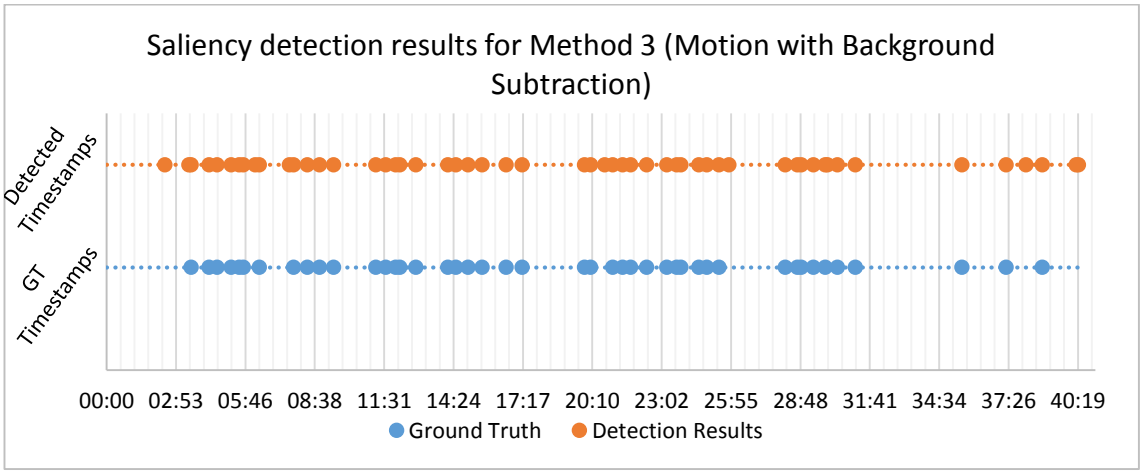

Figure 27. *Salient timestamps detected with the third method using background subtraction for ball detection, compared with the ground truth.*

Even at this stage again most of the false positives arise from occlusions of the ROI by audience which is typically seen during the beginning and towards the end of the game.

In the final method, same steps are used as the prior method using background subtraction with an application of an effective shape recognition technique to more accurately detect

the ball in the ROI. Shape recognition is implemented to recognize the round shape of the ball as explained in sections 2.3.2 of chapter 2 and 4.6 of chapter 4. The results of this

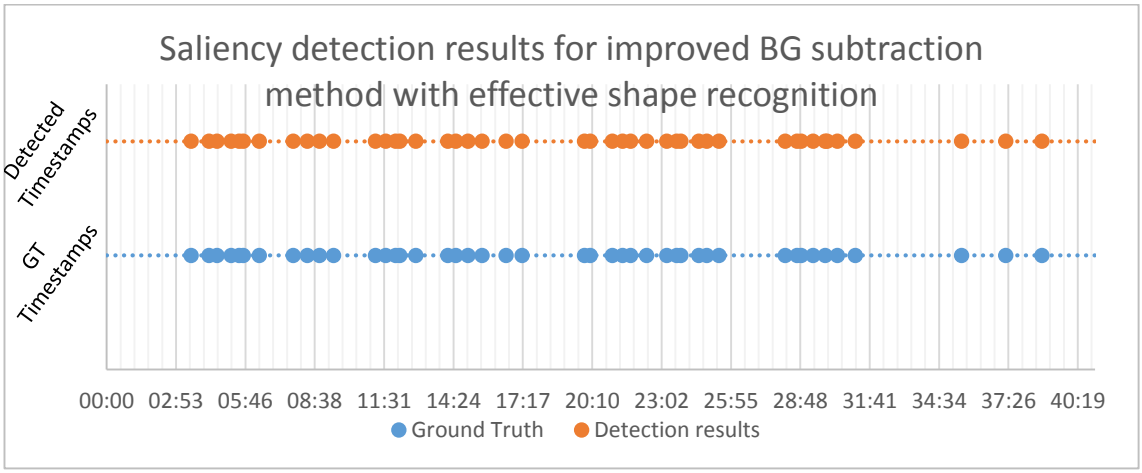

Figure 28. *Salient timestamps detected with the fourth method using background subtraction with effective shape recognition, compared with the ground truth.*

method seem to greatly reduce the false positives rate. Figure 28 shows the detection results of this final method. The final improved saliency detection method produces results with all 45 ground truth instances detected as true positives and only 1 false positive result, thereby giving a recall of 100% and improving the precision to 97.82%.

The overall comparison of the performances of these methods are shown in Figure 29 and Figure 30.

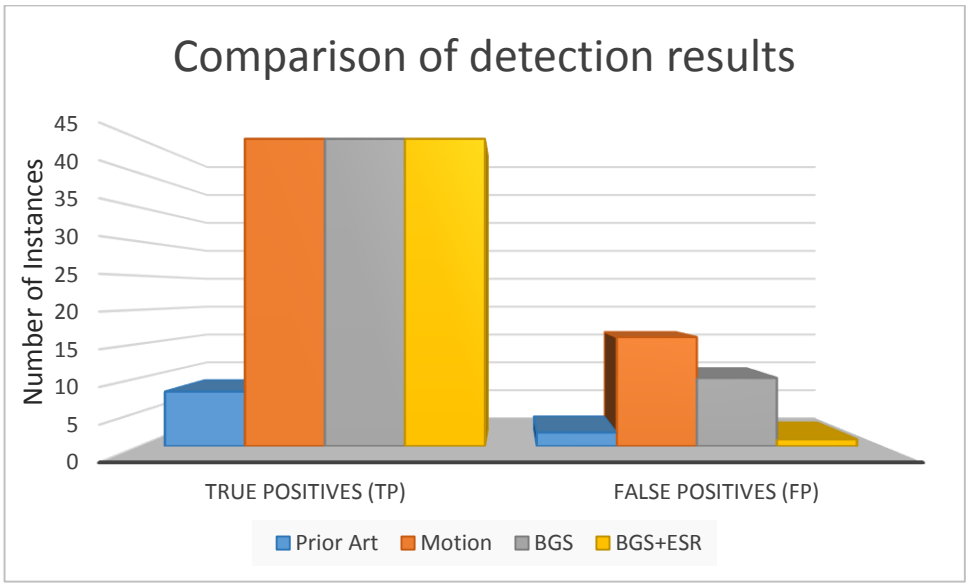

Figure 29. *Comparison of true positives and false positives of the detection results of each of the four methods.*

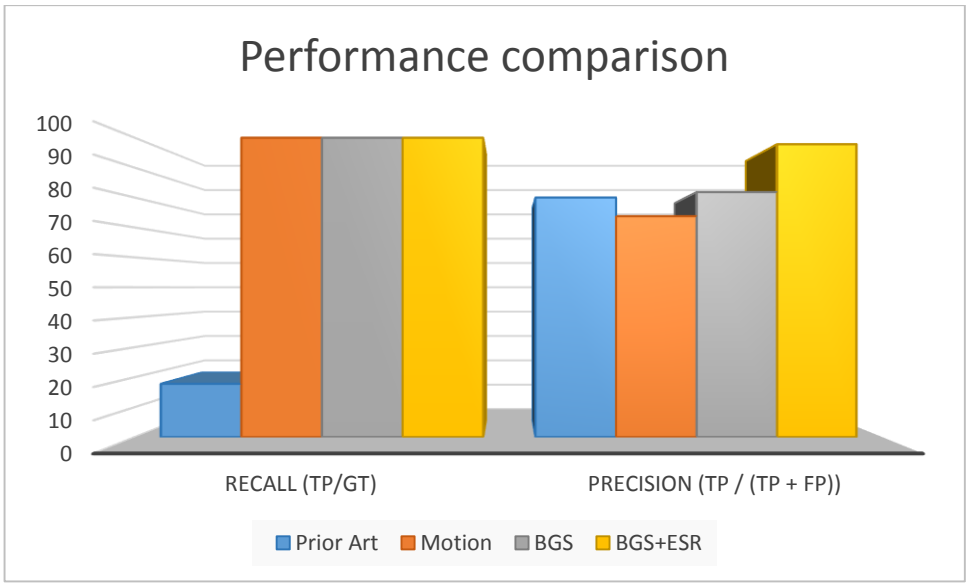

Figure 30. *Performance comparison of the four methods with the last method using background subtraction with effective shape recognition producing 100% recall and 97.82% precision.*

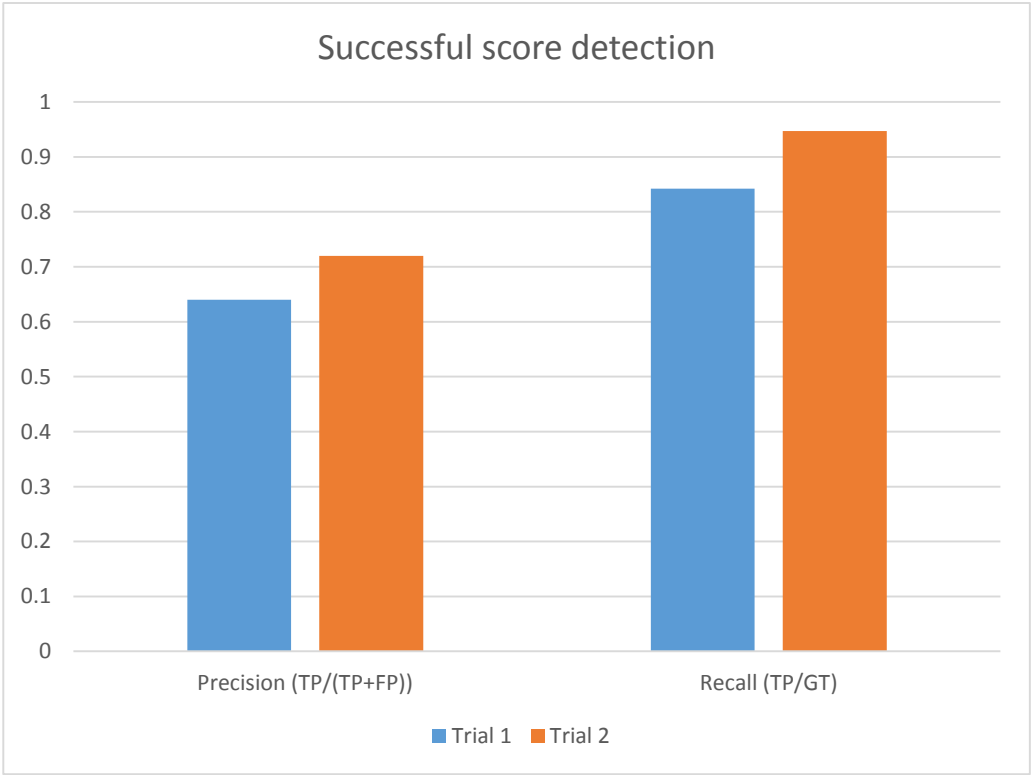

Figure 31. *Results of successful score detection. The second trial with counter measures implementation shows significant improvement over the first one.*

The successful score detection was also implemented and tested as discussed in section 4.8 chapter 4. The accuracy of successful score detection was measured using the same

performance metrics as used in saliency detection. The results of two trials are discussed here. The second trial is a slight improvement over the first one in that it implements the counter measures as discussed in section 4.10 of chapter 4.

The dataset contained a total of 19 successful scoring attempts treated as ground truth. The first trial resulted in 25 instances out of which 16 instances were detected accurately (TP) with 9 false positives, thereby giving a recall of 84.21% and a precision of 64%. After improving the algorithm with the counter measures, the second trial improved the performance resulting in 25 instances out of which 18 were true positives and only 1 was a false positive result, thereby achieving better performance with recall of 84.21% and precision of 94.73%. The results are summarized in the chart in Figure 31.

7. CONCLUSION AND FUTURE WORK

This chapter concludes the thesis with some concluding remarks. Some ideas for further improvement of the system and its possible applications for future work are also discussed.

7.1. Conclusion

In this thesis, a fully automated way of creating summary videos for basketball game was proposed. The work done focused on efficient detection of salient events and automated summary creation process. Various methods, techniques and algorithms of machine learning and image processing were studied and experimented with.

From the experiments and results it is inferred that machine learning techniques such as trained classifiers are useful to some extent. Other traditional image processing techniques and motion analysis when combined and applied seemed very effective in solving the saliency detection problem in this case and produced good results.

A prior art method of saliency detection that uses object detection by machine learning techniques was studied and implemented. This system was further improved by analyzing motion in a specific region of interest and a new systems that uses a static camera input was proposed. The results produced by motion analysis method gave a higher recall but because of large number of false detections the precision was low. Further improvement of the system by the method of background subtraction proved to be very effective in eliminating many false positives and thus increasing the performance up to 81.81% precision while maintaining the 100% recall. In the final stage of improvement more effective shape recognition technique was used to bring down the false detection rate, which improved the performance with 100% recall and 97.82% precision.

Along with detecting scoring attempts, the implementation was further extended to also detect successful attempts. The final method produces considerably good performance of 84.21% recall and 94.73% precision.

The thesis also presented the method to develop summary video from various camera feeds and salient timestamps detected in the saliency detection stage. These videos are clipped, processed and merged automatically. A system for customizing the summary based on the length of each segment, confidence values and choosing of number of highlight segments was also proposed.

Thus, a good performing fully automated and re-customizable system for creating a summary video for basketball games was developed.

7.2. Possible future work

The concept of highlighting basketball game videos to contain the most salient sections of the game in an automated process has been demonstrated in this thesis project. The only areas requiring user intervention are setting up of the cameras and maybe manually handling one or two cameras e.g., wide angle panning and close-ups.

In future the system may be further developed to fully automate the process of recording as well. One approach to achieve this is to have real time processing to analyze motion in the wide angle video feed and control camera movements to follow the actions of the game. The camera may be mounted on a stepper motor board (e.g. Arduino) that is controlled by a program running on a computer. The program could analyze the video feed in real time and send signals to the board to rotate in appropriate direction. The decision making logic to follow the action or in other words follow the ball could be actually very simple. As motion of players running with and around the ball would create huge motion contours and bounding box of top 3 or 4 motion contours would result in the most salient window or region (containing the ball and players around it) in the video frame.

The same idea of the salient region could also be applied to digitally zoom in to that window and clip and mix the videos with different views. This could be very effective in cases where the recording is in ultra-high definition e.g. a static camera with 4K recording capability that covers the entire playing area.

The saliency detection and video mixing process could also be parallelized to improve the speed of execution. Parallelizing the saliency detection process would involve splitting the input video into different parts and then running saliency detection on each part (which could run in parallel in a distributed computing environment) and then merging the results. The video clipping and mixing process also has a lot of scope for parallelization. As multiple video inputs could be processed at the same time with parallel running worker threads, each handling one video at a time. Clipping salient sections and merging of clipped sections into a single highlight segment could run in parallel. Thus multiple highlight segments could be created simultaneously and then merged together.

The final and probably obvious improvement of the system would be to make it work for other similar sports such as soccer, ice-hockey etc. The basket classifier may be replaced with a goal-post classifier. The principle and method of saliency detection and summary creation would still remain the same with some tweaking of parameters required.

REFERENCES

[1] M. Pietikäinen, Local Binary Patterns. Scholarpedia, Vol. 5, pp. 9775, 2010

[2] T. Ojala, M. Pietikäinen, T. Mäenpää, “Multiresolution grayscale and rotation invariant texture classification with local binary patterns”, IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), Vol. 24, 2002.

[3] S. Liao, X. Zhu, Z. Lei, L. Zhang, S. Z. Li, “Learning Multi-scale Block Local Binary Patterns for Face Recognition”, In Proc. of International Conference on Biometrics (ICB), pp. 828-837, 2007.

[4] Y. -H. Lei, Y. -Y. Chen, B. -C. Chen, L. Lida, W. H. Hsu, “Where Is Who: Large scale Photo Retrieval by Facial Attributes and Canvas Layout”, In Proc. of the 35th International ACM SIGR Conference on Research and Development in Information Retrieval, Portland, Oregon, USA, 2012.

[5] H. Jin, Q. Liu, H. Lu, and X. Tong, “Face detection using improved LBP under Bayesian framework”, In Proc. Third International Conference on Image and Graphics (ICIG), Hong Kong, China, pp. 306-309, 2004.

[6] T. Ojala, M. Pietikäinen, “Unsupervised Texture Segmentation Using Feature Distributions”, Pattern Recognition, Vol. 32, pp. 477-486, 1999.

[7] L. Zhang, R. Chu, S. Xiang, S. Liao, S. Li, “Face Detection Based on Multi-Block LBP Representation”, In Proc. International Conference on Biometrics (ICB), pp. 11-18, 2007.

[8] P. Viola, M. Jones, "Rapid object detection using a boosted cascade of simple features", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 511–518, 2001.

[9] C. P. Papageorgiou, M. Oren, T. Poggio, "A general framework for object detection", International Conference on Computer Vision, 1998.

[10] P. Viola, M. Jones, "Robust Real-time object detection", International Journal of Computer Vision, 2001.

[11] M. Heikkiläa, M. Pietikäinena, C. Schmidb, "Description of interest regions with local binary patterns", Pattern Recognition, Vol. 42, 2009.

[12] Y. Freund, R. E. Schapire, "A Short Introduction to Boosting", Journal of Japanese Society for Artificial Intelligence, Vol. 4, pp. 771-780, 1999.

[13] C. Ma, T. Tan, Q. Yang, "Cascade Boosting LBP Feature Based Classifiers for Face Recognition", In Proc. 3rd International Conference on Intelligent System and Knowledge Engineering, 2008.

[14] R. Lienhart, A. Kuranov, V. Pisarevsky, “Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection”, In Proc. of the 25th DAGM-Symposium, Magdeburg, Germany, pp. 297– 304, 2003.

[15] P. KadewTraKuPong, R. Bowden, “An improved adaptive background mixture model for real-time tracking with shadow detection”, In Proc. of the 2nd European Workshop on Advanced Video-Based Surveillance Systems, 2001.

[16] Z. Zivkovic, “Improved adaptive Gaussian mixture model for background subtraction”, International Conference Pattern Recognition, UK, August, 2004.

[17] Z. Zivkovic, F. van der Heijden, “Efficient Adaptive Density Estimation per Image Pixel for the Task of Background Subtraction”, Pattern Recognition Letters, Vol. 27, no. 7, pp. 773-780, 2006.

[18] Z. Zivkovic, F. van der Heijden, “Recursive unsupervised learning of finite mixture models”, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.26, no.5, pp. 651-656, 2004.

[19] P. Shrestha, M. Barbieri, and H. Weda, “Synchronization of multi-camera video recordings based on audio”, In Proc. of the 15th International Conference on Multimedia, pp. 545-548, ACM, 2007.

[20] C. Ridder, O. Munkelt, Ha. Kirchner. “Adaptive Background Estimation and Foreground Detection using Kalman-Filtering,” In Proc. of International Conference on recent Advances in Mechatronics, ICRAM’95, 193-199, 1995.

[21] D. Koller, J. Weber, T. Huang, J. Malik, G. Ogasawara, B. Rao, S. Russel. “Towards robust automatic traffic scene analysis in real-time.” In Proc. of the International Conference on Pattern Recognition, Israel, November 1994.

[22] C. Stauffer, W. Grimson, “Adaptive background mixture models for real-time tracking”, In Proc. Computer Vision and Pattern Recognition (CVPR), pp. 246-252, 1999.

[23] F. Dellaert. “The expectation maximization algorithm”, 2002. Available: http://www.cc.gatech.edu/˜dellaert/em-paper.pdf

[24] A. P. Dempster, N. M. Laird, D. B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”, Journal of the Royal Statistical Society B, Vol. 39, pp. 1–22, 1977.

[25] H. Hartley. “Maximum likelihood estimation from incomplete data”, Biometrics, Vol. 14, pp.174–194, 1958.

[26] T. P. Minka. “Expectation-maximization as lower bound maximization”, 1998. Available: http://citeseer.nj.nec.com/minka98expectationmaximization.html

[27] R. M. Neal, G. E. Hinton. “A new view of the EM algorithm that justifies incremental, sparse and other variants”. In M. I. Jordan, Learning in Graphical Models, pp. 355–368, 1998.

[28] J. Rennie. “A short tutorial on using expectation-maximization with mixture models”, 2004. Available:

http://www.ai.mit.edu/people/jrennie/writing/mixtureEM.pdf.

[29] Y. Weiss. “Motion segmentation using EM – a short tutorial”, 1997. Available: http://www.cs.huji.ac.il/~yweiss/emTutorial.pdf

[30] W. H. Wong, W. C. Siu, “Improved digital filter structure for fast moment computation”, IEE Proc. Vision, Image Signal Processing. Vol. 46, pp. 73–79, 1999.

[31] S. A. Dudani, K. J. Breeding, R. B. Mcghee, “Aircraft identification by moment invariants”, IEEE Trans. on Computers. Vol. 26, pp. 39–46, 1977.

[32] S. X. Liao, M. Pawlak, “On the accuracy of Zernike moments for image analysis”, IEEE Trans. on Pattern Analysis and Machine Intelligence. Vol. 20, pp. 1358–1364, 1998.

[33] M. Hu, “Visual pattern recognition by moment invariants”. IRE Trans. on Information Theory. Vol. 8, pp. 179–187, 1962.

[34] FFmpeg official documentation, Available: https://www.ffmpeg.org/documentation.html

[35] G. Bradski, “The OpenCV Library”, Dr. Dobb’s Journal of Software Tools, 2000. Available: http://www.drdobbs.com/open-source/the-opencv-library/184404319

[36] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves", Computer Graphics and Image Processing, Vol. 1, no. 3, pp. 244–256, 1972.

[37] D. Douglas, T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer Vol. 10, no. 2, pp. 112–122, 1973.

[38] J. Hershberger, J. Snoeyink, "Speeding Up the Douglas–Peucker Line-Simplification Algorithm", In Proc. of the 5th Symposium on Data Handling, pp. 134–143, 1992

[39] D. Soendoro, I. Supriana, "Traffic Sign Recognition with Color-based Method, Shape-arc Estimation and SVM", International Conference on Electrical Engineering and Informatics, Bandung, Indonesia, 2011.

[40] A. Salhi, B. Minaoui, M. Fakir, "Robust Automatic Traffic Signs Recognition Using Fast Polygonal Approximation of Digital Curves and Neural Network", International Journal of Advanced Computer Science and Applications, Special Issue on Advances in Vehicular Ad Hoc Networking and Applications, 2014.

[41] S. Wu, A. da Silva, M. Marquez, "The Douglas-peucker algorithm: sufficiency conditions for non-self-intersections", Journal of the Brazilian Computer Society, Vol. 9, no. 3, pp. 67-84. 2004.

[42] F. Cricri, “Mutlimodal Analysis of Mobile Videos”, Tampere University of Technology, Publication 1207, 2014.