AUTOMATIC EYE TRACKING AND INTERMEDIATE VIEW RECONSTRUCTION FOR 3D IMAGING SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

YUSUF BEDİZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. İsmet Erkmen
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Gözde Bozdağı Akar
Supervisor

Examining Committee Members

Prof. Dr. Uğur Halıcı (METU,EE) _____

Assoc. Prof. Dr. Gözde Bozdağı Akar (METU,EE) _____

Assoc. Prof. Dr. Aydın Alatan (METU,EE) _____

Assist. Prof. Dr. İlkay Ulusoy (METU,EE) _____

Dr. Adem Mülayim (MILSOFT INC.) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Yusuf Bediz

Signature :

ABSTRACT

AUTOMATIC EYE TRACKING AND INTERMEDIATE VIEW RECONSTRUCTION FOR 3D IMAGING SYSTEMS

Bediz, Yusuf
M.S., Department of Electrical and Electronics Engineering
Supervisor : Assoc. Prof. Dr. Gözde Bozdağı Akar

July 2006, 76 pages

In recent years, the utilization of 3D display systems became popular in many application areas. One of the most important issues in the utilization of these systems is to render the correct view to the observer based on his/her position. In this thesis, we propose and implement a single user view rendering system for autostereoscopic/stereoscopic displays. The system can easily be installed on a standard PC together with an autostereoscopic display or stereoscopic glasses (shutter, polarized, pulfrich, and anaglyph) with appropriate video card. Proposed system composes of three main blocks: view point detection, view point tracking and intermediate view reconstruction. Haar object detection method, which is based on boosted cascade of simple feature classifiers, is utilized as the view point detection method. After detection, feature points are found on the detected region and accordingly they are fed to the feature tracker. View point of the observer is calculated by using the tracked position of the observer on the image. Correct stereoscopic view is, then, rendered on the display. A 3D warping-based method is utilized in the system as the intermediate view reconstruction method. System is implemented on a computer with Pentium IV 3.0 GHz processor using E-D 3D shutter glasses and Creative NX Webcam.

Keywords: View rendering, View point detection, view point tracking and intermediate view reconstruction.

ÖZ

3 BOYUTLU GÖRÜNTÜLEME SİSTEMLERİ İÇİN OTOMATİK GÖZ TAKİBİ VE ARA GÖRÜNTÜ OLUŞTURULMASI

Bediz, Yusuf
Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Gözde Bozdağı Akar

Temmuz 2006, 76 sayfa

Son yıllarda 3 boyutlu görüntüleme sistemlerin kullanımı bir çok uygulama alanlında popülerlik kazanmıştır. Bu sistemlerin kullanımı sırasındaki en önemli sorunlardan biri kullanıcının posizyonuna göre doğru görüntünün oluşturulup ekranda gösterilebilmesidir. Bu tezde tek kullanıcılı bir görüntüleme sistemi önerilmekte ve gerçeklenmektedir. Sistem kolaylıkla standard bir bilgisayar üzerine oto-stereoskopik monitör veya stereoskopik gözlük (kepenk, polarize, pulfrik, ve anaglif) kullanılarak kurulabilir. Önerilen sistem üç ana parçadan oluşmaktadır: bakış açısı bulma, bakış açısı takip etme ve ara görüntü oluşturma. Basit öznitelik sınıflandırıcıların desteklenmiş kademeli dizilerine dayanan Haar nesne bulma yöntemi sistemde bakış noktası bulma yöntemi olarak kullanılmıştır. Bu method kullanılarak iki sınıflandırıcı eğitilmiştir. Bakış noktası bulma işleminden sonra bulunan bölge içerisindeki öznitelikler bulunup öznitelik takipçisine verilmektedir. Kullanıcının bakış açısı görüntü üzerinde takip edilen bakış noktası kullanılarak hesaplanmaktadır. Bakış açısı bulunduktan sonra doğru stereoskopik görüntü oluşturulup ekrana çizilmektedir. 3B eğriltmeye dayalı bir method, ara görüntü oluşturma metodu olarak kullanılmıştır. Sistem

Pentium IV 3.0 GHz işlemciye sahip bir bilgisayar üzerinde E-D 3D kepenk gözlükler ve Creative NX Webcam kullanılarak gerçeklenmiştir.

Anahtar Kelimeler: Görüntüleme sistemi, bakış noktası bulma, bakış noktası takip etme, ara görüntü oluşturulması.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my supervisor Assoc. Prof. Dr. Gözde Bozdağı Akar for her guidance, advice, criticism, encouragements and insight throughout the research.

This thesis work is supported by "3DTV" project which is funded by European commission 6th framework.

Thanks go to Multimedia Research Group (MMRG) members for their technical support. Besides, I spent great time in all our activities.

I would like to express my thanks to my friends Ali Ünal and Adem Mülayim. Their support and assistance was invaluable.

I would like to thank my dear friends Erol Aran and Çağdaş Altın. It is a great feeling to know that somebody cares about you and will be by your side in every situation.

Finally, I would like to thank my family for their understanding, support and patience; especially to my father and mother.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

2D	2 Dimensional
3D	3 Dimensional
API	Application Programming Interface
FERET	Facial Recognition Technology
IR	Infrared
OpenCV	Open Computer Vision
OpenGL	Open Graphics Library
PC	Personel Computer
RSAT	Rotated Summed Area Table
SAT	Summed Area Table
SVM	Support Vector Machine
USB	Universal Serial Bus

CHAPTER 1

INTRODUCTION

1.1 General

During the last years, the utilization of 3D Display Systems considerably increased in applications of education, entertainment and presentation. The one which has the most widespread usage and is the most economical system among 3D Display Systems is the stereoscopic eyeglasses. However the utilization of brand new and popular autostereoscopic display systems is also growing with its price getting cheaper. Autostereoscopic displays make the 3D viewing experience more pleasant by eliminating the necessity of glasses to be used. They can be categorized into two main groups: Two-view and multi-view autostereoscopic displays. Two-view autostereoscopic displays render one stereoscopic image at maximum resolution of the display. However observers do not perceive true 3D sense as they move in front of the display for the reason that they see the same image from all positions. This problem is also valid for stereoscopic eye glasses Multi-view autostereoscopic displays present a large number of views so that as the observer moves, a different pair of the views is seen from each new position. On the other hand, the resolution of the display split between the multiple views and image quality falls on these displays. A solution to this problem is to utilize a two-view display or a stereoscopic eyeglass with head tracking. In this way, the correct view can be rendered to the observer at full display resolution based on his/her position while observer moves freely in front of the display. Head tracking can be done in an active way that the observer wears some special

sensors, such as infrared sensors or reflectors, ultrasonic wave receivers and electromagnetic wave sensor. However this kind of methods is uncomfortable and inconvenient. Therefore, video based methods are preferable for tracking observers in a passive manner.

Recently, researches have developed video-based trackers for autostereoscopic displays [1] [2]. In [1] an eye tracking system using two webcam has been proposed. The eyes are detected by fast pattern recognition. They additionally use color information in the images. However, this makes their system very sensitive to lighting conditions. Another eye tracking system has been proposed in [2] where the face of the observer is detected by using multiple eigenspaces of various lighting conditions and then the eyes are located in the obtained face by a convolution based method. Tracking is done with fast block matching in this system.

Besides eye trackers, complete autostereoscopic displays systems with observer tracking has also been proposed. Fraunhofer Heinrich-Hertz-Institut (HHI) developed a system with a special head-tracking lenticular-screen [3]. This screen is mechanically adjusted according to the position of the observer while the observer is tracked. Current products also started to include eye tracking systems like SeeReal C-i 3D Display [4]. However this further increases the price.

1.2 Scope of the thesis

This thesis deals with the problem of developing a real-time view rendering system that detects and tracks the position of the observer and renders the correct view to the observer based on his/her position. System is aimed to work with both autostereoscopic displays and stereoscopic glasses and it is restricted to a single user.

Main building blocks of the system are defined as view point detection, view point tracking and intermediate view reconstruction. Implementation of these blocks are done with Haar object detection method, pyramidal implementation of Lucas Kanade feature tracker and a 3D warping based method respectively. System is implemented on a computer with Pentium IV 3.0 GHz processor using E-D 3D shutter glasses and Creative NX Webcam.

1.3 Outline of the dissertation

In Chapter 2, Haar object detection method is described. It is the method that is utilized as the implementation of view point detection component in the proposed system.

Pyramidal implementation of the Lucas-Kanade Feature Tracker which is utilized as the view point tracker is described in Chapter 3.

The intermediate view reconstruction method used in the system implementation can be found in Chapter 4. This chapter also includes some background information about camera geometry which is nessesary to understand the explained method.

Chapter 5 explains the proposed system and gives detailed information about the implemantation of each block in the system. Results of the blocks and comments on the results are also given in this chapter.

Finally, Chapter 6 gives the summary of the thesis and concluding remarks. Some future work is also suggested in this chapter.

CHAPTER 2

VIEW POINT DETECTION

For 3D display systems, tracking of the observer's view point is necessary to render the correct stereoscopic view according to the observer position. View point is the eye region of the observers for autostereoscopic displays or the eyeglasses of the observer for the stereoscopic glasses (shutter, polarized, pulfrich and anaglyph).

Detection of the view point is the first step in tracking of the view point and very important for the system performance. In the literature, a number of techniques are available for the eye region detection. These techniques are thoroughly described in section 2.1. However, there is no specific algorithm for the detection of stereoscopic glasses. Generic object detection methods can, thus, be used for the detection of stereoscopic glasses.

Haar object detection method is used in the proposed system for the view point detection. It is a generic object detection framework and can be used both for detection of eyes and stereoscopic glasses. Detailed description of the Haar object detection method is given in section 2.2

2.1 Eye Region Detection

Eye region detection techniques can be categorized into two main groups: Active Infrared (IR) illumination based methods and image based passive methods.

Methods mentioned in the first group exploit the spectral properties of pupil under near IR illumination. It is usually called the bright pupil effect of eyes (Eyes look bright when illuminated by an IR light source). In the second group, detection is performed on a priory detected frontal face image. These methods are passive methods and do not utilize an external light source or a special equipment.

2.1.1 Active IR Illumination Based Methods

Active IR based methods are widely used in eye detection and tracking applications. There has been a lot of work using this technique and there are some commercial eye tracking systems such as produced by ISCAN Incorporated, LC Technologies and Applied Science Laboratories (ASL) [5], [6], [7], [8]. These methods use the special bright pupil effect, which can be obtained in daylight by: An IR light source, illuminating the scene, is located near to the camera and a visible light filter is attached to the camera lens. Under IR illumination a very high contrast between eyes (pupils) and the rest of the face can be obtained. Using this contrast, eyes can easily be detected. Although Active IR based methods enable the easy detection of eyes, these methods have some drawbacks. Rapid, large and fast head movements of the observer may cause troubles. The use of thick eye glasses may turn out to be also a problem in that they disturb the infrared light and cause weak pupil appearance. The performance of these methods depends on lighting conditions and the pupil size. Lighting conditions should be stable and the observer should be close to the camera. In order to overcome these drawbacks people have combined the IR illumination with other methods. Zhu proposed an IR illumination based method, which works under variable realistic lighting conditions and is based on combining the bright-pupil effect resulted from IR light and the conventional appearance-based object recognition technique [6]. De Liefde overcame the problem occurred by rapid head movement by the use of probabilistic principal component analysis based classifier in the results of IR illumination [7]. The method proposed by Ramadan

also eliminated the effect of the dynamics of head movements. Proposed method uses IR illumination with active deformable models [5].

2.1.2 Image Based Passive Methods

Image based methods become more popular with the development of the video and image analysis technology. In these methods a frontal face is first detected and then eyes are located inside the detected region. Eye detection can be performed by using different methods. These methods can be classified into three categories: template based methods [9], [10], appearance based methods [11], [12] and feature based methods [13], [14].

In the template based methods [9] [10], first a generic eye model, based on the eye shape, is designed. Then template matching is utilized to search for the eyes on the face. Although these methods can detect eyes accurately, they are normally time-consuming because they match the whole face with an eye template pixel by pixel to improve the accuracy.

The appearance based methods [11] [12] use photometric properties to detect eyes. They usually need large amount of training data. Data should include the eyes of different people under different face orientations and illumination conditions. They use these data to train a classifier like a neural network or a support vector machine. Consequently detection procedure becomes a classification procedure of eye and non-eye regions.

In feature based methods [13] [14], some distinctive features around eyes are identified. These features use the characteristics of the eyes such as edge and intensity of iris, the color distribution of the sclera and the flesh. These methods are usually efficient but the disadvantage is their lack of performance. The accuracy for the detection of the images that have low contrast can not be ensured

2.2 Haar Object Detection

Haar Object Detection Method is chosen as the view point detection method in the system. This method describes a framework for robust and extremely rapid object detection. It has been first proposed by Viola [15] and improved by Lienhart [16]. They have used their framework to train a frontal face detector. Based on their results, the detector is the fastest frontal face detector and its performance is comparable to much slower and more complex detectors [18].

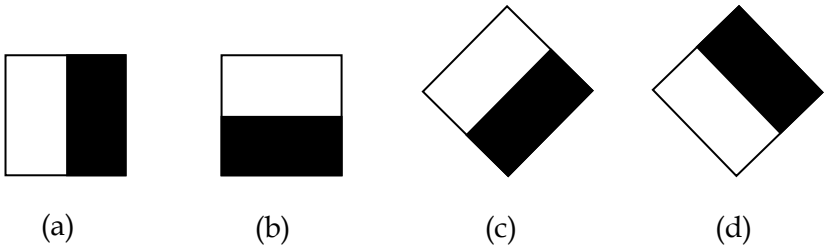
In this method, first a classifier is trained with hundreds of positive and negative samples. Positive samples are sample views of a particular object and negative samples are any other arbitrary images of those which do not contain that object. After a classifier is trained, it can be used to detect an object in a region (same size as the positive samples) on the input image. If the region contains the object, classifier outputs '1' or '0' otherwise. To search the object in the whole image, one can move the search window across the image and check for the object using the classifier. Classifier can be easily resized. Search procedure can be repeated with the classifier of different sizes to find the objects of unknown size.
Classifier is actually a cascade of simple feature classifiers. These simple feature classifiers are weak classifiers. They are built by haar-like features through boosting technique called Adaboost. However, feature calculation is very fast. Cascade structure of the final classifier increase accuracy and decrease the processing time by discarding the background regions quickly.

2.2.1 Features

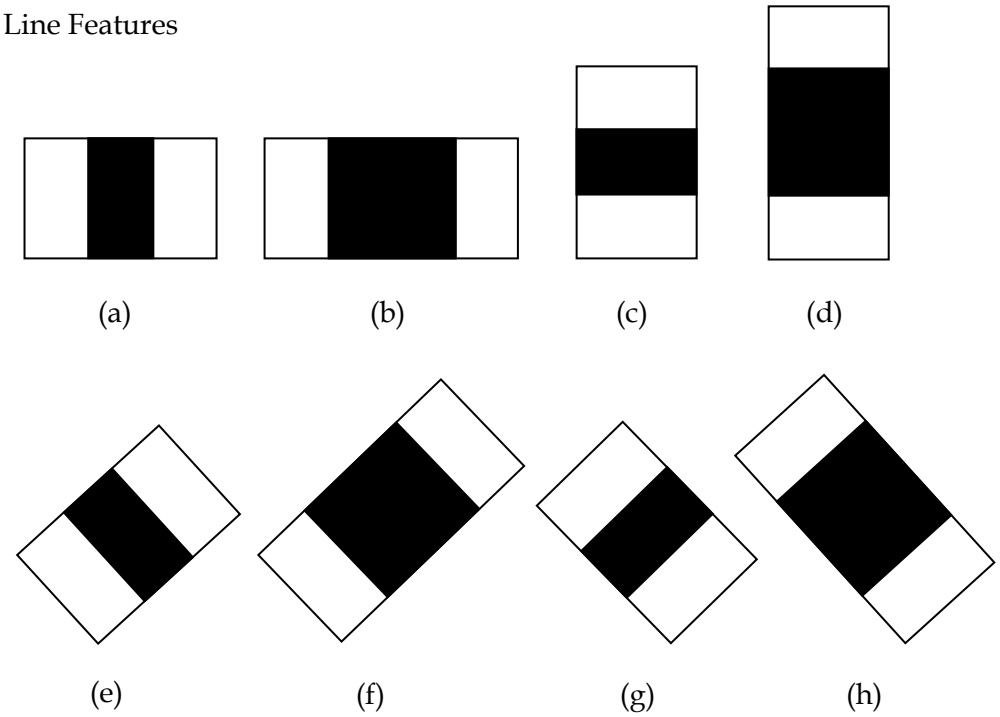
Haar Object Detection method uses very simple features. These features are inspired by the over-complete reminiscent of Haar basis functions used by Papageorgiou et al [17]. There are two most important reasons for using features instead of raw pixel values. The first reason is that features can encode information about the domain that is difficult to be recognized from a raw and finite set of input data. Second one is that features used by detection method can

be computed much faster than a pixel based method. By the help of two auxiliary images of SAT and RSAT (see Section 2.2.1.1), features can be computed with at most 8 table lookups at any position and scale. Haar Object Detection method uses feature prototypes to produce these features. There are 14 feature prototypes (see Figure 2.1) and these prototypes are grouped into 3 categories: Edge features, line features and center-surround features.

1. Edge Features



2. Line Features



3. Center-surround features

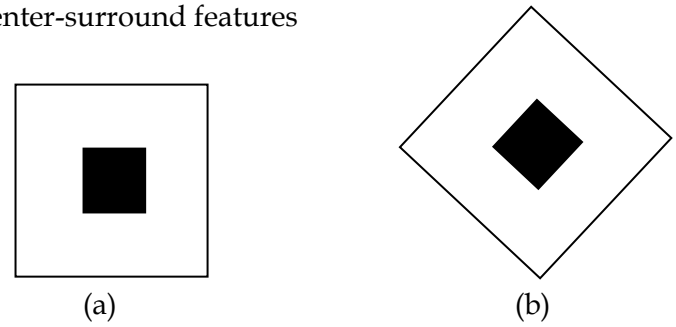


Figure 2.1: Feature prototypes.

The features are produced out of these prototypes and defined in a window of *W* x *H* pixels in the image. A specific feature is produced from a prototype according to its shape, position and scale within the window.

All of the features are the combination of two rectangles and can be defined as the weighted sum of two rectangles. Features are calculated by subtracting the sum of the pixels within the white rectangle from the sum of the pixels within the black rectangle.

The rectangles are specified in the window by the tuple $r = (x, y, w, h, \alpha)$ with $0 \leq x$, $x+w \leq W$, $0 \leq y$, $y+h \leq H$, $w > 0$ and $h > 0$. $\alpha \in \{ 0°, 45° \}$ for the upright and 45° rotated rectangles (see Figure 2.2).

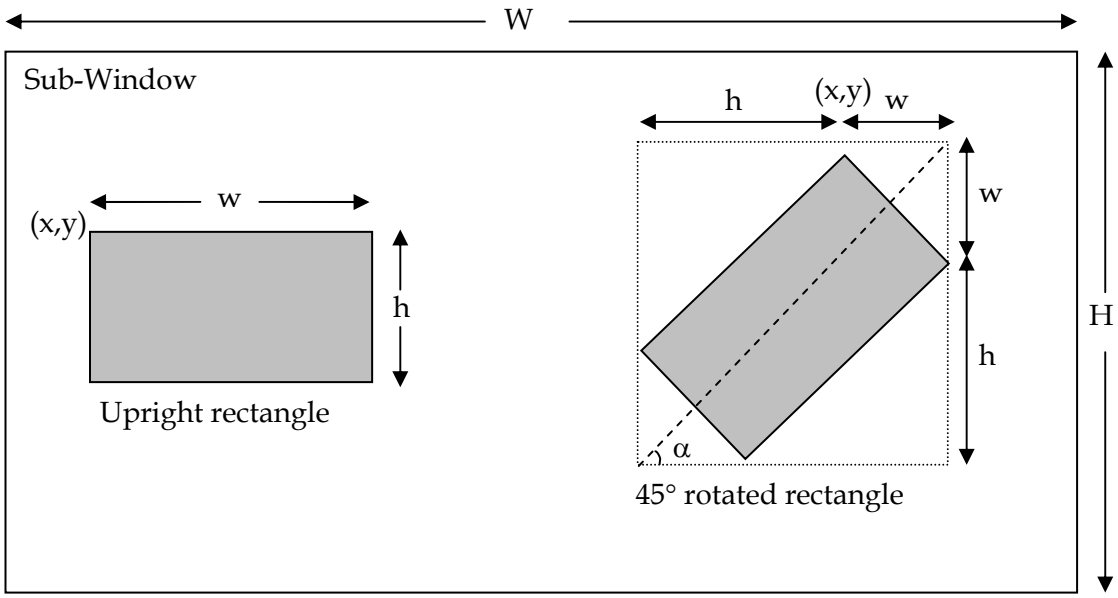


Figure 2.2: Example of an upright and 45° rotated rectangle.

The sum of the pixels in a rectangle is represented by *RecSum(r)*. Then the features can be shown as weighted sum of different rectangles.

$$feature_i = w_0 \cdot RecSum(r_0) + w_1 \cdot RecSum(r_1) \quad (2.1)$$

The weights have opposite signs, and are used to compensate for the difference of the two rectangles in area and size. Hence, we can set $w_0 = -1$ and $w_1 = Area(r_0) / Area(r_1)$. For instance, a feature produced from line feature prototype (2.b) (see Figure 2.1) with total height of 4 and width of 12 at the top left corner (5, 6) can be written as:

$$feature_i = -1 \cdot RecSum(5,6,12,4,0°) + 2 \cdot RecSum(9,6,6,4,0°) \quad (2.2)$$

These rectangular features are primitive relative to other alternative features like steerable filters [19], [20]. However, the number of features in a window is quite large and calculation of these features is extremely fast relative to other alternatives. These advantages compensate for their limited flexibility.

2.2.1.1 Number of Features

We can generate a rich, over-complete set of features by using feature prototypes by translating and scaling in horizontal and vertical directions independently. The number of features derived from a prototype can be calculated as follows. Let *W* and *H* be the width and height of the sub-window respectively (see Figure 2.2) and $X = \lfloor W / w \rfloor$ and $Y = \lfloor H / h \rfloor$ be the maximum scaling factors in *x* and *y* direction. An upright feature of size *w* x *h* then generates

$$XY(W + 1 - w\frac{X+1}{2})(H + 1 - h\frac{Y+1}{2}) \quad (2.3)$$

features for an image of size W x H, while 45^0 rotated features generates

$$XY(W + 1 - z\frac{X+1}{2})(H + 1 - z\frac{Y+1}{2}) \text{ with } z = w + h \quad (2.4)$$

features. For a search window with resolution 24 x 24 pixels, the total set of features is very large, as being 117.941. This is far larger than the number of pixels within the window.

2.2.1.2 Fast Feature Computation

Haar Object Detection method introduces two intermediate image representation "*Summed Area Table* (SAT)" and "*Rotated Summed Area Table* (RSAT)". SAT and RSAT are calculated once from the original image utilizing a few number of operations per pixel. Then, all of the features can be calculated very fast within a fixed and short time period at anywhere and at any scale by the assistance of these images.

For upright rectangles Summed Area Table *SAT (x, y)* is used. *SAT (x, y)* is defined as the sum of the pixels above and left of to the pixel coordinate *(x, y)*. Let *I* be an image and *I(x',y')* is a pixel on the image, then *SAT(x, y)* is defined as:

$$SAT(x, y) = \sum_{x' \le x, y' \le y} I(x', y') \tag{2.5}$$

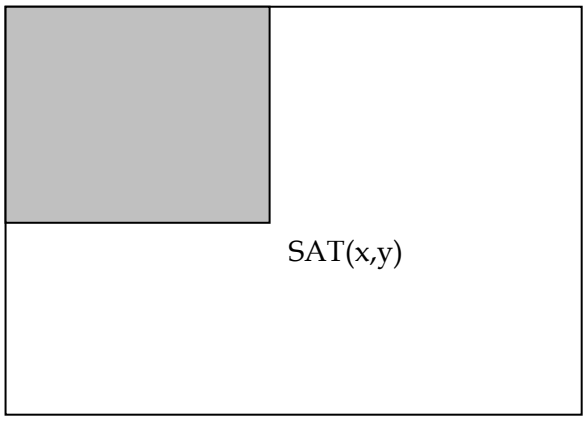


Figure 2.3: Uprigth Summed Area Table (SAT)

SAT (x, y) can be calculated with one pass from left to right and top to bottom by the equation below:

$$SAT(x, y) = SAT(x, y-1) + SAT(x-1, y) + I(x, y) - SAT(x-1, y-1)$$
$$\text{with} \quad SAT(-1, y) = SAT(x, -1) = SAT(-1, -1) = 0 \tag{2.6}$$

By using SAT pixel sum of any of the upright rectangles $r = (x, y, w, h, 0°)$ can be computed in four array references such as:

$$\begin{aligned} \mathrm{Re}\,cSum(r) &= SAT(x-1, y-1) + SAT(x+w-1, y+h-1) \\ &- SAT(x+w-1, y-1) - SAT(x-1, y+h-1) \end{aligned} \tag{2.7}$$

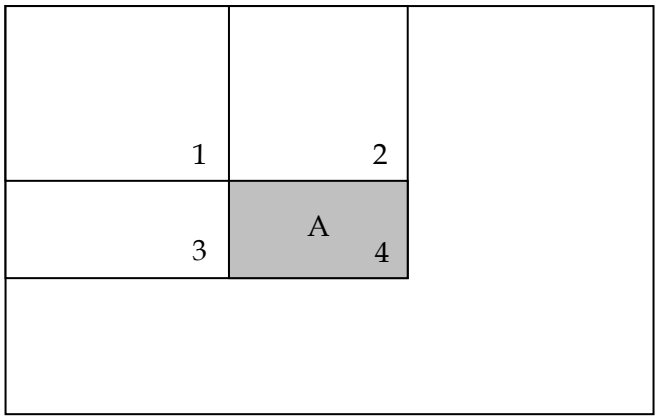


Figure 2.4: The sum of the pixels within area A can be computed with four array references : (4) + (1) – (2) – (3).

For 45° rotated rectangles Rotated Summed Area Table *RSAT (x, y)* is used. *RSAT (x, y)* is defined as the sum of the pixels of a 45° rotated rectangle with the bottom most corner at *(x, y)* and extending upwards till the boundaries of the image. Let *I* be an image and *I(x',y')* is a pixel on the image, then R*SAT(x, y)* is defined as:

$$RSAT(x, y) = \sum_{y' \le y, y' \le y - |x - x'|} I(x', y') \tag{2.8}$$

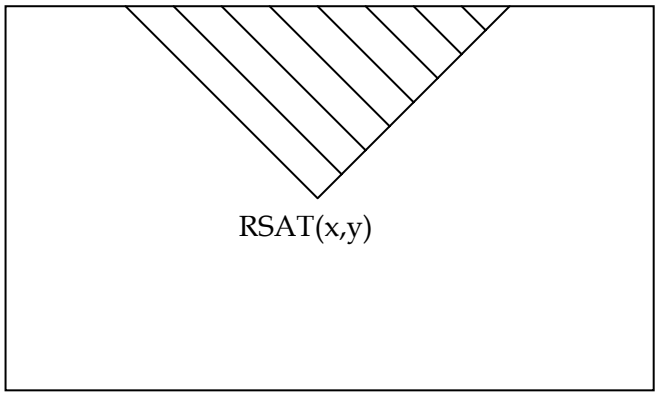


Figure 2.5: Rotated Summed Area Table (RSAT)

RSAT (x, y) can be calculated with one pass from left to right and top to bottom by the equation below:

$$RSAT(x,y) = RSAT(x-1,y-1) + RSAT(x+1,y-1) - RSAT(x,y-2) + I(x,y) + I(x,y-1)$$

with

$$RSAT(-1,y) = RSAT(x,-1) = RSAT(x,-2) = 0 \text{ and}$$

$$RSAT(-1,-1) = RSAT(-1,-2) = 0 \qquad (2.9)$$

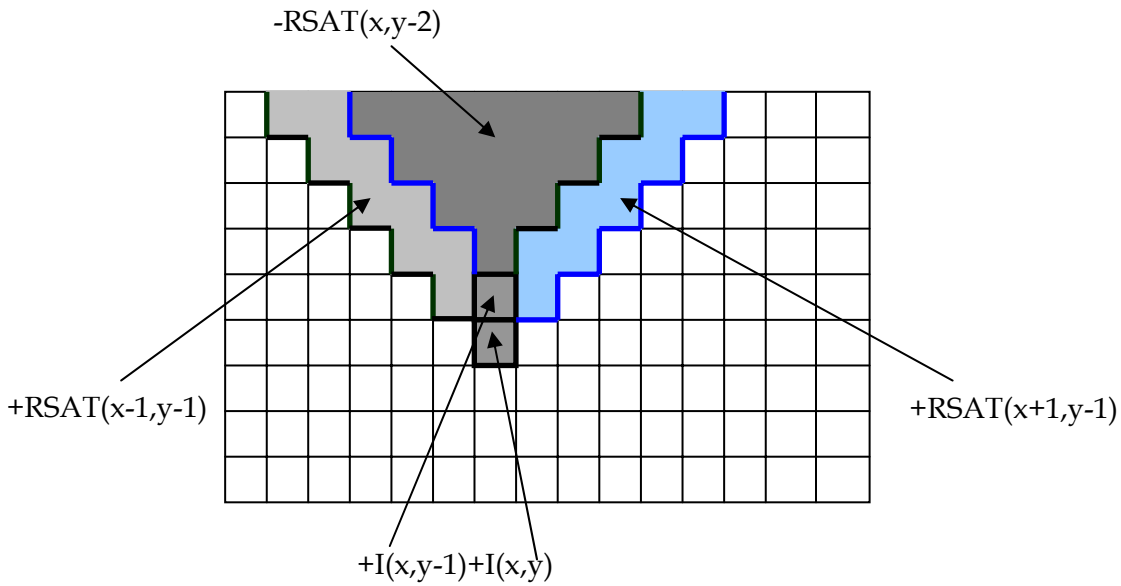


Figure 2.6: Calculation scheme for Rotated Summed Area Table

By using RSAT pixel sum of the any of the 45° rotated rectangles $r = (x, y, w, h, 0°)$ can be computed in four array references

$$\begin{aligned} \mathrm{Re}\,cSum(r) = {} & RSAT(x - h + w, y + w + h - 1) + RSAT(x, y - 1) - \\ & RSAT(x - h, y + h - 1) - RSAT(x + w, y + w - 1) \end{aligned} \tag{2.10}$$

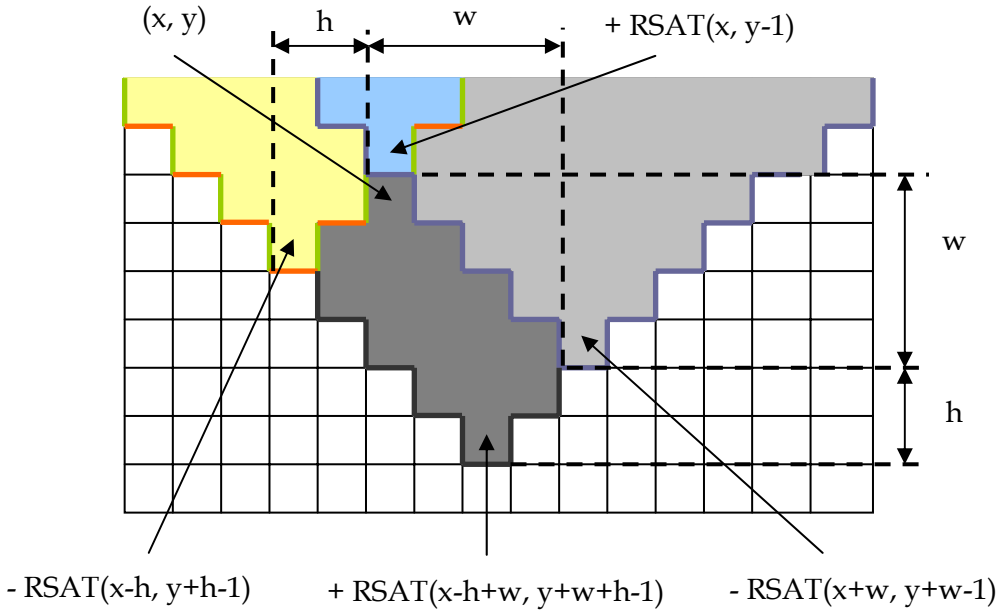


Figure 2.7: Calculation scheme for 45° rotated areas

2.2.1.3 Feature Normalization

All features used in training and detection must be variance normalized in order to minimize the effect of different lighting and contrast circumstances. Let μ be the mean and σ be the variance of the pixel values, then the process can be performed very fast by the equation below:

$$I'(x, y) = (I(x, y) - \mu)/(c\sigma), \quad c \in R \tag{2.11}$$

μ can be easily computed by the help of $SAT(x,y)$ image. However, computation of σ requires the sum of squared pixels. σ can also be easily calculated by SAT and RSAT of image $I^2(x,y)$. If this computation procedure is utilized, then computation of σ requires only 4 array references.

2.2.1.4 Feature Scaling

For training purpose, all positive samples have the same dimensions yielding the final detector operates for a fixed scale. Thus, in order to perform a search for multi-scale objects, one should scale either the image or the features used in classifier. Scaling the features used by the classifier is much more efficient than scaling the whole image. Haar-like features can be easily rescaled by the help of intermediate image representations. But a problem arises when fractional rescaling is performed because new positions become fractional. Rounding all fractional positions to the nearest integer positions can solve this problem. Due to rounding the weights of the different rectangle sums must also be corrected to protect the original area ratio between them.

2.2.2 Training

The aim of training is to find a small set of features in the complete large set of features to form an effective classifier. Each image sub-window contains quite large number of features, even larger than the number of pixels (24 x 24 window contains 117.941 features). Although one can compute each feature very efficiently, computation of all these features is quite expensive and not necessary. A small number of important features can be combined to form an effective classifier leading a fast classification. Therefore, the problem is to eliminate the large majority of the features and find out the critical features.

In Haar Object Detection, Gentle AdaBoost (see figure 2.8) is used to train the classifiers and to select the critical features [21]. There are different boosting algorithms (Discrete, Real and Gentle AdaBoost) and according to Rainer Lienhart [16] Gentle Adaboost outperforms the other algorithms. AdaBoost is an efficient procedure for selecting a small set of useful classification functions. It assigns large weights to each useful classification function and smaller weights to non-useful functions. Modified Gentle AdaBoost algorithm is used in the training. The modification to the procedure is to restrict the learner to use a set of classification

functions each depending on a single feature. Therefore, each stage of the boosting process is a feature selection process.

In the boosting language the simple learning algorithm is called the weak learner. The reason why the learner is called weak is that with this learner one does not expect to classify most of the training data correctly (A classifier can classify % 51 of the data correctly). It is only required to be better than chance. However, weighted combination of many of them can form a strong classifier and can beat the 'monolithic' strong classifiers such as SVMs and Neural Networks [22], [23]. Each weak classifier $h_j(x)$ (x is a sub-window of an image) consists of a single feature (f_j), a threshold (θ_j) and a parity (p_j) indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

Important key measures of the classifiers are "false negative" and "false positive" rates. A classifier must achieve very low false negative rate (fewer than % 1) and a false positive rate below %50 in order to be successful. There is a trade off between false positives and detection rates. This trade off can be adjusted by changing the threshold. Lower thresholds yield more false positive rates but higher detection rates. Higher thresholds yield fewer false positive rates but lower detection rates.

Gentle AdaBoost

Given N examples $(x_1,y_1), ..., (x_N, y_N)$ with $x_i \in R^k$, $y_i \in \{-1,1\}$

Start with weights $w_i = 1/N, i=1, ..., N$

Repeat for $m=1, ..., M$

Fit the regression function $h_m(x)$ by weighted least-squares of y_i to x_i with weights w_i

Set $w_i \leftarrow w_i \cdot \exp(-y_i \cdot h_m(x_i))$, $i=1, ..., N$, and renormalize weights so that $\sum_i w_i = 1$.

Output the classifier $sign[\sum_{m=1}^{M} h_m(x)]$.

Figure 2.8: Gentle AdaBoost training algorithm [16]

2.2.2.1 Cascade of Classifiers

A degenerated decision tree where each stage is a classifier trained to detect objects is called a "cascade":

Cascade:

Stage$_1$:

Classifier$_{11}$:

Feature$_{11}$

Classifier$_{12}$:

Feature$_{12}$

...

Stage$_2$: …

Classifier$_{21}$:

Feature$_{21}$

...

...

Using a cascade of classifiers increases the detection performance and decreases the computation time. A positive result received from one classifier triggers the next classifier which is adjusted to achieve very high detection rates. A sub-window is immediately rejected if any of the stages gives negative result. An object must receive positive results from all stages in order to be marked as an object. This allows the background regions to be rejected rapidly and to give much effort on promising object-like regions. The first stages in the cascade are

simple classifiers and reject the most of the background sub-windows before using more complex classifiers.

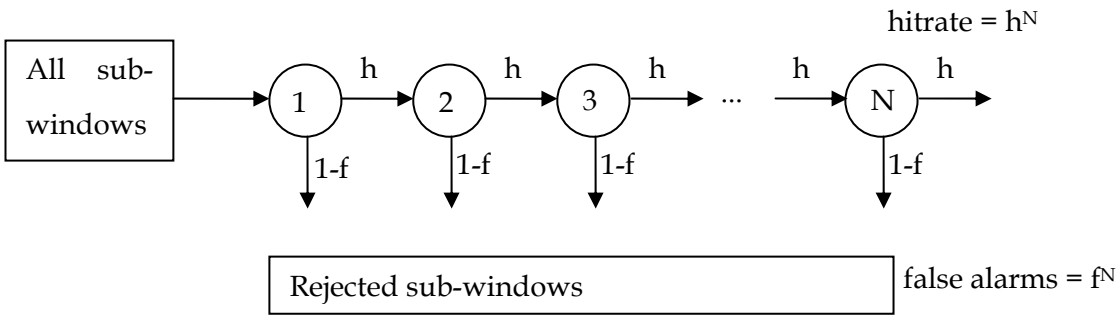


Figure 2.9: Schematic depiction of the detection cascade.

False positive rate of the trained cascade can be calculated by the formula $F = \prod_{i=1}^{K} f_i$, where F is the false positive rate, K is the number of classifiers and f_i is the false positive rate of the i^{th} classifier. Detection rate of the cascade can be calculated by the formula $D = \prod_{i=1}^{K} di$, where D is the detection rate, K is the number of classifiers and d_i is the detection rate of the i^{th} classifier. Utilizing these formulas arranging the number of stages and the stage sizes, a cascade can be designed with the desired detection and performance goals.

Stages in the cascade are constructed by the classifiers trained using adaboost. These classifiers are weak and simple classifiers. They can be trained with a % 40 false positive and % 0.1 false negative values by arranging threshold in such a way that it will give the minimum false negative value. With a % 40 false negative value, a simple classifier can eliminate % 60 of the background sub-windows before using more complex classifiers. If 20 stages were trained with the same false positive and false negative values, cascade can achieve a false alarm rate of 0.4^{20} = 1.099e-08 and a hit rate of 0.999^{20} = 0.98.

2.2.2.2 Training Algorithm

Training of each stage in the cascade requires some care. Adaboost only attempts to minimize errors and is not specifically designed to achieve high detection rates with large false positive rates. One way of providing this property is adjusting the threshold of the perceptron produced by the Adaboost. Higher thresholds yield classifiers with fewer false positives and a lower detection rate. Lower thresholds yield classifiers with more false positives and a higher detection rate.

The overall training process involves two types of tradeoffs. In most cases classifiers with more features will achieve higher detection rates and lower false positive rates. At the same time classifiers with more features require more time to compute. In principle one could define an optimization framework in which

- the number of classifier stages,
- the number of features, n_i, of each stage,
- the threshold of each stage

are traded off in order to minimize the expected number of features N given a target for F (false positive rate) and D (detection rate). Unfortunately finding this optimum is a tremendously difficult problem.

In practice a very simple framework is used to produce an effective classifier which is highly efficient. The user selects the minimum acceptable rates for f_i (false positive rate of the ith stage) and d_i (detection rate of the ith stage). Each layer of the cascade is trained by AdaBoost with the number of features used being increased until the target detection and false positive rates are met for this level. The rates are determined by testing the current detector on a validation set. If the overall target false positive rate is not yet met then another layer is added to the cascade. The negative set for training subsequent layers is obtained by collecting all false detections found by running the current detector on a set of images which do not contain any instances of the object. This algorithm is given more precisely in Figure 2.10.

User selects values for f , the maximum acceptable false positive rate per stage and d, the minimum acceptable detection rate per stage.

User selects target overall false positive rate, F_{target}

P = set of positive examples

N = set of negative examples

$F_0 = 1.0$; $D_0 = 1.0$

$i = 0$

while $F_i > F_{target}$

i <- i + 1

$n_i = 0$; $F_i = F_{i-1}$

while $F_i > f \times F_{i-1}$

n_i <- n_i + 1

Use P and N to train a classifier with ni features using Adaboost

Evaluate current cascaded classifier on validation set to determine F_i and D_i

Figure 2.10: Training algorithm for building a cascaded detector.

2.2.2.3 Training Set

Training is done with a set of positive (object) and negative (non-object) images. Positive samples are sample views of a particular object and negative samples are any other arbitrary images those do not contain that object. All of the positives samples are scaled to same base resolution *(w x h)*. Negative samples are of arbitrary size. The base resolution is also that of the detector. Detector cannot detect objects of smaller size than this base resolution. However, objects of higher resolution can be detected by scaling the detector.

Figure 2.11: Sample positive images for training an eye detector. All samples are scaled to resolution 35 x 16 pixels.

Figure 2.12: Sample negative images for training an eye detector. Images are arbitrary images at arbitrary resolution.

CHAPTER 3

VIEW POINT TRACKING

Tracking is the most important part of the proposed system regarding performance. Tracking algorithm should be robust, accurate and fast in order to achieve a reliable system performance.

In the proposed system Pyramidal Implementation of the Lucas-Kanade Feature Tracker is utilized as the tracking algorithm [24]. It is a very robust and fast algorithm enabling sufficient tracking accuracy. Details of the algorithm are described in section 3.2.

3.1 Eye Region Tracking

In the literature, there are alternative methods for facial feature tracking. Several general-purpose point trackers can be used for this purpose. Lucas and Kanade [25] have worked on the tracking problem and proposed a method that is based on a translation model between images in order to be used for registering two images for stereo matching. Utilizing the initial work of Lucas and Kanade, Tomosi and Kanade [26] developed a feature tracker upon the sum of squared intensity differences (SSD) matching measure using a translation model. Afterwards, Shi and Tomasi [27] proposed an affine transformation model. Over small inter-frame motion, the translation model has higher reliability and accuracy than that of the affine model. However, the affine model is preferable and more adequate over a longer time span.

A number of specific facial feature trackers have also been proposed by researchers. McKenna [28] proposed an approach based on a point distribution model (PDM) and Gabor wavelets in order to track rigid and non-rigid facial motion. Huang and Huang [29] also use a PDM approach to extract facial features. Their method measures the variation of the position of each point. Petajan [30] uses facial feature tracking to track eyes and the nostrils.

3.2 Pyramidal Implementation of Lucas-Kanade Feature Tracker

Lucas-Kanade Feature Tracker, which is based on optical flow algorithm, is a powerful and popular technique used in feature tracking. It is a fast algorithm and provides sufficient accuracy and robustness. Their approach is to define a match measure between fixed-size feature windows in the past and current image as the sum of the squared intensity differences. The motion is modeled as pure translation. The displacement vector, then, $\boldsymbol{d} = (d_x, d_y)$ is defined as the one that minimizes this sum. Let $u = (u_x, u_y)$ a point on the first image and w_x and w_y be two integers, then d the vector that minimizes the residual function defined as follows:

$$e(d) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (I(x,y) - J(x+d_x, y+d_y))^2 \qquad (3.1)$$

where I and J are the past and current images and where the feature window size is $(2w_x+1) \times (2w_y+1)$.

Although this is an efficient and widely preferred algorithm, it suffers from large motions. It can be a problem for the rapid head movements in our case. In order to overcome this problem Bouguet [24] proposed the pyramidal implementation of the Lucas-Kanade Feature Tracker method. In the pyramidal approach image pyramids are formed. They consist of filtered and sub sampled versions of the

original images. The displacement vectors are found iteratively upward from the coarsest level to the original level. At each level, displacement vector is calculated by maximizing a correlation measure over a small window.

3.2.1 Image Pyramid Representation

Image pyramids are formed by filtering and sub sampling the original image. Before sub sampling, original images are filtered by a low pass filter to avoid image anti-aliasing. Images in the pyramid are shown as I^L where $L = 1, 2 \ldots, L_m$. I^0 is the original image and has the highest resolution. L_m is the height of the pyramid and m is the highest level. Then, I^L is computed from I^{L-1} as

$$\begin{aligned} I^L(x,y) = \frac{1}{4} I^{L-1}(2x,2y) + \frac{1}{8}(I^{L-1}(2x-1,2y) + I^{L-1}(2x+1,2y) + \\ I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)) + \frac{1}{16}(I^{L-1}(2x-1,2y-1) + \\ I^{L-1}(2x+1,2y+1) + I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y-1)) \end{aligned} \tag{3.2}$$

The pixels at the image borders are handled according to the following formulas:

$$I^{L-1}(-1,y) = I^{L-1}(0,y) \tag{3.3}$$

$$I^{L-1}(x,-1) = I^{L-1}(x,0) \tag{3.4}$$

$$I^{L-1}(w^{L-1},y) = I^{L-1}(w^{L-1}-1,y) \tag{3.5}$$

$$I^{L-1}(x,h^{L-1}) = I^{L-1}(x,h^{L-1}-1) \tag{3.6}$$

$$I^{L-1}(w^{L-1},h^{L-1}) = I^{L-1}(w^{L-1}-1,h^{L-1}-1) \tag{3.7}$$

The width and height of image I^L are w^L and h^L respectively.

The height of the pyramid should be chosen according to the largest expected optical flow in the image. Practical values for the height are 3 and 4.

3.2.2 Pyramidal Feature Tracking

For a given point u in an image I, our goal is to find its corresponding point $v = u + d$ (d is the displacement vector) in the next image J. u and v are defined as $u^L = [u_x^L \; u_y^L]$ and $v^L = [v_x^L \; v_y^L]$ for image pyramid levels, $L = 0, \ldots, L_m$. The vectors u^L are computed as follows:

$$u^L = \frac{u}{2^L} \tag{3.8}$$

The vectors v^L are computed iteratively from the deepest level L_m to L_0 (original image). At each level, the result of the previous is used as an initial guess. These initial guesses $g^L = [g_x^L \; g_y^L]$ are, then, included in the optical flow calculation at each level. Initial guess at the deepest level of the pyramid is 0 (no initial guess is available at the deepest level of the pyramid). Afterwards, displacement vectors $d^L = [d_x^L \; d_y^L]$ are calculated as follows:

$$e^L(d^L) = \sum_{x=u_x^L-w_x}^{u_x^L+w_x} \sum_{y=u_y^L-w_y}^{u_y^L+w_y} (I^L(x,y) - J^L(x + g_x^L + d_x^L, y + g_y^L + d_y^L))^2 \tag{3.9}$$

Observing that the size of the search window at each level is fixed. Since the initial guess vector g^L is used in order to pre-translate the image patch, the residual flow vector d^L comes out to be small and easy to be computed through a standard Lucas Kanade step. After d^L is calculated, the result is propagated to the next level L-1 by passing the new initial guess g^{L-1}:

$$g^{L-1} = 2(g^L + d^L) \tag{3.10}$$

Then, the next optical flow residual vector $\boldsymbol{d^{L-1}}$ is calculated through the same way. Same procedure continues until the original image is reached ($\boldsymbol{L}$=0). Solution vector d is then available after the final optical flow calculation:

$$d = d^0 + g^0 \text{ or } d = \sum_{L=0}^{L_m} 2^L d^L \tag{3.11}$$

As seen from the second equation, d is expressed as the sum of the residual optical vectors d^L. Since residual flow vectors d^L are small and easy to be computed through a standard Lucas Kanade step, pyramidal implementation can handle large overall pixel displacement vector.

3.2.3 Pseudo-Code of the Algorithm

Let u be a point on image I, then the pseudo code of the entire algorithm that finds the corresponding location v on image J is as follows:

- Build pyramid representations of I and J: $\{I^L\}_{L=0,\ldots,Lm}$ and $\{J^L\}_{L=0,\ldots,Lm}$ (3.12)

- Initialization of pyramidal guess: $$g^{Lm} = \left[g_x^{Lm}\ g_y^{Lm}\right]^T = \left[0\ 0\right]^T \quad (3.13)$$

For $L = Lm$ down to 0 with step of -1:

- Location of point u on image I^L: $$u^L = \left[p_x\ p_y\right]^T = \frac{u}{2^L} \quad (3.14)$$

- Derivative of I^L with respect to x:

$$I_x(x,y) = \frac{I^L(x+1,y) - I^L(x-1,y)}{2} \quad (3.15)$$

- Derivative of I^L with respect to y:

$$I_y(x,y) = \frac{I^L(x,y+1) - I^L(x,y-1)}{2} \quad (3.16)$$

- Spatial gradient matrix:

$$G = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x(x,y)^2 & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y(x,y)^2 \end{bmatrix} \quad (3.17)$$

- Initialization of iterative L-K: $$\bar{v}^0 = \left[0\ 0\right]^T \quad (3.18)$$

For k=1 to K with step of 1 (or until $\left\|\bar{\eta}^k\right\|$ < accuracy threshold):

- Image difference:

$$\partial I_k(x,y) = I^L(x,y) - J^L(x + g_x^L + v_x^{k-1}, y + g_y^L + v_y^{k-1}) \quad (3.19)$$

- Image mismatch vector:

$$\bar{b}_k = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} \delta I_k(x,y) & I_x(x,y) \\ \delta I_k(x,y) & I_y(x,y) \end{bmatrix} \quad (3.20)$$

- Optical flow (Lucas-Kanade):

$$\bar{\eta}^k = G^{-1}\bar{b}_k \quad (3.21)$$

- Guess for next iteration:

$$\bar{v}^k = \bar{v}^{k-1} + \bar{\eta}^k \quad (3.22)$$

End of for loop on k

- Final optical flow at level L:

$$d^L = \bar{v}^K \quad (3.23)$$

- Guess for next level L-1:

$$g^{L-1} = \left[g_x^{L-1} \; g_y^{L-1} \right]^T = 2(g^L + d^L) \quad (3.24)$$

End of for loop on L

- Final optical flow vector: $d = d^0 + g^0$ (3.25)

- Location of point on J: $v = u + d$ (3.26)

The corresponding point is at location *v* on image *J.* It is possible to make computations at sub-pixel accuracy. Algorithm uses bilinear interpolation in order to compute image brightness at sub-pixel locations.

3.2.4 Tracking at the Boundaries

It is necessary to process points that are close to the boundaries of the image since some part of their integration window lies outside of the image. For an integration window $(2w_x+1) \times (2w_y+1)$, there is a forbidden band of width w_x (and w_y) around the image. In the pyramidal implementation, this corresponds an effective forbidden band of width $2^{Lm} w_x$ (Lm is the height of the pyramid). This band can be very significant for large integration windows and for large values of

L_m. For example, for $w_x = w_y = 5$ pixels and $L_m = 4$, forbidden band occurs to be 80 pixels around the image.

Solution to this problem is to calculate the summations in the equations (see section 2.2.3) only for the valid portion of the image neighborhood, i.e. for valid entries of $I_x(x, y)$, $I_y(x, y)$ and $\delta I_k(x, y)$ (see section 2.2.3).

3.2.5 Declaring a Feature "Lost"

Algorithm declares a feature as lost for two cases. In the comparatively simple case, a feature is declared as lost if the feature point falls out of image boundaries, whereas in the second case, which is more complicated, feature is declared as lost if the image patch around the feature point differs too much between the images *I* and *J*. This usually happens due to occlusion in the image. However, observing the occlusion is a quite challenging problem, which can be simply overcome by thresholding the cost function $\varepsilon(d)$. In this case, determining the threshold, though, comes out to be another problem.

3.2.6 Feature Selection

Until now it is described how to track feature points yet mentioned how to select these points. Every point on the image is not suitable for tracking and thus can not be a feature point. Feature points must be selected according to some mathematical criteria. Optical flow vector computation $\overline{\eta}^k = G^{-1}\overline{b}_k$ (eq. 3.21)is the core step of the algorithm. In the formula the matrix *G* must be invertible meaning that the minimum eigenvalue of *G* must be sufficiently large (larger than a threshold). Pixels satisfying this condition are called "easy to track". Overall process is as fallows:

1. At every pixel on the image *I*, compute *G* matrix and its minimum eigenvalue λ_m.
2. Find the maximum value of all λ_m and call it $\lambda_{\max}$.

3. Choose the pixels that have larger λ_m value than a certain percentage value of λ_{max}. (Can be %10 or %5).
4. Choose local max. pixels among the found in (3). (A pixel is local max if its λ_m value is maximum in its 3 x 3 neighborhood).
5. Eliminate the pixels, distance of which to another pixel of a large λ_m value is smaller than a threshold (e.g. 10 or 5).

After this process, remaining pixels are called "good to track" and can be fed to the tracking algorithm.

CHAPTER 4

INTERMEDIATE VIEW RECONSTRUCTION

This chapter presents a 3D warping based intermediate view reconstruction algorithm from multiple calibrated views and depth maps. Calibrated views mean that camera calibration parameters, the interior and exterior parameters, of the cameras capturing the images are known and depth maps are used in the construction of scene in 3D.

This chapter is organized as three main sections. First section discusses the previous work about intermediate view reconstruction. Second section includes background information about camera model, which is necessary to understand the presented algorithm. Detailed description of the algorithm is given in section three.

4.1 Intermediate View Reconstruction

Creating virtual views using stereoscopic views is called intermediate view reconstruction. There are a number of techniques available in the literature for creating virtual views. A simple case of the problem is the perfectly parallel cameras. In this case virtual view can be computed by weighted averaging between other views. There are methods that utilize this approach [31] [32] [33].

Havaldar et al. proposed a view synthesis technique in [34] which uses projective invariants. They do not require knowledge of the camera positions. An invariant, defined with respect to a transformation T, is a property which remains unchanged under transformation T. For example, parallel lines always map to parallel line under orthographic projection. Also, the ratio of line segments with respect to each other named cross ratio remains unchanged.

Another work by Irani et al. proposes to reconstruct views at any point in the scene without computing any correspondence estimation [35]. In their algorithm, they first align and compare all the projections of each line of sight emerging from the virtual camera center in the input views and then they create the virtual view. They describe the line-of-sight as the sight of the viewpoint along the line that stretches from the point to the direction desired.

Another approach in intermediate view reconstruction used especially in computer vision area is to reconstruct a 3D volumetric model of the scene and then to use this model to map new virtual views. They first relate the images with correspondence estimation and then work towards view reconstruction. Volumetric methods first create a 3D model of the scene and then manipulate and transform the object and easily create new views. However, these methods' execution speed is dependent on scene complexity. Moreover, they require sophisticated software and hardware for a realistic result.

Image-based rendering techniques are also popular in intermediate view reconstruction. The term image-based is used here to describe that the methods use explicit images rather than volumetric 3D models. A survey by Kang [36] gives details about image-based rendering techniques.

4.2 Camera Model

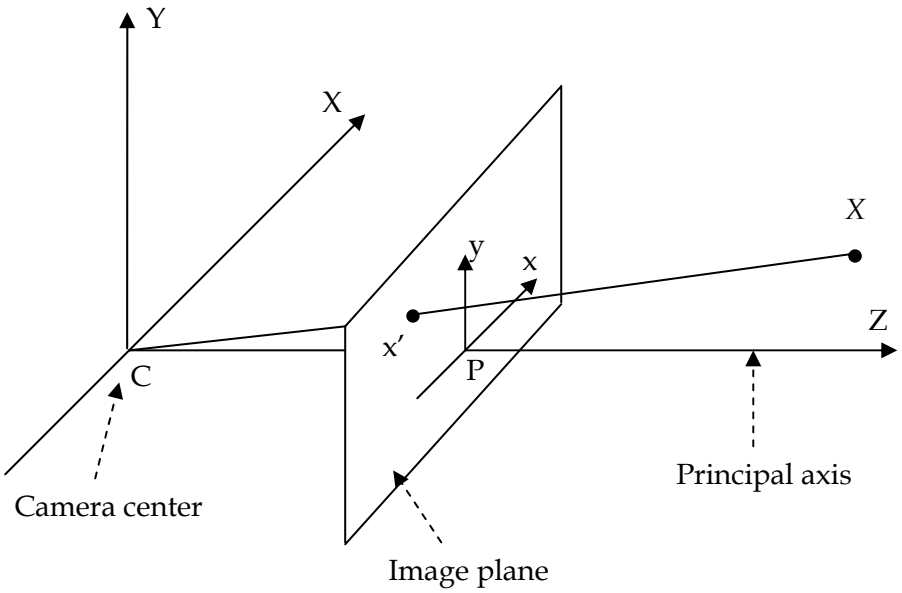
Modeling the camera provides us the relation between 3D world coordinate system and 2D image coordinate system. Camera is usually modeled in matrix

form called as the *Camera Matrix*. This matrix maps 3D points in world coordinates system to 2D points on the image plane.

In this thesis, *finite projective camera model* is used. This model is described in the next sections.

4.2.1 Finite Projective Camera Model

Finite projective camera model is also known as the *pinhole camera model*. Most of the cameras are described relatively well by this model. In the model, light enters the camera through an infinitesimally small hole and forms an inverted image on the camera surface facing the hole. To simplify things, image plane can be placed between the focal point of the camera and the object, so that the image is not inverted. The projection of 3D points to 2D points is called perspective projection. Focal point of the camera is the perspective projection center (*C – Camera center*). The ray passing through the camera center, which is perpendicular to the image plane, is called *principle axis* and the point of intersection of this ray with the image plane is known as *principal point*.



$x' = PX, \quad x' \in P^2, X \in P^3$

$\boldsymbol{x'} = [a\ b\ 1]^T$ (2D image corrdinate)

$\boldsymbol{X} = [A\ B\ C\ 1]^T$ (3D points)

Figure 4.1 – Pinhole camera geometry

Finite projective camera model is described by a 3 x 4 matrix called *camera projection matrix* (*P matrix*). *P* Matrix can be defined as the product of two matrixes K and M ($P = KM = K[R \mid t]$). *K* matrix is known as the *camera calibration matrix* and includes the intrinsic parameters of the camera. *M* matrix includes the *extrinsic parameters,* rotation and transformation.

The camera center can be found as the right null vector of the projection matrix, $PC = 0$;

Pseudo-inverse of the matrix *P* is called back-projection matrix *(P^+)*. P^+ is calculated as $P^+ = P^T(PP^T)^{-1}$ for which $PP^+ = I$.

4.2.2 Intrinsic Parameters

Camera calibration matrix is composed of 5 intrinsic parameters:

1) Focal length, (f)
2) Principal point, (u_0, v_0)
3) Aspect ratio, (a)
4) Skew, (s)

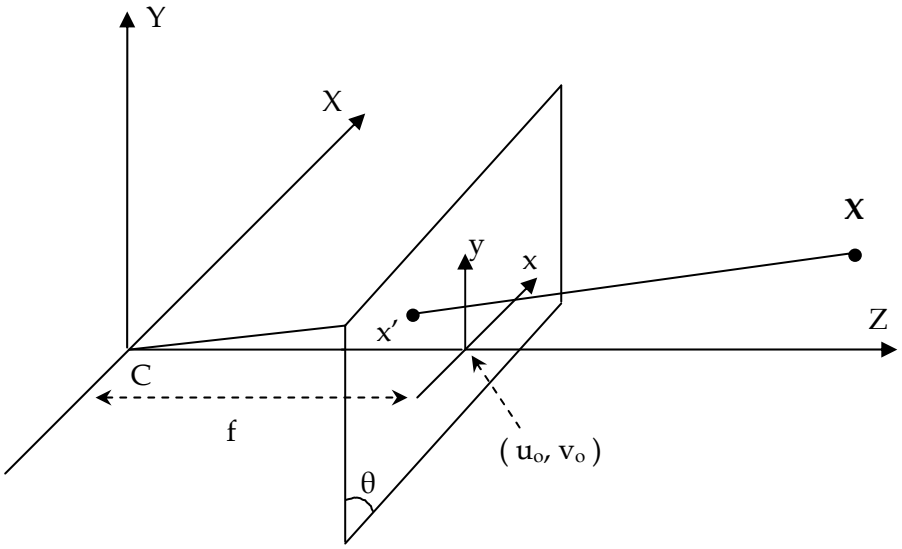


Figure 4.2 – Camera intrinsic parameters

Focal length (f) of the camera is the most important intrinsic parameter.

If 3D point $\boldsymbol{X}$ is taken as $\begin{bmatrix} A & B & C & 1 \end{bmatrix}^T$, then the projected coordinate 2D point $\boldsymbol{x}'$ can be calculated as $\boldsymbol{x}' = \begin{bmatrix} f(A/B) & f(B/C) \end{bmatrix}^T$ from the similarity of triangles.

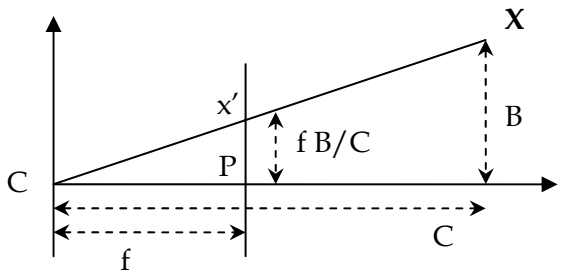


Figure 4.3 – Side view of the projection of a 3-D point

Principal point is assumed as the origin of the image plane. However, image origin is usually taken as the upper left corner of the image. This case adds a shift to the formula of ***x*** as**:**

$$x' = \left[f(A/B) + u_0 \qquad f(B/C) + v_0 \right] \tag{4.1}$$

Aspect ratio is the ratio of the width of a pixel to the height of the pixel. In the case that pixels are not square but rectangular, i.e., we have an aspect ratio (a_x/a_y) different from unity. As scaling by pixel dimensions and scaling by focal length is algebraically the same, therefore, we express the focal length in terms of pixel dimensions as $\alpha_x = f / a_x$ and $\alpha_y = f / a_y$. Then the formula becomes as:

$$x' = \left[\alpha_x (A/B) + u_0 \qquad \alpha_y (B/C) + v_0 \right] \tag{4.2}$$

Skew parameter is added to the formula in case when the angle (θ) between optical axes is not 90-degrees:

$$x' = \left[\alpha_x (A/B) - \alpha_x \cot\theta (B/C) + u_0 \qquad (\alpha_y / \sin\theta)(B/C) + v_0 \right] \tag{4.3}$$

This transformation can be explained in matrix form as:

$$x' = K[I \mid 0]X = \begin{bmatrix} \alpha_x & -\alpha_x \cot\theta & u_0 \\ 0 & \dfrac{\alpha_y}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} [I \mid 0]X \tag{4.4}$$

Parameters in the K matrix are called the intrinsic parameters and estimation of these parameters is called the *interior calibration* of the camera.

4.2.3 Extrinsic Parameters

Extrinsic parameters are used when the 3D point coordinates are measured according to a coordinate system other than the camera coordinate system.

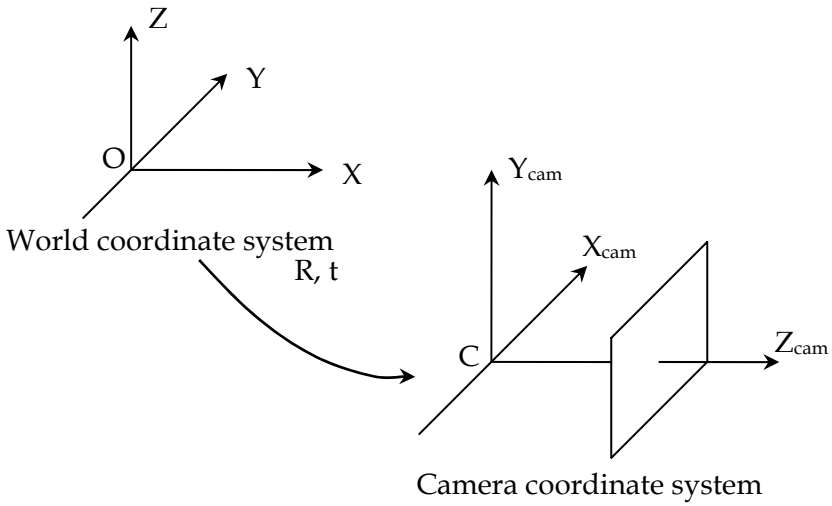


Figure 4.4 – Transformation between world and camera coordinate systems

In that case, the translation (t) and rotation (R) between the two coordinate systems must be added to the formula:

$$x' = K\left[I \mid 0\right]\begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} X = K\left[R \mid t\right]X = KMX \tag{4.5}$$

and hence

$$x' = PX \quad \text{with } P = K\left[R \mid t\right] \tag{4.6}$$

Parameters in the M matrix are called the *exterior parameters* and estimation of these parameters is called the *exterior calibration* of the camera. There are 6 exterior parameters, 3 rotations and 3 translations. Together with the 5 internal parameters, P matrix has eleven degrees of freedom.

4.3 Intermediate View Reconstruction Algorithm

Intermediate view reconstruction algorithm used in the system is a 3D warping based algorithm that uses multiple calibrated views of a scene and a related depth map for each view. It is a simple and straightforward method. However, satisfactory results can be obtained. Calibrated views mean that camera calibration parameters, the interior and exterior parameters, of the cameras

capturing the images are known. Exterior parameters donate the camera position for that view.

Each view has a related depth map image. Depth value of a pixel on the view can be calculated by using the intensity value at the same coordinate on the depth map image. By using the depth value and camera calibration parameters, 3D point for that pixel can be calculated.

Intermediate view is constructed for a virtual camera position. Projection matrix for the virtual camera is constructed using the same internal parameters for the calibrated views and virtual camera position (external parameters).

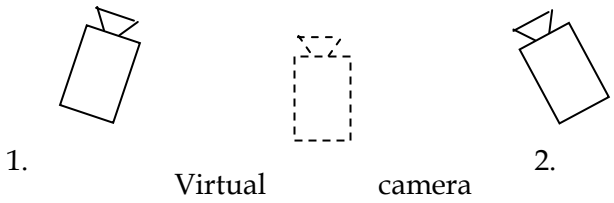


Figure 4.5: Virtual camera position

In the implementation of method, 3D Video Color and Depth Sequences (Ballet and Break dancing) from Sing Bing Kang (Microsoft research) [37] is used. These sequences contain 100 frames and depth maps for 8 calibrated cameras.

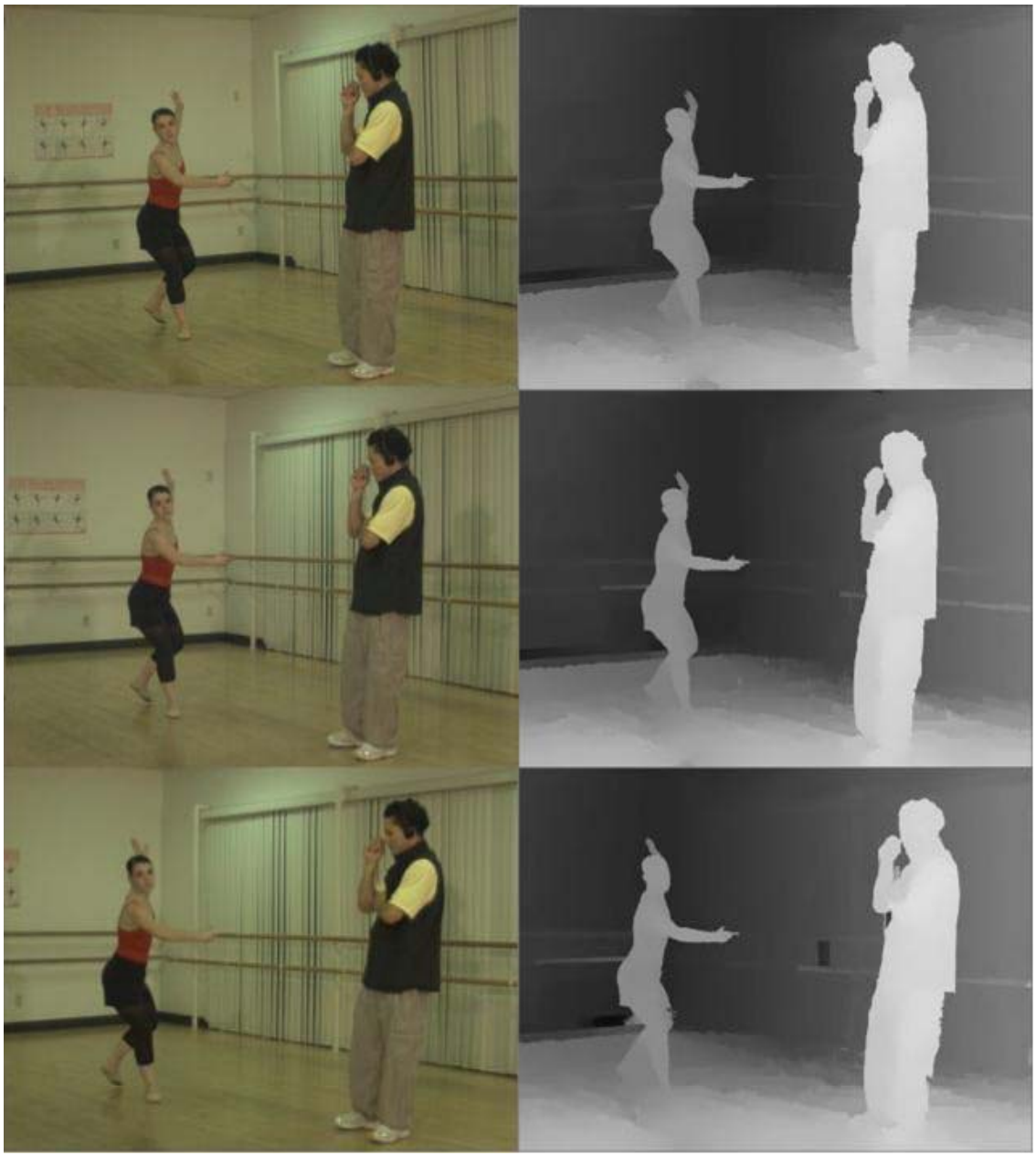


Figure 4.6 – Multiple views of a scene and depth maps

4.3.1 Outline of the Algorithm

For a given virtual camera position, outline of the algorithm can be described in 5 steps:

1) Find the two views taken at nearest camera positions to the virtual camera position.
2) Find the 3D points for the first nearest view by using the back projection matrix and depth map of the view.
3) Project 3D points on the virtual image plane using the virtual camera matrix.
4) Repeat step 2 and 3 for the second nearest view but fill only the gaps on the virtual view.
5) Fill the remaining gaps by interpolation.

4.3.1.1 Choosing the views

In order to calculate the intermediate view, two views among the calibrated views are used in the algorithm. First view is chosen according to its camera position that is the nearest to the virtual camera position. Distance is measured as the *Euclidean distance* between camera centers. The nearest positions are used because these views are the most similar views to the virtual view. Therefore, there would be less occluded regions. Second view is selected as the first one among the remaining views but with a constraint that the distance of the selected camera position to the virtual camera position must be smaller than the distance to the first selected camera position. This constraint is applied in order to be sure that the occluded regions for the second view are different than the first one.

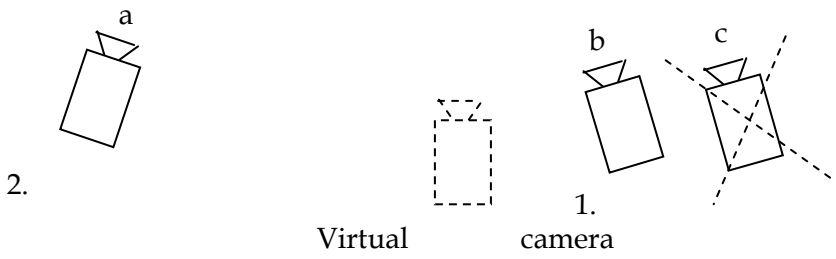


Figure 4.7: Camera b is selected as the first view. Camera c is not selected as the second view because its distance to the first selected camera is smaller than the distance to the virtual camera. Instead of c, camera a is selected as the second view

4.3.1.2 Back-Projection and Forming the Virtual View

Pixels of the first selected view are back projected to the 3D points using the back projection matrix and the depth map of the view. This is also known as 3D reconstruction of the scene (see Figure-4.8). Virtual view is constructed by projecting 3D points on the virtual image plane. If more than one point is projected to the same pixel coordinate, the point with the smaller depth value is chosen because the point which has smaller depth value is the nearest point to the camera and occludes the other points. This process forms most of the virtual view, but there remain gaps in the image because of occlusions (see Figure-4.9). In Figure-4.9, virtual camera position is taken as the position of the 4th camera position in the Ballet sequence [37] and the first selected view is the 3rd view.

Figure 4.8 - Back projected 3D points using one view

Figure 4.9 - Re-projection of 3rd camera view to 4th camera position in ballet sequence

Second selected view is used to fill the gaps in the virtual image. Points on the view are back-projected and then projected on the virtual image plane if it falls in a gap. In Figure-4.10, 5th camera view is selected as the second view on the result shown in Figure-4.9.

Figure 4.10 - Occluded regions are filled by using the second view as 5th view of ballet sequence

4.3.1.2.1 Back Projection Using Depth Map

Each point on the image plane maps a ray passing through camera center and point itself in 3D world, i.e. the projected 3D point X is somewhere but on the ray. This ray can be calculated from the two points that are on the ray: camera center C and the point $P^{+}x'$:

$$X(\lambda) = P^{+}x' + \lambda C \tag{4.2.1}$$

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} p \\ r \\ s \\ t \end{bmatrix} + \lambda \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ 1 \end{bmatrix} \tag{4.2.2}$$

This equation itself does not enough to find the projected 3D point X. But if the depth value ($Z = (C / D)$) of the point is known, one can find λ:

$$\lambda = \frac{(s - Zt)}{Z - C_3} \tag{4.2.3}$$

Once λ is known, equation 4.2.2 can be solved for the 3D point X.

4.3.1.3 Interpolation

In order to fill out the holes in the image, some interpolation/extrapolation or patching methods are used in the literature. These methods fill such holes by using the intensity information of the nearest rendered region to avoid disturbing effects [38] [39]. In the method that is used in the thesis, remaining gaps in the virtual image is filled by interpolation using the adjacent pixels. Finally, 3x3 median filter is applied on the image. Figure-4.11 shows the resulted image after interpolation step and 4th view of the ballet sequence. The PSNR of the constructed view is 33.39 dB.

Figure 4.11 – Real 4th camera view (on the top) and constructed virtual 4th camera view (on the bottom). PSNR value is 33.39 dB.

CHAPTER 5

PROPOSED SYSTEM

In this chapter, detailed description of the proposed system and system components are given. The chapter includes system architecture, system algorithmic flow and implementation of the system blocks.

5.1 System Architecture

The proposed system consists of hardware and software parts. Hardware and software architectures can be seen in sections 5.1.1 and 5.1.2 respectively.

5.1.1 Hardware Architecture

System hardware consists of three main parts:

1) Computing unit,
2) 3D Display,
3) Camera.

A camera is mounted on top of the display in such a way that it can track the user. It is connected to the computing unit. A standard webcam can be used as camera. Whereas, a camera with a frame grabber could be utilized as a better solution. Images are captured from the camera by the software running on the computing unit. Software detects and tracks the view point of the observer by using the

captured images. Afterwards, correct stereoscopic image is rendered by the software on the 3D Display according to the view point of the observer. 3D display can be an autostereoscopic display or stereographic glasses

In the system, the following hardware components are used:

Camera: Creative NX Webcam.

Computing unit: Computer with Pentium IV 3.0 GHz processor.

3D Display: E-D 3D shutter glasses with NVIDIA Quadro2 MX graphics card on 19'' monitor.

Figure 5.1: System hardware setup

Figure 3.2: E-D 3D Shutter glasses

5.1.2 Software Architecture

System software consists of six main parts:

1) Image Acquisition: Captures images from the camera.
2) View Point Detection: Detects the view point of the observer from the captured images.
3) View Point Tracking: Tracks the view point of the observer from the captured images.
4) View Point Calculation: Extract the view point of the observer.
5) Intermediate View Reconstruction: Construct the correct stereoscopic image pair.
6) Rendering: Render the stereoscopic view on the 3D Display.

Detailed implementations of each part are described in sections 5.1.2.2 through 5.1.2.7.

5.1.2.1 Software Operation

Software operation starts with acquiring an image from webcam. The next step depends on to the current mode of the software. Software has two modes of operation: detection or tracking. Starting mode is the detection mode. View point detection component is called in this mode. Software continues its operation in detection mode until a successful output is obtained. Afterwards, features are found in the detected region and software changes its mode to tracking. View point tracking component is called in this mode. Tracking continues while the number of features is more than 2 or the number of tracked images is below 100. Otherwise, the software resets itself and changes its modes to detection. After a successful detection or tracking step view point of the observer is calculated. Then, correct stereoscopic image pair is formed by the intermediate view reconstruction component and rendered on the display.

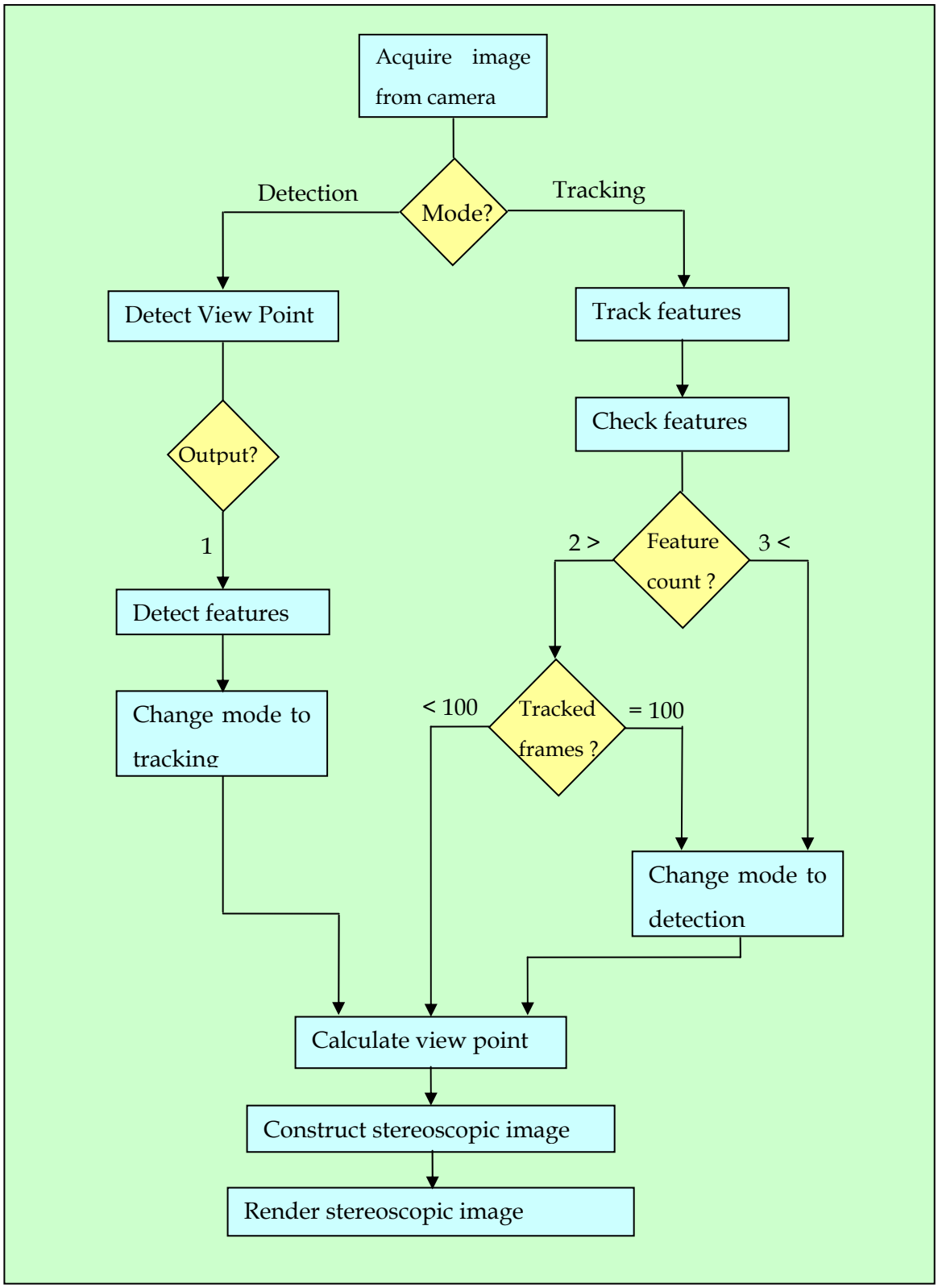


Figure 5.3: Flow chart of the software operation

5.1.2.2 Image Acquisition

This part of the software is dependent to the camera used in the system. A webcam connected to the computer via USB is used in the system implementation. Image Acquisition part is implemented by using Intel Open Source Computer Vision Library (OpenCV) Video I/O API .

Image capture resolution is 320 x 240 pixels and this is the image resolution used by other algorithms in the system. In this resolution, maximum frame rate of the camera is 30 Hz and this is the upper bound for the system frame rate.

5.1.2.3 View Point Detection

View Point Detection is implemented by using the Haar Object Detection Method. Details of the algorithm are described in Chapter 2. Implementation of the algorithm is done by using OpenCV Library (see Appendix A). In order to detect a specific object, first a classifier is trained and then the classifier is applied on the image to detect the object. Two classifiers are trained by using this method. One of them is trained for autostereoscopic displays, while the other is trained for stereoscopic glasses. Classifier for the autostereoscopic displays detects the eyes of a person on an image. On the other hand, classifier for stereoscopic glasses detects the face of the user wearing glasses.

Classifier used for eye detection is trained with 3305 positive samples and 1105 negative samples by using Gentle Adaboost. Positive samples are obtained from Facial Recognition Technology (FERET) Database [40]. FERET Database actually contains only facial images. However, eye coordinates for some of the images are also given in the database. Positive samples are obtained from those images by cropping an area of 35 x 16 pixels size around the eye locations. All of the positive sample images have the same resolution of 35 x 16 pixels and they are grayscale images. Some of the eye sample images can be seen in Figure 5.4.

Figure 5.4: Some of the positive eye sample images

Negative samples for the training are selected from any arbitrary images which do not contain eyes of a person. They do not have a fixed resolution unlike in the case of positive samples. Some of the negative sample images can be seen in Figure 5.5.

Figure 5.5: Some of the negative samples

Training parameters (see section 2.2.2.2) *f*, the maximum acceptable false positive rate per stage, and *d*, the minimum acceptable detection rate per stage, are chosen as 0.5 and 0.995 respectively as suggested by [18]. 16 stages are trained with these parameters and finally, detection rate of 0.931619 and false alarm rate of 0.000018 are obtained.

Trained classifier is tested on MIT+CMU frontal face set [41] for comparison. In the test, 66 images with 122 labeled frontal faces are used which are suitable for the eye detection. Even though MIT+CMU database is created for evaluating algorithms for detecting frontal views of human faces, detection rate of 0.795 and false alarm rate of 0.07377 are obtained.

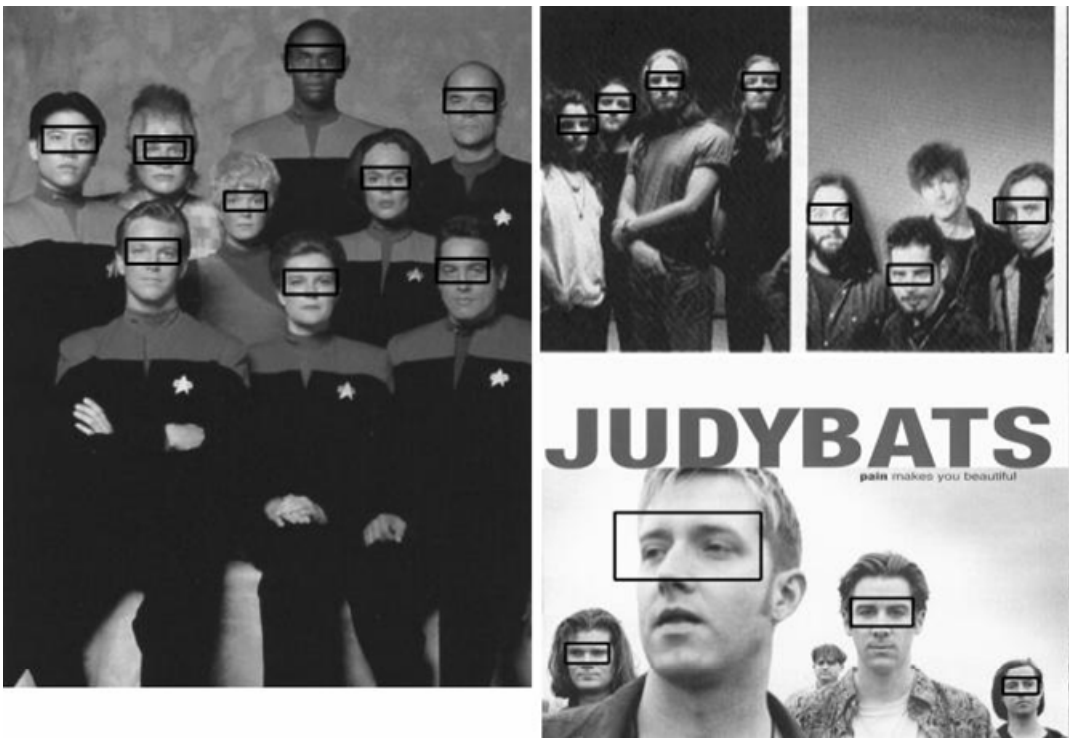


Figure 5.6: Some of the test results for eye classifier

The output images of Image Acqusition block are used for the eye detection. They are 8-bit grayscale images. Classifier trained for eye detection is applied on these images in order to detect the eyes. Scale parameter (see section 2.2.1.3) for the classifier is set to be 1.1 in order to detect objects at multiple scales. Eye detection results for different people can be seen in Figure 5.7. Average speed of the detection with on the images having resolution of 320 x 240 pixels is measured as 62 milliseconds.

Figure 5.7: Eye detection results for different resolution faces.

Classifier used for stereoscopic glasses detection is trained with 67 positive samples and 1105 negative samples by using Gentle Adaboost. The number of positive samples is quite low compared to the eye case since there is no specific image database for the stereoscopic glasses. Therefore, positive samples are formed by taking the photographs of the people wearing stereoscopic glasses in the laboratory environment. Resolution for the positive samples is 30 x 40 pixels. Some of the positive sample images can be seen in Figure 5.8. Negative samples for the training are the same with the ones used for the eye case.

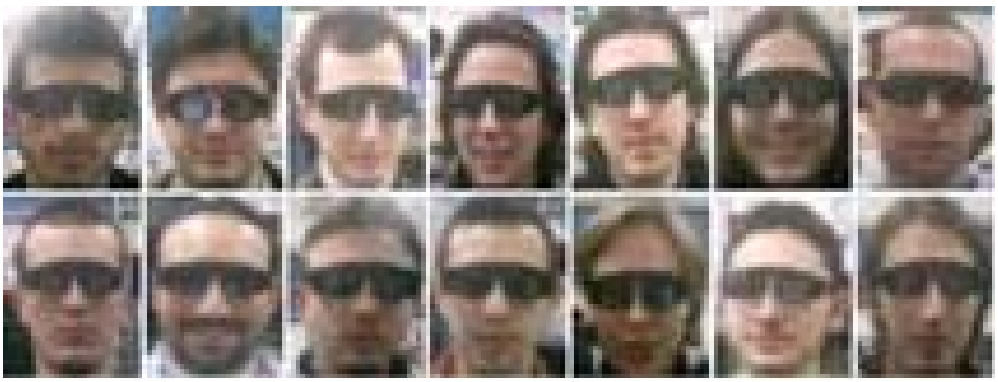


Figure 5.8: Some positive sample images for stereoscopic glasses

Training parameters (see section 2.2.2.2) *f,* the maximum acceptable false positive rate per stage, and *d*, the minimum acceptable detection rate per stage, are chosen as 0.5 and 0.995 respectively as suggested by [18]. 12 stages are trained with these parameters and finally, detection rate of 1.000000 and false alarm rate of 0.000006 are obtained.

Performance of the trained classifier is tested on the 43 test image. Detection rate of 0.95 and false positive rate of 0.004 are obtained as the test result. Trained classifier can detect different types of stereoscopic glasses which have black color.

The output images of Image Acqusition block are used for the detection of stereoscopic glasses. Trained classifier is applied on these images in order to

detect stereoscopic glasses. Scale parameter (see section 2.2.1.3) for the classifier is set to be 1.1 in order to detect objects at multiple scales. Stereoscopic glasses detection results for different people and different eyeglasses can be seen in Figure 5.9. Average speed of detection on the images having resolution of 320 x 240 pixels is measured as 24 milliseconds. This result is better compared to that of the eye classifier (62 milliseconds). The reason why different results are obtained is the complexity and base resolutions of the classifiers. Eye classifier contains 16 stages and it is much more complex than the other classifier which contains 12 stages. The base resolution of the stereoscopic glasses classifier is 30 x 40 pixels whereas the other has the resolution of 35 x 16 pixels which is nearly at half size. For this reason, eye classifier runs on two times more search windows for the same image. This process requires two times more time since computation of the features is scale independent.

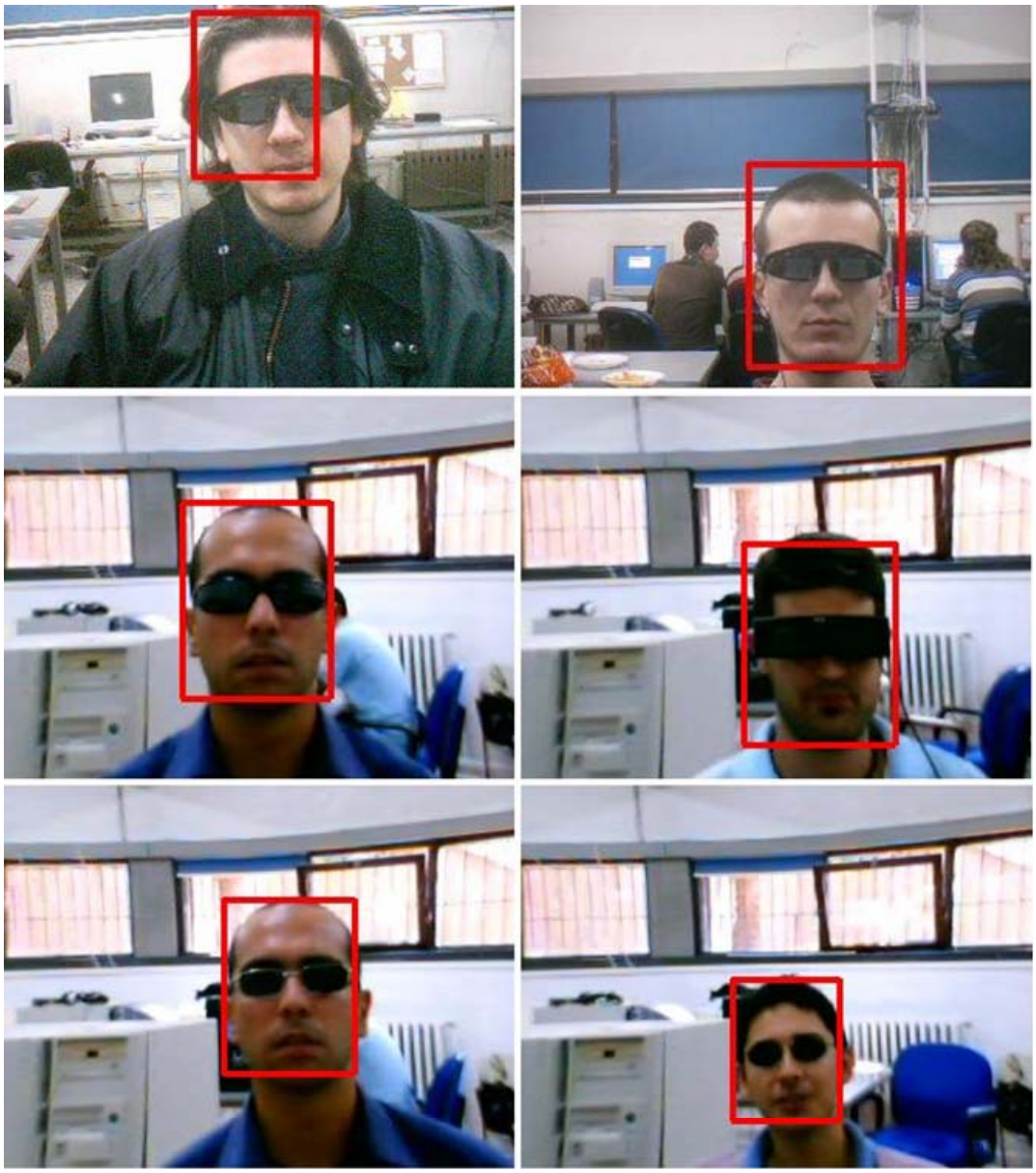


Figure 5.9: Stereoscopic glasses detection results for different people and different eye glasses

5.1.2.4 View Point Tracking

View Point Tracking is implemented by using the Pyramidal Implementation of the Lucas-Kanade Feature Tracker. Details of the algorithm can be seen in Chapter 3.

Tracking is performed after a positive result of the detection part. First, 10 feature points are identified in the detected region (see section 3.2.6). Then, these points are fed to the tracking algorithm. 3 pyramid levels are used. Search window size is determined to be as 15 x 15 pixels.

Speed of the implementation is extremely high. For the frame resolution of 320 x 240 pixels, average tracking time is found to be 8.3 milliseconds on a Pentium IV 3.0 GHz computer. Eventually 60 fps is achieved together with other jobs (e.g. frame capture from video file, render on a window, etc.).

In the experiments, tracker is tested on a 330 frames length video. In the video, a person looks at the monitor and moves his head. In the first frame, eyes of the person are detected and then feature points are identified on the detected area. These points are being tracked until the end of the video. Tracking results can be seen in Figure 5.10.

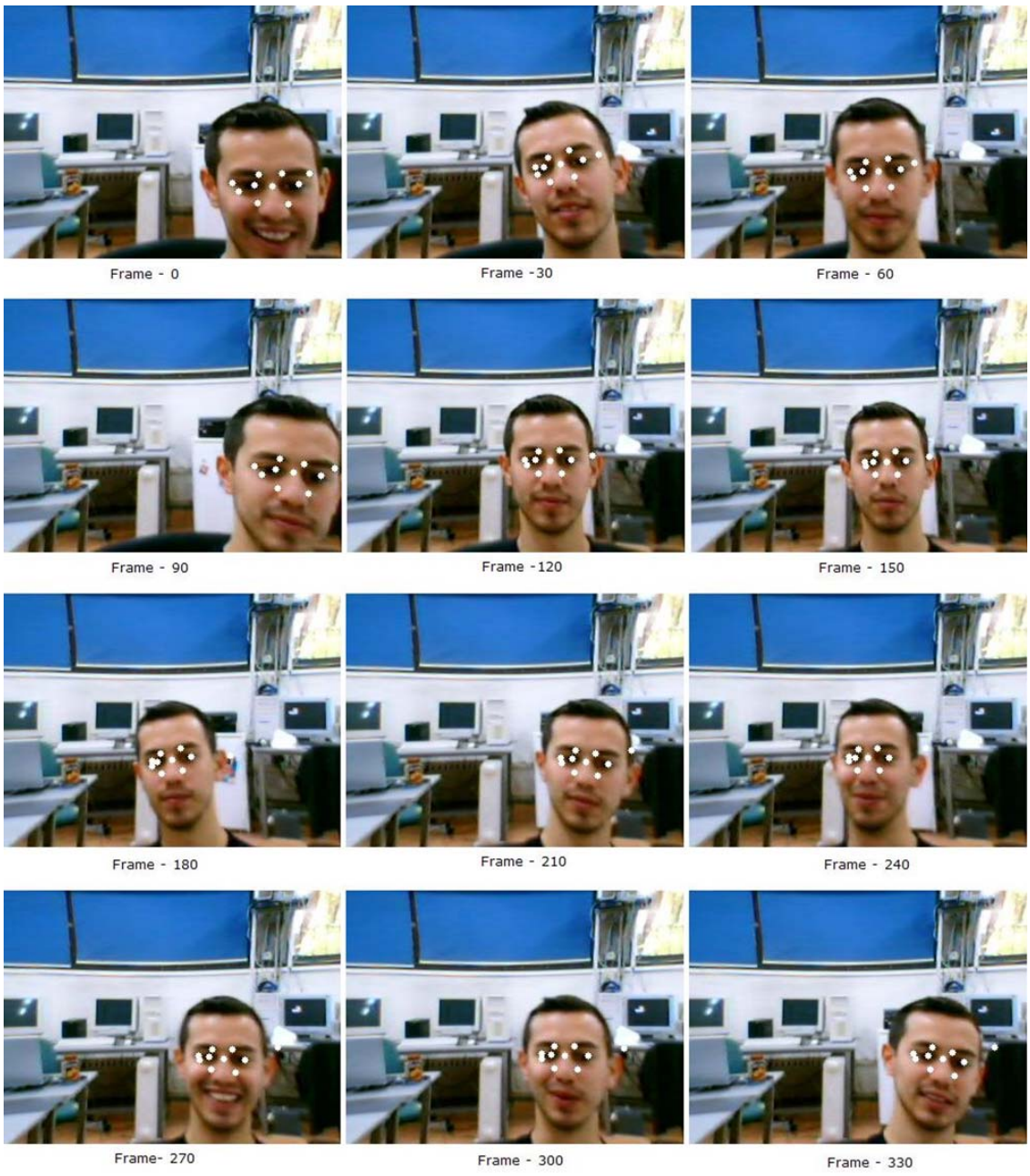


Figure 5.10: Output frames of the tracked video. Output frames are taken at every 30 frames.

5.1.2.5 View Point Calculation

Calculation of the view point of the observer is started after the region of the observer on the captured images is detected (View Point Detection). Let w and h be the width and height of the detected region respectively. The point at $(w/2, h/2)$ is, then, defined as the position of the observer on the image (see Figure 5.11). Afterwards, position of the observer is tracked together with the tracking of the feature points (View Point Tracking) (see Figure 5.12). The motion vector of the observer position is assigned to the median of the feature motion vectors at every frame.

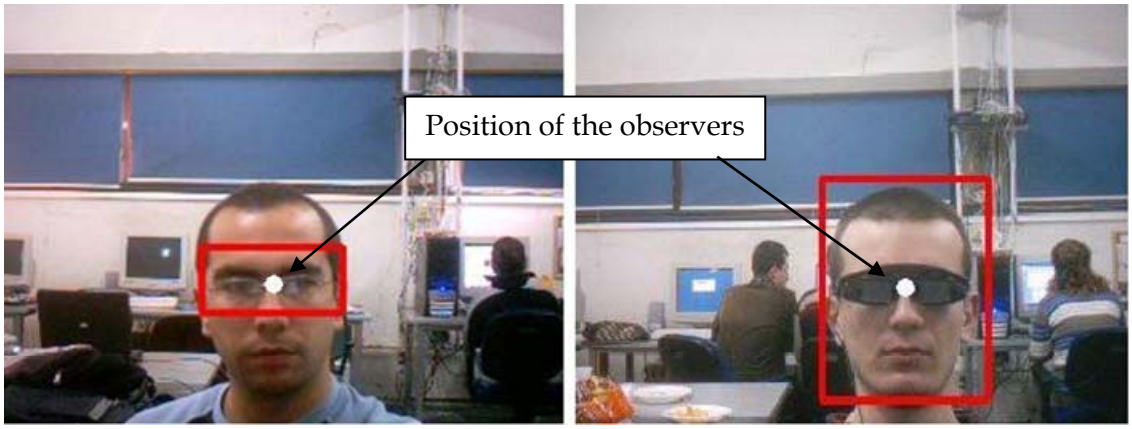


Figure 5.11: Position of the observers on the detected regions

Figure 5.12: Position of the observers while tracking features

Position of the observer on the image is used to calculate two angles θ and φ. When a coordinate system is attached to the center of the camera as the positive z-axis is towards the observer, θ becomes the angle between observer and y-z plane and φ turnes out to be the angle between the observer and x-z plane (see Figure 5.13). The relation between image and world coordinates is obtained by internal camera calibration. The focal length of the camera, *f*, and image coordinates of the observer position, p(x, y) are used in the calculations that are shown in Figure 5.14 and Figure 5.15. Focal length of the webcam used in the system is found to be 258 pixels as the result of the camera calibration

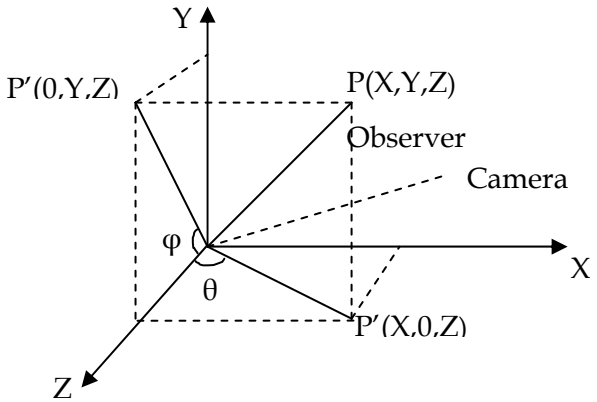
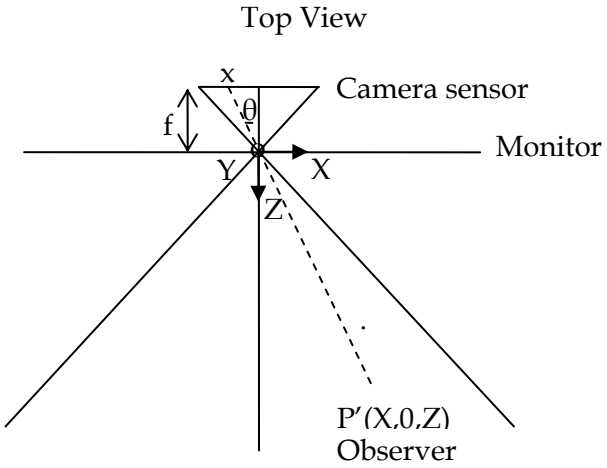


Figure 5.13: θ and φ

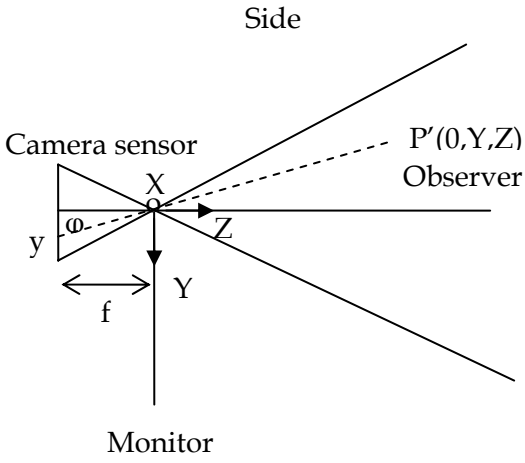


f: Camera focal length

x: X coordinate of the observer on image

$\theta = \arctan(x, f)$

Figure 5.14: Calculation of θ



f: Camera focal length

y: Y coordinate of the observer on image

$\varphi = \arctan(y, f)$

Figure 5.15: Calculation of φ

View point of the observer can exactly be identified by the two angles θ ,φ and Z value being the distance between the observer and monitor. In the proposed system Z value is not being calculated. The distance between the observer and monitor is assumed to fall between 50-80 cm.

5.1.2.6 Intermediate View Reconstruction

Intermediate view reconstruction block is implemented by using a 3D warping based algorithm as described in chapter 4. However, it is not utilized as a part of the whole software, since the algorithm does not operate in real-time. Constructing a virtual frame of size 640 x 480 pixels requires 2.13 seconds and this is far away from real-time requirements. Consequently, correct virtual stereoscopic view can not be constructed in real-time. Instead, 8 pre-constructed videos are used in the system. Videos are constructed offline for different view points. View point angle φ is taken as 0. The other angle θ is ranging from minimum value to maximum value at equal distance. The sight angle of the webcam used in the system is 64 degree. Thus, θ is incremented by 8 degrees (see Figure 5.16). Finally, distance to the camera is taken as 60 cm.

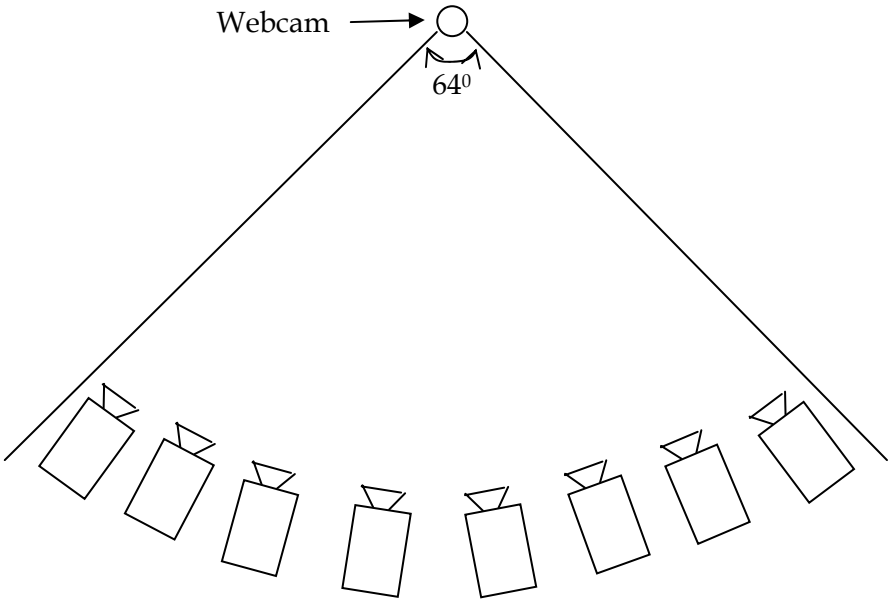


Figure 5.16: Position of the virtual cameras

As the system is operating, the videos are used to construct stereoscopic images. Two nearest virtual camera positions to the view point of the observer are selected thus yielding current frame is being constructed from the corresponding videos.

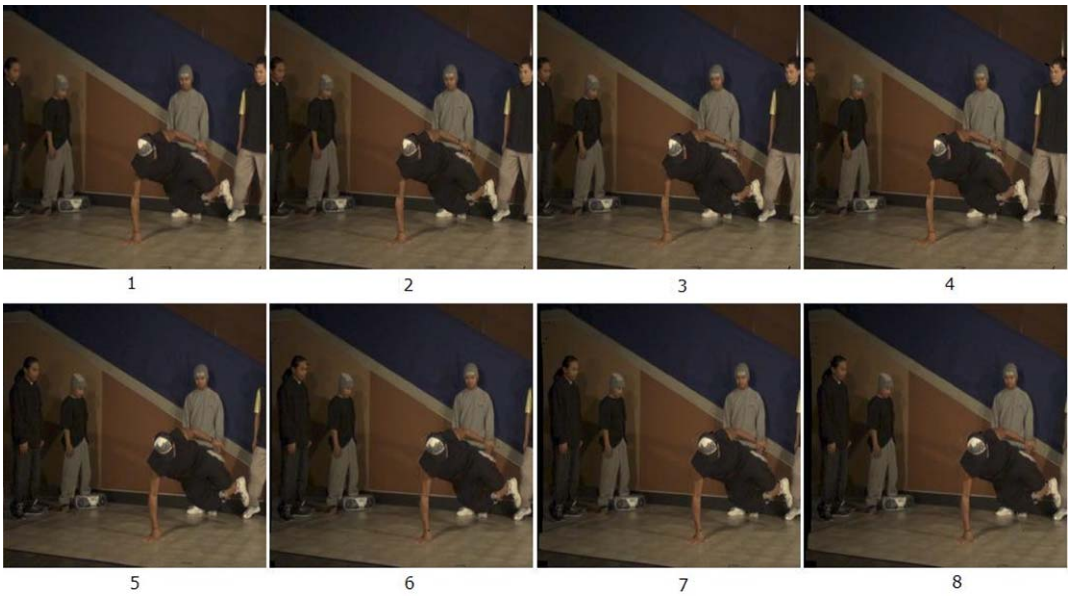


Figure 5.17: Sample frames from 8 virtual constructed videos.

5.1.2.7 Rendering

Rendering part of the system is dependent to the 3D display used in the system. E-D 3D shutter glasses with NVIDIA Quadro2 MX graphics card is used in the system as the 3D display block. E-D 3D shutter glasses require a special graphic card with stereo support. NVIDIA Quadro2 MX graphics cards provide a synch signal to the eye glasses while rendering images for the left and right eyes on the monitor. Left and right images are rendered by using the OpenGL Quad-buffered Stereo API that is enabled by the graphics card.

CHAPTER 6

CONCLUSION

6.1 Summary of the Thesis

In this thesis, a single user view rendering system is proposed and implemented. The system can easily be installed on a standard PC together with an autostereoscopic display or stereoscopic glasses (shutter, polarized, pulfrich, and anaglyph) with appropriate video card. System is compose of three hardware units and six software components.

Hardware components are display unit, computing unit and a single camera. In the implementation of the system, they are selected as: E-D 3D shutter glasses with NVIDIA Quadro2 MX graphics card, Computer with Pentium IV 3.0 GHz processor and Creative NX Webcam. Webcam is mounted on top of the display in such a way that it can track the user. Developed software is running on the computer. Software captures images from camera and detects and tracks the view point of the observer. Correct stereoscopic image is, then, rendered by the software on the display based on the observer.

Developed software is compose of six components: Image acquisition, view point detection, view point tracking, view point calculation, intermediate view reconstruction and rendering. Image acquisition part captures image from webcam. Observer position is then detected on the captured image by the view point detection component. Haar object detection method is utilized in the

implementation as the view point detection method. Two classifiers are trained by using this method. One of them is trained for autostereoscopic displays, while the other is trained for stereoscopic glasses. Classifier for the autostereoscopic displays detects the eyes of a person on an image. On the other hand, classifier for stereoscopic glasses detects the face of the user wearing glasses. After a successful detection step, images are fed to view point tracking component. View Point Tracking is implemented by using the Pyramidal Implementation of the Lucas-Kanade Feature Tracker. Tracking continues until the observer is lost or a pre-determined period is elapsed. System then resets itself and continues with detection step. View point calculation is done after a successful detection or tracking step. Exact 3D location of the observer is not calculated. Instead, looking direction of observer is calculated (two angles, θ and φ) and the distance between the observer and monitor is assumed to fall between 50-80 cm. Correct stereoscopic view is, then, rendered on the display by using the calculated view point. A 3D warping-based method is utilized in the system as the intermediate view reconstruction method. Rendering is done by using the OpenGL Quad-buffered Stereo API that is enabled by the graphics card.

6.2 Discussions and Future Work

The performance of the classifier which detects the eyes of a person is better than the performance of the other which detects the face of the user wearing glasses. The reason for this result is the positive samples used for the training. 3305 positive samples which are obtained from Facial Recognition Technology (FERET) Database are used for the eye classifier. However, the number of positive samples used for the second classifier is only 67, because there is no specific image database for the stereoscopic glasses. Samples are formed by taking the photographs of the 12 people wearing stereoscopic glasses in the laboratory environment. The number of people was quite low for the classifier to generalize the case. As a result, classifier fails more often for the different people. Therefore, to form a training set is the vital point of the training a classifier. A more comprehensive training set can be formed for the training of the second classifier.

Classifier which detects the face of the user wearing glasses is trained with sample images of people wearing E-D 3D shutter glasses. However, trained classifier can detect different types of stereoscopic glasses which have black color.

The trained classifiers detect the observer only if the frontal face of the user appears in the acquired image. They fail for the side views of the face. However, the system has to track the observer only when the user is watching the display and it is assumed that the observer intends to watch the display in a comfortable way. That is, the observer will not look askew at the display.

Lighting conditions affects the performance of the detection and tracking. Classifiers fail in the poor lighting conditions. This is due to the fact that features which the classifier looks for do not appear in the image. Camera used in the system is also related to this issue. The use of a good quality camera may provide sufficient images for the classifiers in the poor lighting condition.

People wearing eye glasses may also cause problem in the eye detection step. Eye glasses may reflect light coming from the display. In this case, the silhouette of the display appears on the eye glasses and the eyes of the person can not be seen on the image, so the classifier fails to detect eyes.

In the tracking step, tracked feature points may drift in the long run or may be lost if the user points fall out of image or turns his/her head to another location. In order to overcome these situations, system resets itself in pre-determined periods and continues with the detection step. This property provides the stability of the system.

Intermediate view reconstruction part of the system is the only part that does not operate in real-time. The construction speed of a virtual view is far away from the real-time requirements. Consequently, correct virtual stereoscopic view can not be constructed in real-time. Instead, 8 virtual videos are constructed for the demonstration of the system. Two nearest virtual camera positions to the view

point of the observer are selected thus yielding current frame is being constructed from the corresponding videos. However, virtual views should be constructed for the view point of the observer in every frame in the ideal system. This can be achieved with an intermediate view reconstruction method that operates in real-time. As a future work, intermediate view reconstruction can be replaced with a new method that operates in real-time.

REFERENCES

[1] M. Andiel, S. Hentschke, T. Elle, E. Fuchs, "Eye-Tracking for Autostereoscopic Displays using Web Cams", *Proceedings of SPIE Vol. 4660, Stereoscopic Displays and Virtual Reality Systems IX*, 2002

[2] Yong-Sheng Chen, Chan-Hung Su, Jiun-Hung Chen, Chu-Song Chen, Yi-Ping Hung, and Chiou-Shann Fuh, "Video-based Eye Tracking for Autostereoscopic Displays," Optical Engineering, Vol. 40, Issue 12, pp. 2726-2734, December, 2001.

[3] Fraunhofer-Institute for Telecommunications – Heinrich-Hertz-Institut (HHI), "Evaluation of a Single User autostereoscopic Display System for 3D-TV and PC oriented applications – an example of a user centered design cycle", ECMAST'98

[4] SeeReal Technologies Inc, http://www.seereal.com, visited 31 July 2006

[5] Samah Ramadan, Wael Abd-Almageed and Christopher Smith, "Eye Tracking using Active Deformable Models," The III Indian Conference on Computer Vision, Graphics and Image Processing, Ahmedabad, India, December 2002.

[6] Zhiwei Zhu , Kikuo Fujimura , Qiang Ji, "Real-time eye detection and tracking under various light conditions", Proceedings of the symposium on Eye tracking research & applications, March 25-27, 2002, New Orleans, Louisiana

[7] Bart de Liefde, Andre Redert, Emile Hendriks, "Eye Tracking for Viewpoint Adaptive Video Systems in Living Room Situations", Proceeding WIAMIS, 2003

[8] A. Haro, M. Flickner, and I. Essa, "Detecting and tracking eyes by using their physiological properties, dynamics, and appearance," in Proceedings IEEE CVPR 2000.

[9] A.L. Yuille, P.W. Hallinan, D.S. Cohen, Feature extraction from faces using deformable templates, Int. J. Comput. Vision 8 (2) (1992) 99–111.

[10] X. Xie, R. Sudhakar, H. Zhuang, On improving eye feature extraction using deformable templates, Pattern Recognit. 27 (1994) 791–799.

[11] A. Pentland, B. Moghaddam, T. Starner, View-based and modular eigenspaces for face recognition, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'94), Seattle, WA, 1994.

[12] W. min Huang and R. Mariani, Face detection and precise eyes location, in: Proc. Int. Conf. on Pattern Recognition (ICPR'00), 2000.

[13] G.C. Feng, P.C. Yuen, Variance projection function and its application to eye detection for human face recognition, Int. J. Comput. Vis. 19 (1998) 899–906.

[14] G.C. Feng, P.C. Yuen, Multi-cues eye detection on gray intensity image, Pattern Recognit. 34 (2001) 1033–1046.

[15] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001.

[16] R. Lienhart, A. Kuranov, and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. Technical report, MRL, Intel Labs, 2002

[17] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In International Conference on Computer Vision, 1998.

[18] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001.

[19] William T. Freeman and Edward H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[20] H. Greenspan, S. Belongie, R. Gooodman, P. Perona, S. Rakshit, and C. Anderson. Overcomplete steerable pyramid filters and rotation invariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[21] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In Machine Learning: Proceedings of the Thirteenth International Conference, Morgan Kauman, San Francisco, pp. 148-156, 1996.

[22] Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1997.

[23] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In IEEE Patt. Anal. Mach. Intell., volume 20, pages 22–38, 1998.

[24] Jean-Yves Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm", Intel Corporation, Microprocessor Research Labs, OpenCV Documents, 1999.

[25] B. D. Lucas and T. Kanade. "An iterative image registration technique with an application to stereo vision". *Proc. International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[26] C. Tomasi and T. Kanade. "Detection and tracking of feature points". *Carnegie Mellon University Technical Report CMU-CS-91-132, Pittsburgh, PA*, 1991.

[27] J. Shi and C. Tomasi. "Good features to track". *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[28] S. McKenna, S. Gong, R. P. Wurtz, J. Tanner, and D. Banin. "Tracking facial feature points with Gabor wavelets and shape models". *Proc. Int. Conf. on Audio- and Video-Based Biometric Person Authentication, Crans-Montana, Switzerland*, pages 35–42, 1997.

[29] C. Huang and Y. Huang. "Facial expression recognition using model-based feature extraction and action parameters classification". *Journal of Visual Communication and Image Representation*, 8(3):278–290, 1997

[30] E. Petajan and H. P. Graf. "Robust face feature analysis for automatic speechreading and charcter animation". *Proc. Second International Conference on Automatic Face and Gesture Recognition, Killington, Vermont*, pages 357–362, 1996.

[31] J. Konrad, "View reconstruction for 3-D video entertainment: issues, algorithms and applications," in Proc. Int. Conf. on Image Process. and its Applications, pp. 8–12, July 1999.

[32] A. Mancini and J. Konrad, "Robust quadtree-based disparity estimation for the reconstruction of intermediate stereoscopic images," in Proc. SPIE Stereoscopic Displays and Virtual Reality Systems, vol. 3295, pp. 53–64, Jan. 1998.

[33] A.-R. Mansouri and J. Konrad, "Bayesian winner-take-all reconstruction of intermediate views from stereoscopic images," IEEE Trans. Image Process., vol. 9, pp. 1710–1722, Oct. 2000.

[34] P. Havaldar, M. Lee, and G. Medioni, "View synthesis from unregistered 2D images," in Graphics Interface I96, pp. 61–69, 1996.

[35] M. Irani, T. Hassner, and P. Anandan, "What does the scene look like from a scene point?," in Proc. European Conf. on Computer Vision, vol. 2, pp. 883–897, 2002.

[36] S.-B. Kang, "A survey of image-based rendering techniques," Tech. Rep. CRL 97/4, Digital Equipment Corp., Cambridge Research Lab, Aug. 1997.

[37] Microsoft Research, http://research.microsoft.com/vision/InteractiveVisualMediaGroup/3DVideoDownload/, visited 31 july 2006

[38] J.I. Park and S. Inoue. Arbitrary view generation from multiple cameras. Proceedings of International Conference on Image Processing, 1:149–152, 1997.

[39] Paul E. Debevec, Yizhou Yu, and George D. Borshukov. Efficient view dependent image-based rendering with projective texture-mapping. Eurographics Rendering Workshop, pages 105–116, 1998.

[40] Face Recognition Technology (FERET) program, http://www.frvt.org/FERET/, visited 31 july 2006

[41] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *IEEE Patt. Anal. Mach. Intell.*, volume 20, pages 22–38, 1998.