Next generation 3D Display Applications using Laser Scanning Pico Projectors

by

Kaan Akşit

A Thesis Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Electrical and Electronics Engineering

Koç University

June, 2014

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a PhD thesis by

Kaan Akşit

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Hakan Urey

_____

Prof. Murat Tekalp

_____

Prof. Metin Sezgin

_____

Prof. Levent Onural

_____

Prof. Göksen Göksenin Yaralıoğlu

Date: _____

Baligh Tom Letnor'un anısına...

REALITY IS EVERYTHING that exists. That sounds straightforward, doesn't it? Actually, it isn't. There are various problems. What about dinosaurs, which once existed but exist no longer? What about stars, which are so far away that, by the time their light reaches us and we can see them, they may have fizzled out?

Richard Dawkins

ABSTRACT

This thesis presents four novel 3D display methods and a number of new applications using commercial laser scanning pico projectors that have been developed with major contributions from our laboratory during the last decade. The new methods developed in this thesis are (i) mixed-polarization stereoscopic display using single-projector and passive polarized glasses, (ii) a new type of glasses that can relieve the accommodation and vergence conflict associated with all the stereoscopic displays, (iii) rotating screen based auto-stereoscopic display for a single-viewer using two pico-projectors, and (iv) multi-user multi-view auto-stereoscopic display using an array of pico-projectors. Furthermore, a head-mounted pico-projector is also demonstrated as a new augmented-reality application aimed at motion-capture actors.

The first method introduces a new twist on glasses based stereoscopic projection displays as it utilizes polarization and color multiplexing simultaneously and avoids weaknesses associated with previous methods. This new method is named Mixed Polarization 3D. Color imbalance artifacts associated with anaglyph glasses are avoided by alternating the colors presented to each eye. In mixed polarization 3D, flicker is not observed even at 60 Hz since both eyes receive at least one color frame in every single frame.

The second method, named Super Stereoscopic 3D Display (SS3D), introduces a major improvement on stereoscopic 3D displays for avoiding accommodation-vergence conflict. Our method provides at least two views to a single eye by using special apertures equipped with selective filters in front of the users' eye. Our designs can be embedded into conventional stereoscopic glasses or can be developed as special contact lenses.

The third method introduces a single viewer multi-view autostereoscopic projection

display, which is invented in our laboratory by a co-worker. The technique uses two mobile projectors, a rotating retro-reflective diffuser screen, and a head-tracking camera. As two dynamic viewing slits are created at the viewers position, the slits can track the position of the eyes by rotating the screen.

The last method introduces a new modular multi-user multi-view autostereoscopic display architecture based on an array of pico-projectors and a vertical diffuser screen. A single projector provides a portion of the each perspective observed through the vertical diffuser. Unlike similar projector-array based displays, the screen assembly complexity and the size of the projection display hardware have been decreased dramatically. Our proposal has the capability to provide horizontal expansion in the screen size, and an increase in the number of different perspectives as well.

Lastly, an augmented reality (AR) system is proposed and developed for supporting motion capture actors. This system allows seeing and exploring the digital environment without occluding the actors visual field. An in-house prototype is built by combining a retro-reflective screen that covers the walls and a headband consisting of a laser scanning projector with a smartphone. Built-in sensors on the smartphone provide navigation capabilities in the digital world.

ÖZETÇE

Bu tez, laboratuvarımızın yüksek katkılarıyla geliştirilmiş olan lazer tarama temelli projeksiyon ürünlerini kullanarak yenilikçi üç boyutlu (3B) görüntüleme yöntemleri ve yeni uygulamlarını sunmaktadır. Tez bünyesinde bahsi geçen yöntemler şu şekildedir: (i) kutuplanma (polarizasyon) karmalı tek projektör temelli edilgen (pasif) gözlükler kullanarak 3B yansıtım, (ii) 3B ekranlarda odaklanma ve göz hareketi sebepli sorunların etkisini azaltmak için yeni bir gözlük, (iii) İki piko projektör kullanarak tek kullanıcı için döner ekran temelli gözlük gerektirmeyen 3B ekran, (iv) Projektör dizini kullanarak çok kullanıcı için çoklu görüntüleme yapabilen gözlüksüz 3B ekran. Bütün bu yöntemlere ek olarak, giyilebilir projektör kullanarak dış dünya ile örtüşen bir sanal gerçeklik uygulaması da tez içinde sunulmuştur.

Sunulan ilk yöntem, gözlüklü 3B projeksiyon ekranlarına, eş zamanlı olarak kutuplanma ve renk çoklama özelliklerini ekleyerek doğrudan bir yenilik sunmakta ve eski yöntemlerin zayıf yönlerini gidermeyi sağlamaktadır. Bahsi geçen yönteme kutuplanma karmalı 3B yöntemi adı verilmiştir. Renk temelli 3B gözlüklerde var olan renk dağılım dengesizliği, her bir göze renklerin sırayla gösterilmesi sayesinde aşılmıştır. Yöntemimiz sayesinde kırpışma olmadan 3B gösterim, 60 Hz tazeleme frekansında da yapılabilir hale gelmiştir.

İkinci yöntem yani yüksek seviye gözlüklü 3B gösterimi yöntemi (SS3D), mevcut olan gözlüklü 3B ekranlarda odaklanma ve göz bebeğ uyumu adına önemli bir iyileştirme sunmaktadır. Bizim yöntemimiz, tasarladığımız ısık seçiç filtreli özel iğne delikleri sayesinde her bir göze en az iki görüntü sağlamaktadır. Bahsi geçen yapı mevcut gözlüklü 3B yapılara doğrudan eklenebilmekte veya özel kontak lens olarak da geliştirilebilmektedir.

Üçüncü yöntem laboratuvarımız bünyesinde çalışmakta olan bir iş arkadaşımın

icadı olup tek kişiye çoklu görüntülemeli gözlüksüz 3B deneyimi sunmaktadır. Yöntem iki adet taşınabilir projektörü, göz takip kameraları ve dönebilen retro-yansıtıcı-ışık dağıtıcı çiftini birleştirmektedir. Etkin pencereler kullanıcının gözünü takip ederek konumunu değiştirebilmektedir.

Son yöntem, bir dizi piko projektör ile dikey ışık dağıtıcı kullanarak, çoklu kullanıcı desteği olan çoklu görüntülemeli gözlüksüz bir 3B ekran mimarisi sunmaktadır. Mimari içerisinde her bir projektör tek başına bakılan açıya göre ışık dağıtıcı üzerinde sadece görüntünün bir parçasını sunabilmektedir. Benzerlerinden farklı olarak, ekrannın karmaşıklığı ve projektör dizinin hacmi gözle görülür şekilde iyileştirilmiştir. Yöntemimiz ekranın yatay düzlemde genişlemesini ve gösterilen perspektif sayısının artırılmasını da desteklemektedir.

Son olarak, gerçek dünya ile örtüşen bir sanal gerçeklik uygulaması (AR) sunulmuştur. Bahsi geçen mimari sanal ortamı, oyuncunun görüş alanını engellemeden gösterebilme ve keşfedilebilme özelliği sunmaktadır. Mimarimiz, duvarları kaplayan retro-yansıtıcılar ve giyilebilir lazer taramalı projektör-akıllı cep telefonu çiftini birleştirerek gerçeklenmiştir. Akıllı cep telefonu üzerinde yön bulmak için kullanılan algılayıcılar sayesinde sanal ortamda gezinme kabiliyetini sunulabilmektedir.

ACKNOWLEDGMENTS

My sincere appreciations goes to my supervisor, Prof. Hakan Urey, whose vast knowledge guided me as if a bright torch in the darkness during my path to degree of PhD. I am thankful to him for providing me the chance to be a member of the Optical Microsystems Laboratory (OML) at Koç University. Things could not have evolved without the trust and the support of him.

I should also thank the valuable collaborators outside of OML, Dr. Özgür Çakmak, Dr. Phil Surman, Dr. Mark Freeman, and Daniel Kade for their aid and guidance.

A very special thanks goes to a valued member of OML, Selim Ölçer for his continuous support. I should also thank to my dear colleagues at OML: Erdem Ulusoy, Osman Eldeş, Hazal Er, Amir Hossein Ghanbari Niaki, Başarbatu Can, Mehmet Kadıoğlu, and Ekin Akyürek. They were always helpful in many ways, and provided a good team spirit with an endless effort. I am grateful for the never ending support and the friendship from all previous and current members of OML.

I recognize that the whole research through my PhD studies would not have been possible without the financial support of TÜBİTAK, Microvision, Inc., and European Union's seventh framework.

I must also acknowledge Dr. Stefan Mangold, who has provided me the opportunity to be a part of Disney Research at Zurich for the duration of three months during my PhD studies. It was a great experience to work in hand-to-hand with valuable researchers, Stefan Schmid, Dr. Giorgio Corbellini, and Dr. Thomas Gross.

Finally, I am grateful to my parents, and my whole family for all the love and support I received over the years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

$3D$	:	Three dimensional
IPD	:	Interpupillary distance
SLM	:	Spatial Light modulator
QWP	:	Quarter wave plate
HWP	:	Half wave plate
PMS	:	Polarization Maintaining Screen
$LHCP$	:	Left handed circular polarizer
$RHCP$	:	Right handed circular polarizer
SMV	:	Super Multiview
$MEMS$	:	Micro electromechanical system
$SS3D$	:	Super Stereoscopic 3D
$HMPD$	:	Head mounted projection display
HMD	:	Head mounted display
CGI	:	Computer generated imagery
OML	:	Optical Microsystems Laboratory
LCD	:	Liquid Crystal Display
$RFID$	:	Radio Frequency identification
$HDMI$	:	High-definition multimedia interface
GPS	:	Global pointing service
MHL	:	Mobile High-definition Link
$VSYNC$	:	Vertical Synchronization

Chapter 1

MOTIVATION

Our motivation is to introduce and exploit new 3D display methods with and without requiring any special eye-wear. At the core of our research on 3D displays, we use laser scanning based pico projectors, which is a new category of displays that are portable and battery operated. Pico projectors combines red, green, and blue laser among with a MEMS scanner. This ideal combination provides a large gamut, focus-free operation, and a small modular design. We showed that ,when used as the light engine of the 3D displays, pico projectors can facilitate various new ways in all subcategories marked under Figure 1.1. The Pico projectors' miniature size helps us to decrease hardware volume into levels, where mobility, and modularity can be introduced. In some of our research cases, we employed retro-reflective materials to enable bright and shiny images with battery powered devices. Thus, pico-projectors are a great candidate for energy efficient solutions. Scaling down the 3D displays will help us to introduce immersive 3D applications, such as virtual reality, interactivity, or augmented reality, into our daily lives.

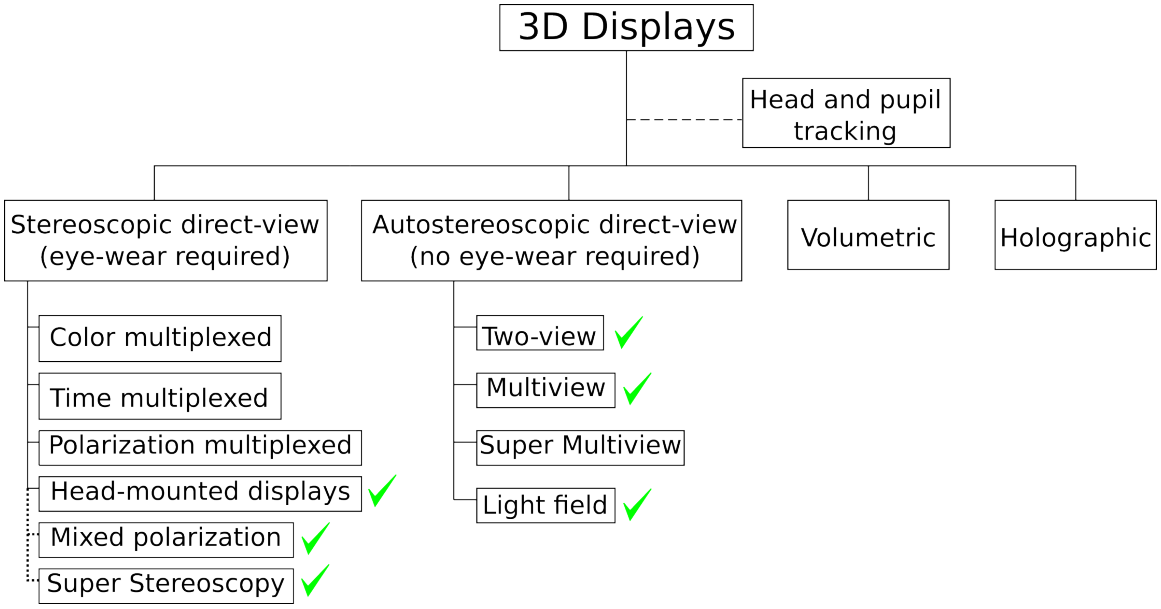


Figure 1.1: Sketch categorizing 3D displays in terms of their eye-aid requirement. The categories that this thesis contributed are highlighted with a green check mark.

The overall philosophy and the drive-force of this work is to reveal key-enabling information on (i) hologram like improved 3D vision by forming monocular parallax in binocular type stereoscopic displays, and (ii) optimized flicker-free 3D vision by using mobile projectors with limited hardware specifications. Thus, the 3D displays depending on this revealed information will provide discomfort-free better 3D images satisfying the needs of human visual systems. On the other hand, the laser scanning mobile displays' capabilities were extended into a new domain. A valuable information on new architectures were extracted for future researchers in the field, who are willing to pursuit the dream of life-like 3D images through laser scanning displays. We are confident that the derived architectures can find themselves a great place in the consumer-electronics domain.

Foundations of our motivation will be detailed in the next sections. Color and polarization properties of the light are exploited heavily in this thesis. An introduction to light will be given in the most simplistic manner for curious beginner readers. A quick overview on laser scanning pico projectors will be introduced. Human visual

system will be introduced in a simple manner. Lastly, 3D display types will be introduced. Overall, readers will find a grounding introduction on basis of 3D vision.

1.1 The Light

The light as we understand is an electromagnetic phenomena, which is made of energy packs called photons. The photons have an electrical field and a magnetic field. Those two fields are perpendicular to each other as illustrated under Figure 1.2. The direction of the electric field determines the polarization state of the light. In the exemplary sketch under Figure 1.2, the case of linear polarization is sketched.

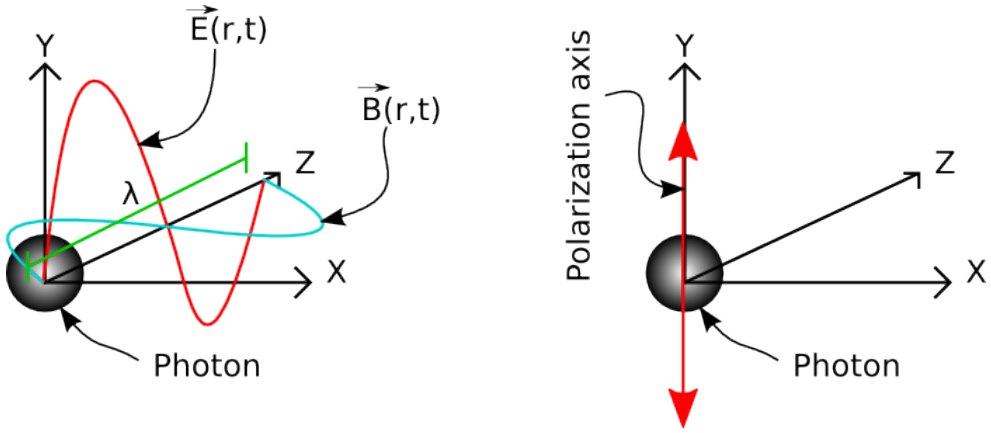


Figure 1.2: Sketch on the left hand side illustrates a photon with a electromagnetic field , sketch on the right hand side highlights the polarization state of the exemplary photon.

The electric field of a photon can vary in space and time. Thus, different types of polarization states exist for a photon. Figure 1.3 shows two different commonly used polarization states: linear and circular polarization.

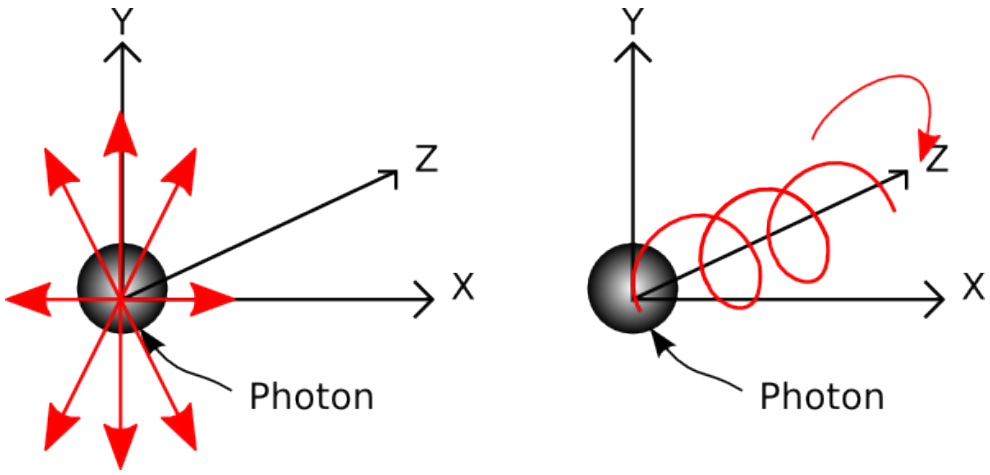


Figure 1.3: Sketch on the left hand side illustrates a photon with linear polarization , sketch on the right hand side shows a photon with a circular polarization.

The polarization state of the light provides light selectivity in many different applications of 3D displays. In such applications, the polarization filters either block or pass the light using polarization property. Overall, photons from the same source can build up to form different types of beams. A beam can be written in the form of a superposition of plane waves with different magnitude, phase, and direction. The exemplary photon introduced in the figures can be written in the form of a general plane wave. In that case, electric field of the exemplary photon will be as in Equation 1.1.

$$\vec{E}(\vec{r},t) = (A_x \times \cos(kz - \omega t), A_y \times \cos(kz - \omega t + \phi), 0) \tag{1.1}$$

The beam can be crafted in a certain way that the light is visible to the observers at a certain desired space. This is another trick used in 3D display applications. Thus, light can be focused at a certain region by simply designing each component of the beam in means of direction, magnitude and phase. Through this trick, different perspective images can be visible in different places in space. A viewer can perceive 3D by observing at least two of these perspective images.

The light selectivity can also be performed by using the color of light. Under

Equation 1.1, the wavelength λ -the color- of the light is embedded inside the constants of $k = 2\pi/\lambda$, and $w = 2\pi \times f$. The frequency f of the light is proportional to the wavelength through the formula of $\lambda = c/f$, in which c represents the speed of the light in the medium that it is present. Wavelength often used in 3D displays as the light selectivity parameter. Again in such applications, the wavelength filters either blocks or passes the light depending on the wavelength. Further detailed reading on the nature of light is available under [9, 10].

1.2 Laser scanning pico projector

Our core technology: laser scanning pico projector is an off-the-shelf device from Microvision, Incorporated (USA). The collaboration between our laboratory, Optical Microsystems Laboratory (OML), and Microvision since 2002 lead to a number of innovations in MEMS scanners, display architectures, and micro-structured screen technologies. The research and development effort resulted in commercial product: Microelectromechanical System (MEMS) scanner and laser based pico projector. The device combines linearly polarized three different laser light sources, in this case corresponding to red, green, and blue colors, into a single white light source. Combined beam is sent to a MEMS scanner. The scanner scans the beam in two axes using a raster scan pattern. Thus, an image from the pico projector can be formed on any scattering surface.

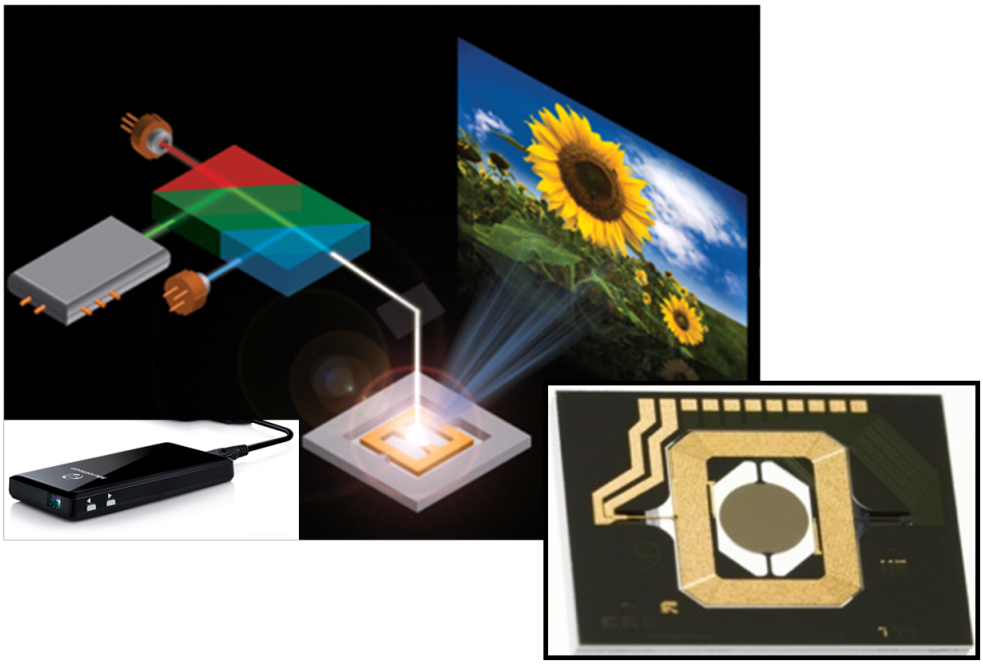


Figure 1.4: On the left, a photograph of PicoP from Microvision,Inc is shown with a sketch of the interior photonics module. On the right, MEMS scanner is shown [1].

The laser scanning pico projectors do not include any internal optics inside. Thus, a focus free projection is obtained through this technology. Each laser source from the pico projector is nearly collimated. These sources behave as a Gaussian beam traveling in free space with a beam waist of 0.5 mm in the perpendicular axis, 0.3 mm in the parallel axis. The beam waist is reached, when the beam from the photonics module of the projector travels a distance of 500 mm. The photonics module of the laser scanning pico projectors are nearly at the same size of a coin as shown in the photograph under Figure 1.5.

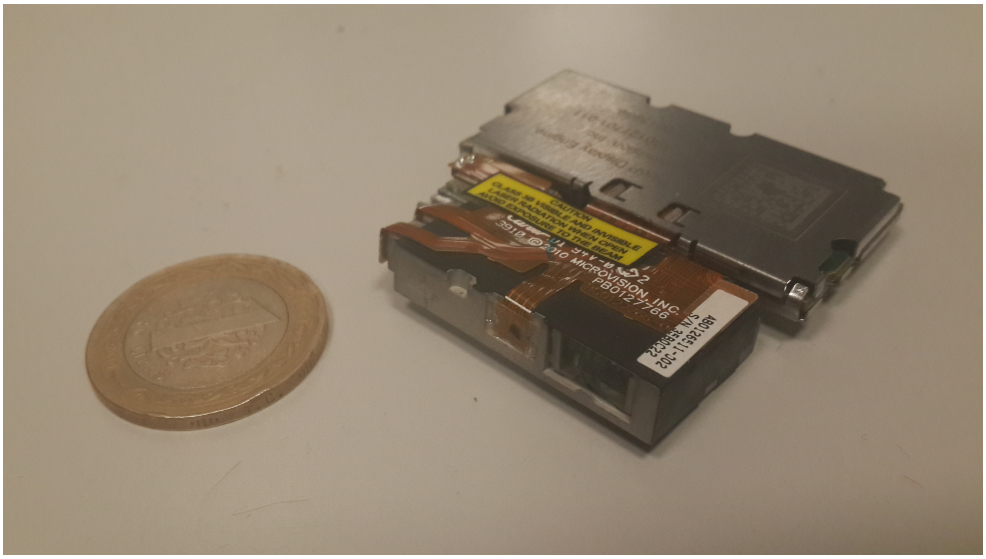


Figure 1.5: A photograph of the photonics module of the laser scanning pico projector from Microvision [1].

The size of the photonics module serves as a good core to miniaturize existing 3D displays. On the other hand, polarization characteristics of the laser light sources provide a light selectivity through polarization for 3D display applications. Collimated beam characteristic provides true directionality and focus free design, which plays an essential role in true 3D vision.

1.3 Human visual system

Healthy human beings are equipped with a pair of eyes, a sensory system for viewing the surrounding. The eyes are separated from each other with a certain interpupillary distance (IPD) [11]. Typically, an adult human being has an IPD of 65 mm. Thus, each eye has the capability to observe surroundings from a slightly different perspective determined by the interpupillary distance. In another aspect, it can be thought as each eye seeing totally different images. This effect can be easily observed while looking at a certain object in motion, and observing the object with each eye stand-a-lone. The observed object would probably be seemed slightly shifted when two images are compared, or so to say with different disparities. But there are still exceptions, where

objects at certain distances appear to be at the same point on both eyes. This is called as horopter or in another saying range of depth. The points appeared to be imaged at the same spot forms a circular boundary as in Figure 1.6. Note that the sketch provides a theoretical horopter using the ideal case and simple geometry. There has been empirical studies [12] investigating on physical shape of a real horopter of our visual system.

A careful reader would immediately recognize the rotation of the eye balls under Figure 1.6. The rotation of the eyeballs is entitled as vergence in the literature. When two eyeballs focus on the same object, they converge by inward rotation towards the object. The angle of convergence is larger when the eye is fixating on nearby objects. Change of focal length of the eye (accommodation) to focus at different depths provides information about the depth. Both the accommodation and convergence depth cues require the use of eye muscles and provide neural feedback about the depth of the object. Under Figure 1.6, a cross-vergence case is sampled, where eye balls are rotated towards each other. For a careful reader, it is easy to realize that with the changing vergence, horopter can take different sizes and shapes.

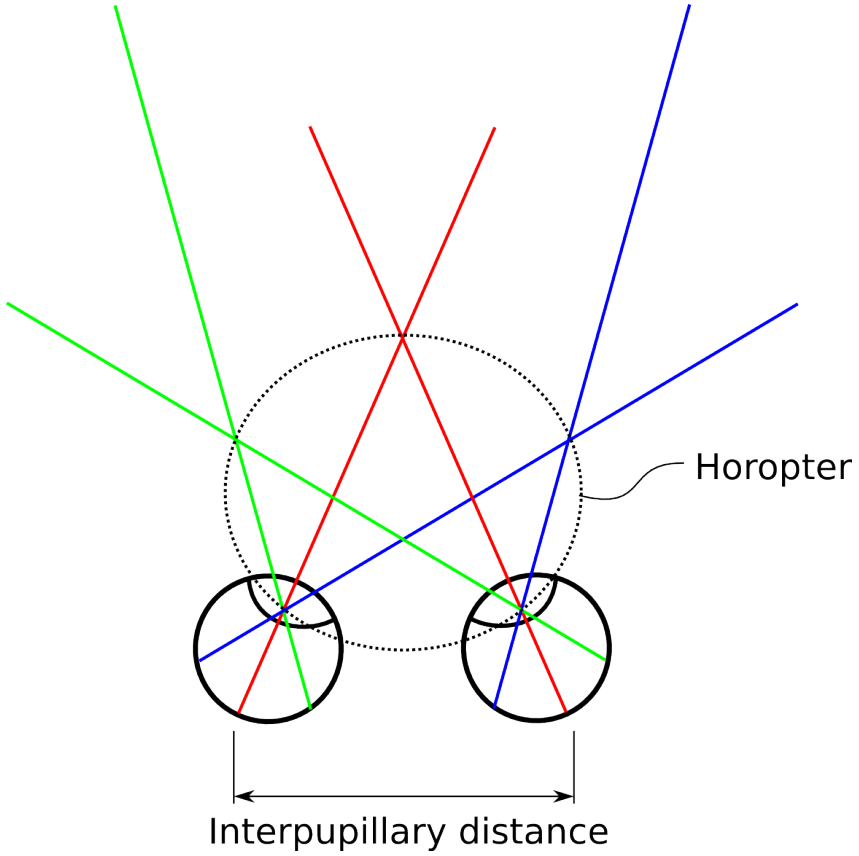


Figure 1.6: Sketch showing the interpupillary distance (IPD) and the theoretical horopter.

3D displays generally provide only a stereo image pair to two eyes. As a consequence, the image appears focused only at the display screen and the eyes fixate on the display screen rather than the correct depth of the object. Fixating at the wrong depth forces the eyes to an unnatural viewing condition by decoupling the accommodation and convergence. The conflict between the decoupled accommodation and convergence has been implicated as a cause of perceptual error (failure of size constancy), viewing discomfort, and visual system changes in all stereoscopic displays (including 3D TVs, 3D cinema).

A single eye of a human being has an eye pupil entrance, which can be thought as an aperture. A typical eye pupil diameter d_{eye} in the horizontal axis is around $2\ mm - 8\ mm$. The eye has the mechanism to adapt the aperture diameter according

to the illumination level of the surrounding [13]. A typical human eye has a focal length of 17 mm, when focused at infinity. Thus, optical power of $1/17\ mm = 60\ diopters$ is a typical value for a human eye. Rayleigh resolution criterion dictates a changing resolution limit with the changing eye pupil entrance. Angular resolution can be calculated as $sin\theta = 2.44 \times \lambda \times F\#$ according to Rayleigh's resolution limit, in which λ presents the wavelength of the light, and $F\#$ is the F-number, which is the ratio between focal length of the system and the limiting aperture size. A typical angular resolution value of an human eye is $\theta = 0.000316°$, when $f = 17\ mm$, $d_{eye} = 4\ mm$, and $\lambda = 532\ nm$ are used. Note that this is a rough calculation, true resolution characteristics can be read from Modular Transfer Function (MTF) curve of the human eye [14], which takes into account the effects such spherical aberrations, astigmatism, or coma.

On the other hand, the sensory cells found in the human eyes have different properties in terms of light sensitivity and response time to the light. This property of the human eye had been a foundation for Pulfrich effect [15], in which response time of the eye plays an important role on detection of depth through lateral motion of the viewer. All explained concepts play an important role on three dimensional (3D) vision. Yet, there are still parameters that take important role on 3D vision for human visual system. The human beings have the capability to benefit from the past experiences to guess the size and the depth information of the objects.

1.4 Three Dimensional Display systems

The human visual system can be fooled to perceive virtually created 3D objects as real 3D objects taking into account of explained concepts. The art of providing a pair of imagery to fool human eyes in terms of 3D perception is called as stereoscopy. Stereoscopy has been in use for more than a century, Stereoscopic vision theory was first proposed in 1852 by Sir Charles Wheatstone, [2]. His first invention on the field is a device composed of mirrors, and two perspective images. With the help of his mirrors, he was able to direct each image to a single eye as in Figure 1.7. The basic

understanding lying beneath the existing 3D displays in movie theaters still relies on this foundation.

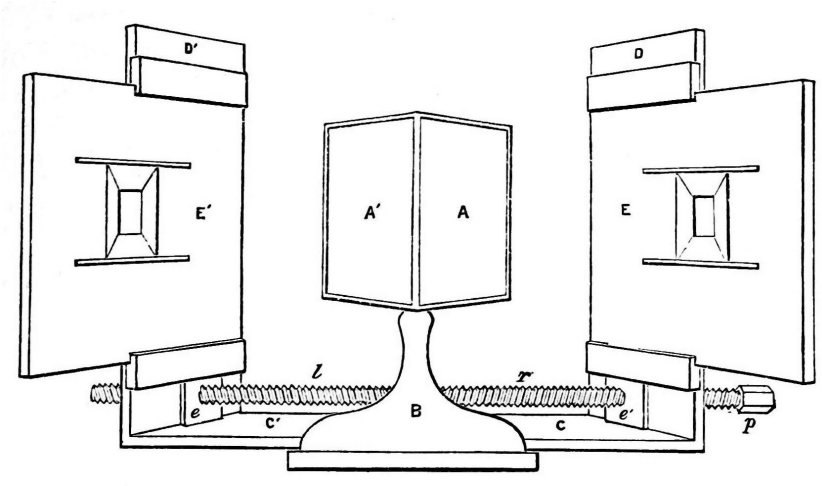


Figure 1.7: Sketch showing the invention of first stereoscopic display, which was built by Sir Charles Wheatstone [2].

From the beginning, many different techniques were derived to provide stereoscopic imagery [16]. Most frequently used stereoscopy methods require special aids in front of the viewers' eyes. In most cases, the required aid is in the form of binoculars. Some of these binoculars benefits from the polarization state of the light, or wavelength of the light. Besides, light selectivity can also be achieved through time multiplexing, which is another form derived from the polarization techniques. Alternatively, contact lenses [17, 18] can also be used as a replace to binoculars for 3D vision systems. But there are also other possibilities, which do not require any special aids in front of the viewer's eyes. This case is generalized as autostereoscopic displays. There are many different forms of autostereoscopic displays, to name a few: Light field displays, Super Multi View displays, Holographic displays, and similar. A simplistic categorization of 3D displays can be found under Figure 1.1. The contribution of this research is marked under Figure 1.1, note that two different techniques were added to the literature. Besides, activities within this research related to the existing technology is also marked under Figure 1.1. The display types introduced at this paragraph will be detailed

within the corresponding sections.

Chapter 2 introduces two different types of 3D display method using an eye-aid. The first introduced method under Section 2.1 is named as Mixed Polarization technique, in which a novel scheme is introduced to project 3D images using a pico projector. The second method under Section 2.2 introduces a new method entitled as Super Stereoscopy technique, which improves depth information on both stereoscopic and autostereoscopic displays. Section 2.3 introduces an augmented reality application using head mounted pico projectors

Chapter 3 introduces two different types of 3D display methods, which does not require any use of special eye-aid. The first method under Section 3.1 introduces a single viewer autostereoscopic display, in which a pair of pico projectors, a rotating screen consists of a retro-reflective foil with a vertical diffuser on top, and an eye tracker are used. The second method under Section 3.2 introduces a modular multiviewer autostereoscopic display, in which an array of pico projectors are used among with a vertical diffuser as the screen. In both methods, a viewer perceives a different perspective from the screen according to his/her observation point in front of the screen.

Finally, the last Chapter 4 summarizes the contribution in a combined manner. Additionally, an appendix available under Chapter 5, which consists of algorithms for different purposes such as: the control algorithm of the multiviewer multiview display, conversion algorithm of Mixed Polarization technique and simulation of voxel shape in Super Stereoscopy method.

1.5 Contribution to the literature during PhD studies

This section summarizes the contributions of this thesis to the literature. Additionally, other relevant publications within my PhD studies are also mentioned. At the end of this thesis, readers can find a bibliography entitled as "Publication Record" citing all the publications within my PhD studies including those that are not directly related to this thesis.

The technique introduced under Section 2.1.6 was published as a journal paper [K1]. The technique introduces flicker free single projector stereoscopy method at low refresh rates. A demonstrator was built using a pico-projector and a polarization rotator among with a polarization maintaining screen. The technique was also presented in different conferences [K2–K4]. We received two international awards for this work: "Young scientist award" prior to the conference publication [K2] by the Merck group in Korea, and "Best 3D product of the year" award by the International 3D society in USA. On the other hand, the introduced technique under Section 2.2 is not yet submitted as a journal publication. Though it was presented in an international conference [K5], and filed as a patent. Additionally, an augmented reality application is derived as an Augmented Reality system using head-worn pico-projectors. This technique is demonstrated and presented in an international conference [K7] as well.

The introduced autostereoscopic single viewer display under Section 3.1 was published as a journal paper [K8]. The display provides a head-tracking autostereoscopy solution for a single viewer with a low level of complexity. The display provides bright images due to high light gain characteristics of the retro-reflective screen. The display method is also presented in different international conferences [K4, K9, K10]. The output of our work appeared on Optics Society of America's Spotlight on Optics. Under Section 3.2, a Multi-viewer Autostereoscopic display is presented and demonstrated. It is planned to publish the outcome of this work as a journal paper. The proposal is already presented in different international conferences [K11, K12].

Overall, 3 journal papers and a magazine paper were published, 3 patents were filed, 15 conference papers were presented. Two more journal papers on Section 2.2 and Section 3.2 are in the preparation.

Chapter 2

STEREOSCOPIC DISPLAYS

This chapter focuses on our scientific contribution in the domain of stereoscopic displays, which require certain aid such as glasses, binocular, or spatial apertures. The chapter contains two different methods of stereoscopy, where each one provides unique opportunities in their own use-cases. Additionally, an augmented reality display application using head mounted pico projectors is introduced.

2.1 Portable 3D Laser Projector Using Mixed Polarization Technique

The section introduces a new stereoscopy method for projectors with low refresh rates, which we have published previously under [19]. The idea was first initiated in the weekly teleconferences in between Microvision, Inc. located at Seattle, U.S.A., and Optical Microsystems Laboratory (OML) located at Koç University, Istanbul, Turkey. The main attendees of these teleconferences are Mark Freeman, Hakan Urey, and Kaan Akşit. The idea was first tested by Kaan Akşit. Later, a M.Sc student, Osman Eldeş joined the project. He contributed in the experimental work related to the prototype. Selvan Viswanathan, a research engineer at Microvision, Inc., provided the necessary code to get the required signal from Pico projector's core processor. This part of our research was funded by Microvision Incorporated. SilverFabric-UMA GmbH had provided sample silver screen elements to be used in the setup bench.

2.1.1 Introduction

The most common stereoscopic display approaches are color-multiplexed (using anaglyph glasses), polarization-multiplexed, and time-multiplexed techniques, which are reviewed in [16]. In the anaglyph method, the viewer wears a pair of colored

glasses so that the left and right eyes each receive non-overlapping parts of the visible spectrum, with different stereo views of the 3D scene coded in each partial spectrum. From this information, the brain synthesizes a full-color 3D scene. Several types of anaglyph color schemes are available, the most common being red-cyan color colored glasses [20], [21] and [22]. The main drawbacks of this type of display are the loss of color information, challenges in obtaining the correct color balance, binocular rivalry [23] and reproduction, and the increased degree of crosstalk between the two eyes [23], [24].

In the polarization-multiplexed approach, the polarization state of the light corresponding to each image in the stereo pair is made mutually orthogonal. One common setup of polarization-multiplexed approach consists of passive polarization glasses, a projector with a liquid crystal polarization rotator with at least 120 Hz refresh rate, and a polarization maintaining screen. A straight-forward way of building a polarization-multiplexed setup is to employ two projectors with projected light in orthogonal polarizations, and one projector for left-eye images and the other projector for the right-images. Alternative operation modes of this technique are described in [16]. Passive glasses with orthogonal polarizers for the left and right eyes blocks the image not intended for that eye. A single projector operated at a minimum of 120 Hz refresh rate with an active polarization rotator to rotate the polarization in between frames has also been demonstrated [25], [26], [27] and is commercially available for 3D TVs. The main drawback of these schemes is the increased frame rate and the blanking time required between switching cycles to reduce the crosstalk. Patterned micro-polarizers and patterned retarders have also been employed, particularly for 3D TVs, to achieve polarization multiplexing at the expense of resolution [28], [29].

Table 2.1: Comparison of stereoscopy methods (Assuming minimum 60 Hz per eye refresh rate and a flicker free image).

Method	Hardware options	Pros / Cons
Anaglyph	60 Hz projector; Any surface; Color coded glasses	(+) Simple hardware, (+) Standard refresh rate, (+) passive glasses, (-) Loss of color
Polarization multiplexing	120 Hz projector + polarization rotator (or two 60 Hz projectors); Polarization maintaining surface; Passive polarized glasses	(+) Passive glasses, (-) High refresh rates (or image registration problem), (-) Special screen requirement
Time multiplexing	120 Hz projector; Any surface; Active shutter glasses	(-) Active glasses are expensive, require battery, (-) High refresh rate
Mixed polarization	60 Hz projector and polarization rotator; Polarization maintaining surface; Passive polarized glasses	(+) Standard refresh rate, (+) Passive glasses, (-) Special screen requirement

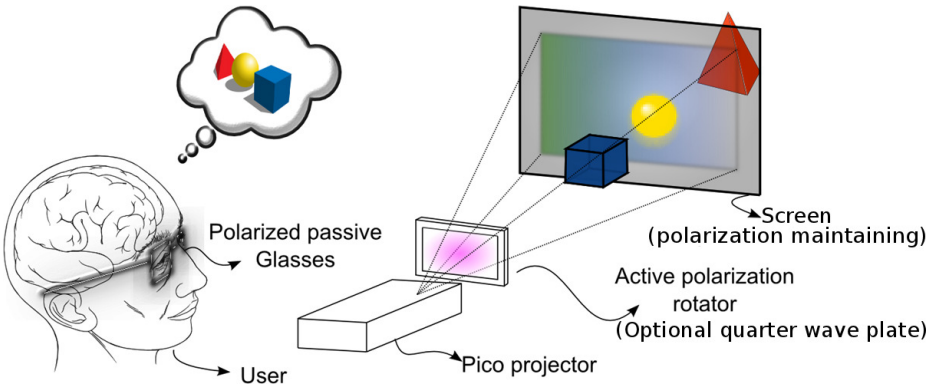
In the time-multiplexed approach, the left and right eye images are displayed on the screen in an alternating fashion. Each eye receives light only on alternate frames which leads to flicker unless the frame rate is increased to at least twice the frequency where flicker is not perceived. Those displays are typically operated at 120 Hz or greater. The viewer is required to wear battery-powered active shutter glasses, which are synchronized to the content being displayed, [30]. Active glasses are more costly, heavier than inexpensive passive glasses, and require careful handling. All of the described techniques with their hardware options, advantages and disadvantages are tabulated in Table 2.1.

This work introduces a new method for 3D projection displays one that has similarities to the existing methods discussed above in that it utilizes polarization and color-multiplexing to present different stereo 3D perspectives to each eye, but by combining the use of polarization and color, it avoids the weaknesses associated

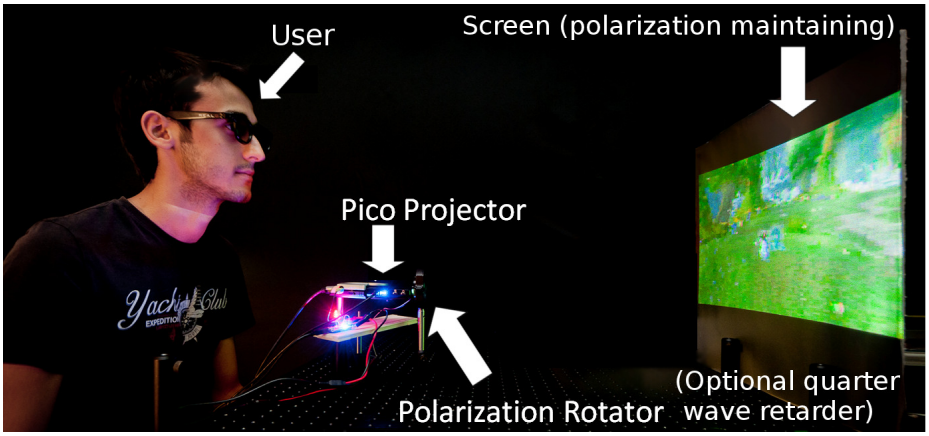
with previous methods. This new method is named mixed-polarization 3D. It is particularly aimed at use in a scanned laser projectors where all three primary colors (R,G,B) are simultaneously displayed using already polarized lasers [31]. Microvision's PicoP^{TM} hand held pico projector is a good example of this type of scanned laser projector [32]. Section 2.1.2 discusses the key components and provides Jones calculus analysis of the system, Section 2.1.3 discusses the content creation, Section 2.1.4 discusses the test results and Section 2.1.6 discusses the polarization maintaining screens.

2.1.2 Concept of the display

The system contains four different elements as illustrated in Figure 2.1a: a scanned laser pico projector, an active polarization rotator coupled to the projector, a polarization maintaining screen and passive polarized glasses.



(a)



(b)

Figure 2.1: (a) Single pico projector 3D display system. (b) A photograph of the overall system during operation.

The scanned laser light from the pico projector passes through an active polarization rotator and then is incident on a polarization maintaining screen. A user with polarized glasses views the light beams reflected from the polarization maintaining screen such as a silver screen. Figure 2.1b shows the latest prototype from our test-bench which is adaptable to both linearly polarized scheme and circularly polarized scheme, which require additional quarter wave plates.

The pico projector contains red, green, and blue laser light sources, a MEMS scanner and drive electronics. Figure 2.2 illustrates layout of the laser module and the associated combiner optics that result in different polarization states in the combined beam. The red and blue lasers are shown to be polarized perpendicular to the plane

of the figure while the green laser is polarized in the plane of the figure. The current bench top prototype system shown in Figure 2.1b has the polarization rotator after the pico projector, but it can also be embedded in the system before the MEMS scanner as shown in Figure 2.2. In the embedded case, the polarization rotator acts on the beam before it is scanned by the MEMS scanner, the beam passes through it at fixed angle of incidence, which improves the performance of the polarization rotator. Also, by operating on the pre-scanned laser beam, the size and cost of the polarization rotator can be minimized.

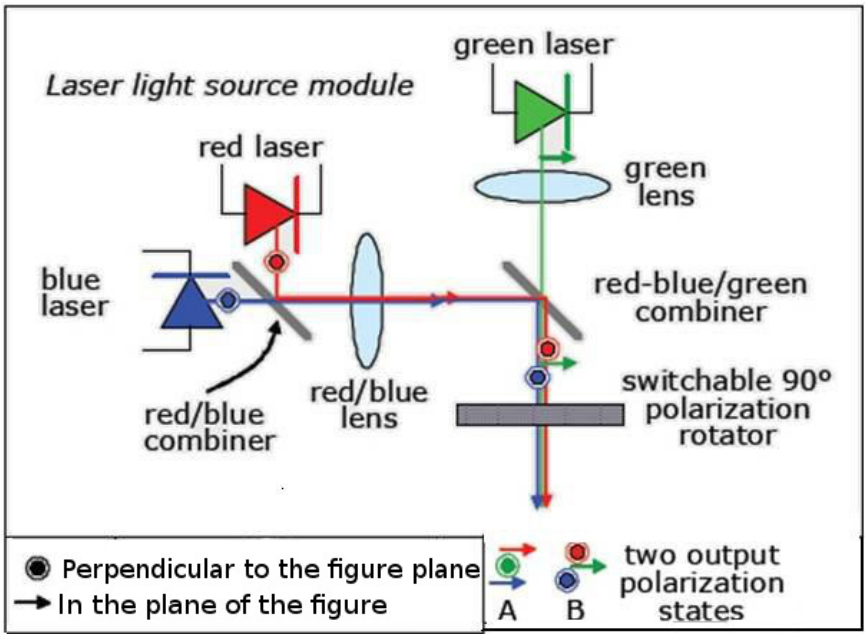


Figure 2.2: System sketch showing laser light source module with two orthogonal polarization states.

The active polarization rotator is synchronized with the frame rate of the projector such that it switches state during the off time between frames (i.e. scanner retrace time). Thus, mixed polarization states of lasers are alternated for successive frames with red and blue having in-plane polarization and green having out-of-plane polarization in frame i, and then flipping in frame i+1 so that red and blue have out-of plane polarization and green has in-plane polarization. Any type of polarization rotator could be used; we chose a ferro-electric liquid crystal rotator. It acts as a half-wave

retarder with its fast axis lined up with the polarization in the ON state and rotated by 45 degrees in the OFF state. The incident light is therefore unaffected in the ON state and rotated by 90 degrees in the OFF state. The polarization rotator is synchronized with the refresh rate of the pico projector which is typically around 60 Hz.

Table 2.2: Polarization Vectors at Different Stages without a Quarter Wave Plate (Fast axis of the polarization rotator at 45° angle with laser axis is used as the reference coordinate axis to avoid matrix rotations in the Jones Calculus).

	Jones Matrix	Frame i		Frame i+1	
		Red & Blue	Green	Red & Blue	Green
After Laser Combiner	-	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$
Polarization Rotator (R)	$\begin{cases} L_{R_i} = \begin{bmatrix}0 & 1\\-1 & 0\end{bmatrix} \\ L_{R_{i+1}} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \end{cases}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$
Polarization Maintaining Screen (PMS)	$L_{PMS} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$
Left Eye (Left eye linear polarizer, LLP)	$L_{LLP} = \begin{bmatrix}\frac{1}{2} & \frac{1}{2}\\\frac{1}{2} & \frac{1}{2}\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$
Right Eye (Right eye linear polarizer, RLP)	$L_{RLP} = \begin{bmatrix}\frac{1}{2} & -\frac{1}{2}\\-\frac{1}{2} & \frac{1}{2}\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$

Table 2.3: Polarization Vectors at Different Stages without a Quarter Wave Plate (Fast axis of the polarization rotator at 45° angle with laser axis is used as the reference coordinate axis to avoid matrix rotations in the Jones Calculus).

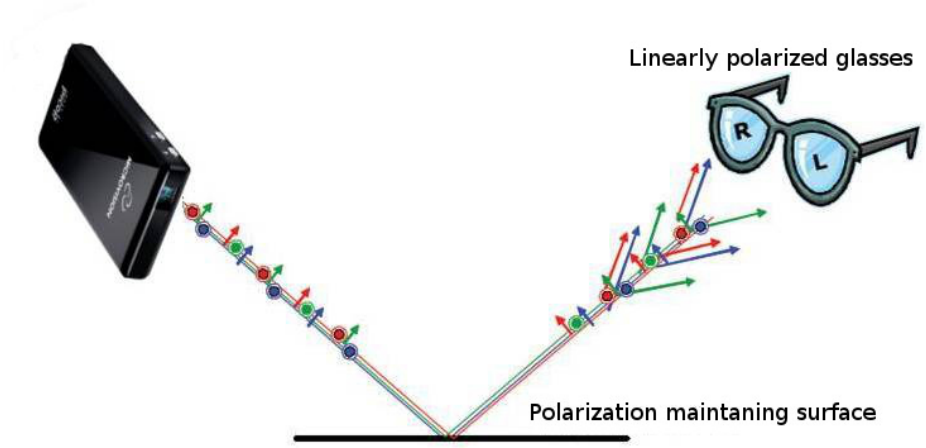
	Jones Matrix	Frame i		Frame i+1	
		Red & Blue	Green	Red & Blue	Green
After Laser Combiner	-	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$
Polarization Rotator (R)	$\begin{cases} L_{R_i} = \begin{bmatrix}0 & 1\\-1 & 0\end{bmatrix} \\ L_{R_{i+1}} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix} \end{cases}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\-1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}$
Quarter Wave Plate (QWP)	$L_{QWP} = \begin{bmatrix}1 & 0\\0 & -j\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\j\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\j\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-j\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\j\end{bmatrix}$
Polarization Maintaining Screen (PMS)	$L_{PMS} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\j\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\j\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-j\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\j\end{bmatrix}$
Left Eye (Left handed circular polarizer, LHCP)	$L_{LHCP} = \frac{1}{2}\begin{bmatrix}1 & -j\\j & 1\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\j\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\j\end{bmatrix}$
Right Eye (Right handed circular polarizer - RHCP)	$L_{RHCP} = \frac{1}{2}\begin{bmatrix}1 & j\\-j & 1\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}-1\\j\end{bmatrix}$	$\frac{1}{\sqrt{2}}\begin{bmatrix}1\\-j\end{bmatrix}$	$\begin{bmatrix}0\\0\end{bmatrix}$

The mixed-polarization method introduced in this work takes advantage of the perpendicular relationship between polarization state of the red and blue lasers and that of the green laser in the Microvision PicoP display engine. The polarization states of this system can be analyzed using Jones calculus, [33]. The analysis is carried out for the green channel and the blue/red channels separately and tracks only the polarization direction without taking the beam power and efficiency of the components into account. The method is suitable for two different approaches: (i) using linear polarization illustrated in Figure 2.3a and (ii) using circular polarization illustrated in Figure 2.3b. In the circular polarization case, QWPs are placed to convert the linear polarization into circular polarization and vice versa.

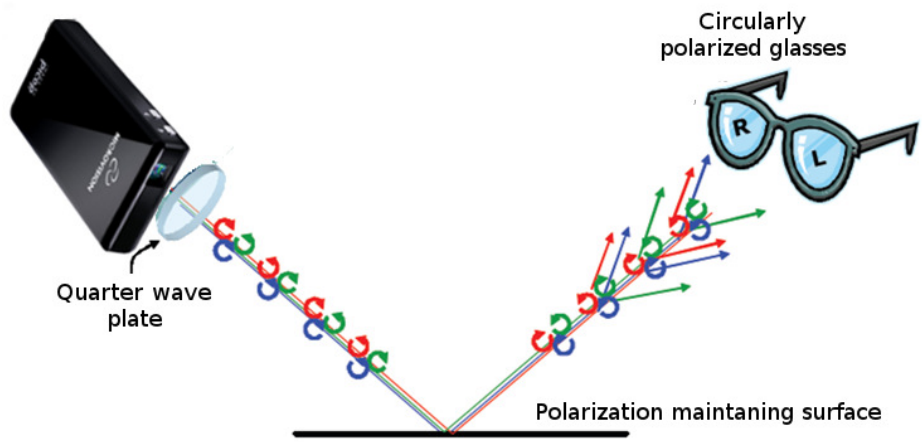
Tables 2.2 and 2.3 show the Jones matrices of the system components and the Jones vectors for the all color channels after each component in the system. The fast axis of the QWP in circular polarization and the polarization rotator case are identical and chosen to be the reference axes of the whole system to avoid rotation matrices in the analysis and to provide ease of understanding.

A polarization maintaining diffuser screen is necessary to avoid depolarization of the light. The reflected light from the screen arrives at the polarized glasses. In the linear polarization case, the passive glasses contain two linear polarizers with perpendicular polarization states at $\pm 45^{\circ}$ angles to the coordinate axis. In the circular polarization case, the passive glasses contains left and right handed circular polarizers. Circular polarization is generally preferred as it allows more head tilt without crosstalk. The passive glasses act as polarization analyzers and separate left and right eye images. The polarization filters are assumed ideal without any cross talk.

As a result, the left eye receives the green channel, while the right eye receives the red and blue channels in one frame. In the next frame, the left eye receives the red and blue channels while the right eye receives the green channel.



(a)



(b)

Figure 2.3: (a) Polarization state of the light at different stages. (b) Polarization state of the light at different stages with a quarter wave plate.

2.1.3 Content creation

The operating scheme requires specially encoded video or image sources to provide stereoscopic vision. Conventional frame-compatible stereoscopic content contains two images per frame, one for each eye. In the mixed polarization scheme, the green channel is interchanged between the stereo image pairs to create a modified stereoscopic content. Figure 2.4 illustrates the required content modification.

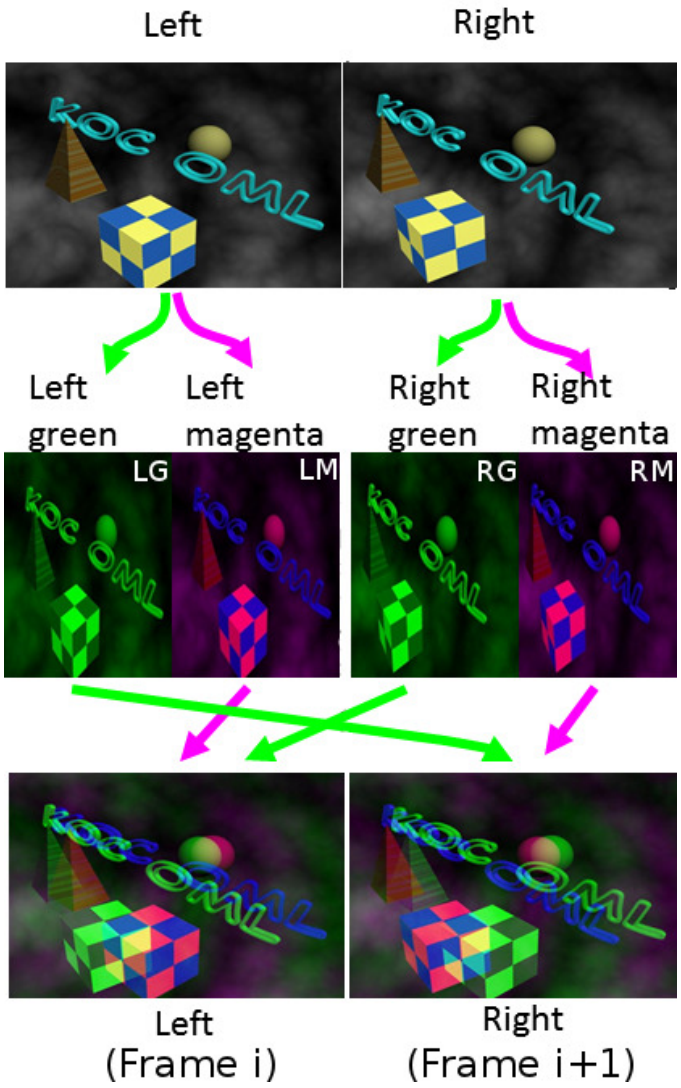


Figure 2.4: Demonstration on how single frame is modified.

Note that the conversion operation is straight-forward and can be performed in real-time. We modified the content with the help of scripts developed both in MATLAB and Python. Thus the conversion can run across different operating systems and platforms. Figure 2.5 explains how a user receives the modified stereoscopic content with the glasses.

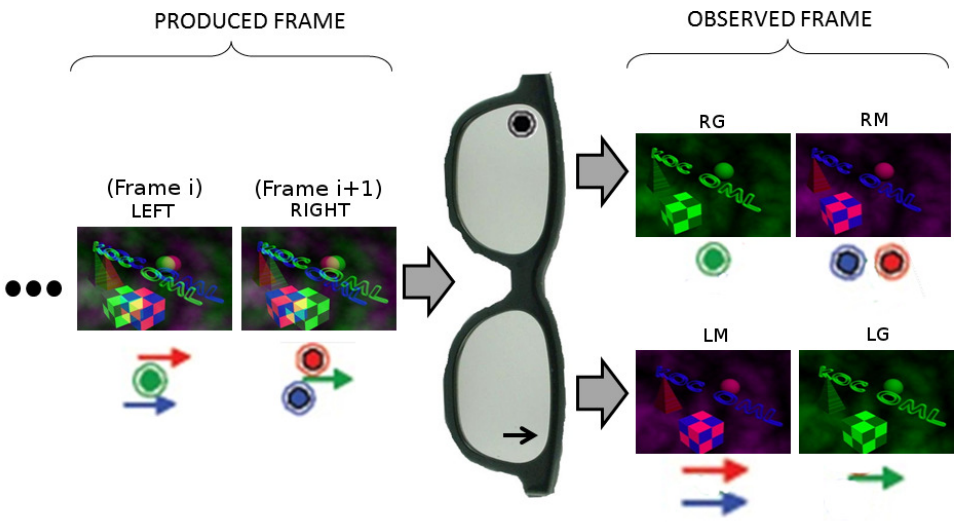


Figure 2.5: Frames produced in Figure 2.4 are displayed as demonstrated above.

2.1.4 Results and Discussions

The display system shown in Figure 2.1b was built to demonstrate the mixed polarization 3D approach. The elements used are a scanned laser pico projector from Microvision [1], an active polarization rotator from Micron Tech. [34] (driver model DR-95) coupled to the projector, a polarization maintaining screen from SilverFabric-UMA [35] and passive polarized glasses. The 3D content shown in Figure 2.6 is convincing and high-quality. Even though this is a 60 Hz projector that provides separate images to both eyes, there is no noticeable flicker in the display. When the images are projected on a partially reflective micro lens array screen [36] with polarization maintaining property, then the image quality is even better due to lower speckle contrast.

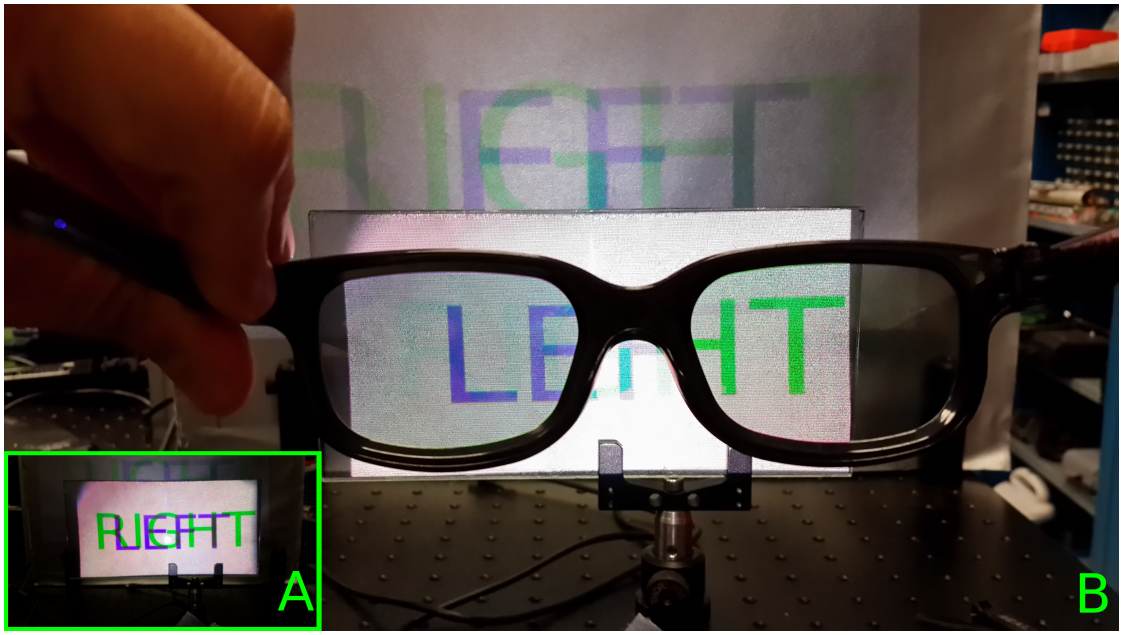


Figure 2.6: A set of photographs showing (a) superimposed left and right images seen with a naked eye, (b) a demonstartion with glasses, in which each eye see totally different image with a low crosstalk.

Cross talk performance of a stereoscopic display is an important measure in providing true stereoscopic vision to the users. We measured crosstalk using the test bench illustrated in Figure 2.7. A portion of the projected content goes through the polarization rotator and the linear polarization filter. The transmitted power is then measured using a power meter from Edmund Optics.

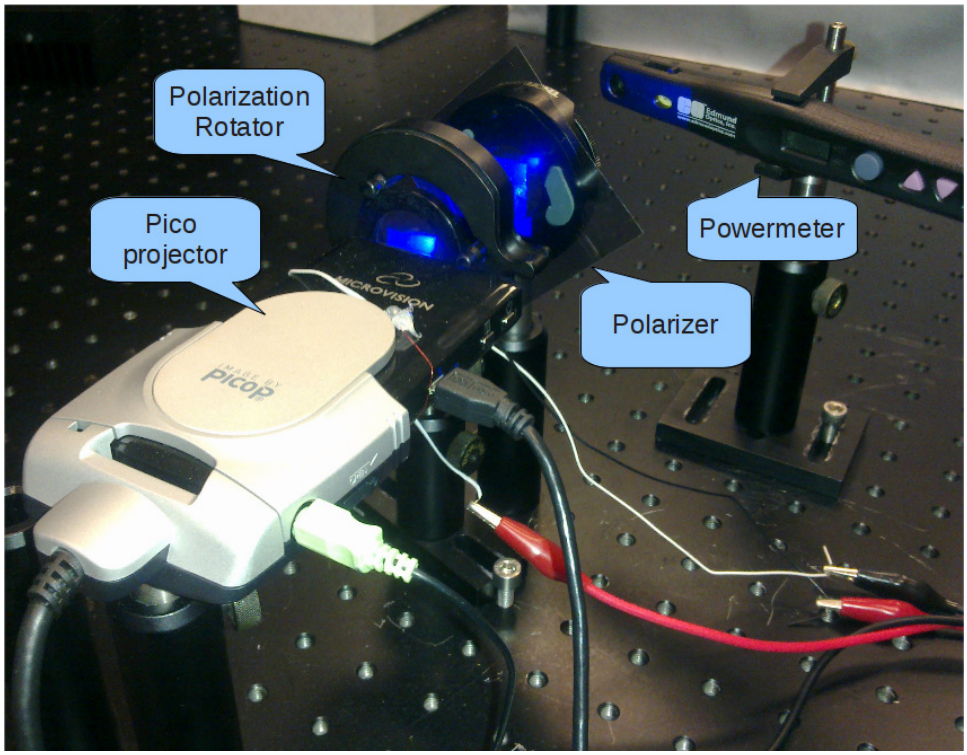


Figure 2.7: A picture of the experiment bench.

The experiment is repeated for each color channel. Note that the effect of the polarization maintaining screen is neglected for this experiment and the polarization maintaining screen is assumed to be ideal. Table 2.4 presents the measured values from the power meter. During the experiment, it is observed that the polarization states of green and red color channels are orthogonal to each other. But the polarization state of blue has an additional angular difference of 4° with respect to red. Photometric weights of red and green dominate over blue and therefore the polarization rotator and the linear filters should be aligned with respect to red and green to achieve good crosstalk attenuation levels.

Table 2.4: Power measurements ("None" represents the case when polarization rotator is removed from the beam path.).

Rotator state	Polarizer	Red only	Green only	Blue only
On	Left	82 μW	1.11 μW	35.5 μW
On	Right	4.76 μW	62.5 μW	0.6 μW
Off	Left	8.48 μW	59.7 μW	0.47 μW
Off	Right	76.1 μW	3.17 μW	35.1 μW
None	Left	91.7 μW	1.2 μW	47.9 μW
None	Right	3.32 μW	75 μW	1.45 μW

The radiometric crosstalk ratio is the direct ratio of power between the unwanted and wanted content. For example with the rotator in the "ON" state and with the left eye polarizer, the radiometric crosstalk ratio is 1.11 $\mu W/(82\ \mu W + 35.5\ \mu W) \times 100 =$ 0.9 %. The photometric weights of the used wavelengths are 23.2 % for red, 75.0 % for green and 1.8 % for blue (RGB: 643 nm, 530 nm and 446 nm respectively). The photometric crosstalk ratio is the ratio of undesired light power to desired light power for a particular state of the rotator and for a particular eye [37]. For example with the rotator in the "ON" state and the left eye polarizer, the photometric crosstalk ratio is 1.11 $\mu W \times 0.75/(82\ \mu W \times 0.23 + 35.5\ \mu W \times 0.02) \times 100 = 4.0$ %. Table 2.5 presents the full set of calculated cross-talk ratios.

Table 2.5: Cross-talk levels: ratio of desired to undesired light powers.

Rotator state	Polarizer	Photometric crosstalk ratio	Radiometric crosstalk ratio
On	Left	4.0 %	0.9 %
On	Right	2.4 %	8.6 %
Off	Left	4.5 %	15.0 %
Off	Right	12.5 %	2.9 %
None	Left	3.9 %	0.9 %
None	Right	1.5 %	6.4 %

From Table 2.5, average cross talk levels during the operation are calculated as

4.3% for left eye and 7.5% for right eye using the photometric values. In [38] it is stated that the image distortions caused by crosstalk percentages up to 5 % are hardly noticeable and the same source suggested that ideally the designers should keep the crosstalk below 2 %. The degree of polarization can be defined as [39]:

$$Q = \frac{|I_{max} - I_{min}|}{I_{max} + I_{min}} \tag{2.1}$$

If the polarization rotator is in the "ON" state, the Q for red color is calculated as 0.89 using $I_{max} = 82\ \mu W$, and $I_{min} = 4.76\ \mu W$ from Table 2.4. Table 2.6 shows degree of polarization (Q) values for each channel. Note that the polarization rotator is optimized for green laser.

Table 2.6: Degree of polarization for overall system.

Rotator state	Red	Green	Blue
On	0.89	0.97	0.97
Off	0.80	0.90	0.97
None	0.93	0.97	0.94

In Table 2.4, power of red in the "ON" state and the "OFF" state for left eye are 82 μW and 8.48 μW, respectively. Therefore, the contrast ratio of this case can be defined as $10 : 1$. The contrast ratios for each color channel is shown in Table 2.7.

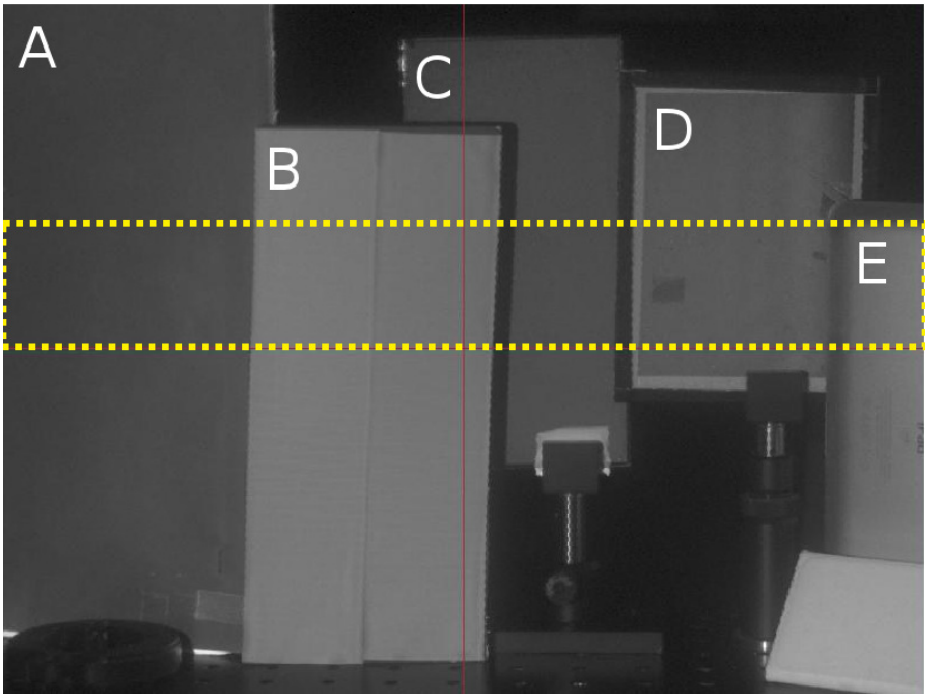
Table 2.7: Contrast ratio for each color channel: the ratio of switched-ON to switched-OFF brightness.

	Red	Green	Blue
Left	10:1	54:1	75:1
Right	16:1	20:1	59:1

The prepared test benches both in Microvision and in Koç University have been tried by many testers (> 100 people) and all of the testers have stated that the quality level of the stereoscopic display is good and satisfactory.

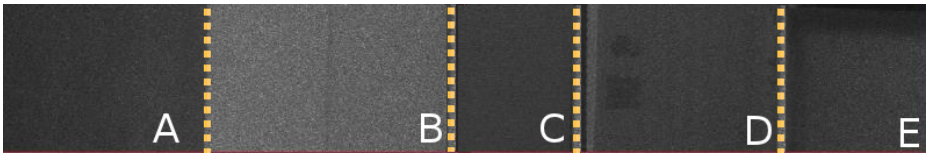
2.1.5 Polarization maintaining surfaces

We also performed experiments comparing various surfaces for their polarization-maintaining properties. Sometimes people have the mistaken impression that the polarization-maintaining "silver screen" for observing 3D is expensive. Here, we demonstrate that this need not be the case. Different surfaces such as micro lens array screen, duct tape, a commercial silver screen and metallic laptop surface were illuminated with linearly polarized light (using a single color channel of the pico projector, see Figure 2.8b). Another photograph shown in Figure 2.8c is taken with a linear polarizer in front of the camera which is orthogonal to the polarization state of the illumination source. Illumination values of the cases in Figure 2.8 are tabulated in Table 2.8. A zoomed out version of Figure 2.8b is depicted in Figure 2.8a. As seen in Figure 2.8, duct tape, silver screen and metallic laptop surface are sufficient for this technique. Likewise, one can also use non-scattering surfaces such as micro lens array based beam expanders, which are also polarization maintaining [40]. There are a number of options for back projection polarization maintaining screens as well.



(a)

(b)



(c)

Figure 2.8: (a) A photograph of surfaces, region of interest is shown (A: Silver Screen, B: Duct tape, C: New generation micro lens array screen from Microvision, D: Old generation micro lens array screen from Microvision, E: Surface of a tablet computer (Apple iPad). (b) Sample regions from a photograph of the illuminated surfaces. (c) Sample regions from a photograph of the illuminated surfaces with a cross polarizer in front of the camera.

Table 2.8: Measured brightness (cd/m^2) in the region of interest shown in Figure 2.8.

	A	B	C	D	E
Figure 2.8b	4.936	14.166	9.546	12.532	11.891
Figure 2.8c	0.545	1.961	0.590	0.673	0.598
Ratios	10 : 1	7 : 1	20 : 1	20 : 1	20 : 1

2.1.6 Conclusion

We have successfully demonstrated a novel glasses-type stereoscopic pico projector based projection display with an active polarization rotator. The prototype operates on source 3D content with a display refresh rate value of 60 Hz without any noticeable flicker effect. It was also demonstrated that there are several usable surfaces such as microlens array screens, silver screens and even duct tape or the surface of a tablet computer. In other words, with a little creativity, one can find suitable surfaces for polarization-based 3D display in everyday environments. As conventional stereoscopic contents cannot be used directly in this approach, a script was written for on the fly conversion and off-line conversion of the video. The average cross talk level is measured as 4.3% for left eye and 7.5% for right eye. Viewers generally agreed that the current level of crosstalk did not degrade the 3D experience. The techniques and hardware described in this work can easily be incorporated into the pico projector engine. This new approach opens a path for displaying and sharing 3D content with a mobile device.

2.2 Super Stereocopy 3D Technique

This section introduces a new twist in stereoscopy domain. The idea was first raised in the weekly progress meetings in between Kaan Akşit and Hakan Urey. It was first thought as a novel new type of contact lens. But through experiments, it was found out that the same effect can be achieved even with a spacing in between the proposed structure and the eye lens. The idea was first tested by Kaan Akşit and Amir Hossein Ghanbari Niaki, an M.Sc. student at OML. The text of this section was made possible through consensus in between Kaan Akşit, Amir Hossein Ghanbari Niaki, and Hakan Urey. This part of our research is sponsored by The Scientific and Technological Research Council of Turkey (TÜBİTAK), Project No: 111E183.

2.2.1 Introduction

3D displays generally provide only a stereo image pair to two eyes. As a consequence, the image appears focused only at the display screen and the eyes fixate on the display screen rather than the correct depth of the object. Fixating at the wrong depth forces the eyes to an unnatural viewing condition by decoupling the accommodation[1] and convergence[2]. The conflict between the decoupled accommodation and convergence has been implicated as a cause of perceptual error (failure of size constancy), viewing discomfort, and visual system changes in all stereoscopic displays (including 3D TVs, 3D cinema).

This conflict becomes a problem in practice, when the virtual objects are shown at distances closer than 0.5 m [41]. The second major problem of the conventional autostereoscopic displays is the lack of monocular parallax, which will be introduced

[1]Change of focal length of the eye to focus at different depths provides information about the depth. The focal length changes by voluntary or involuntary adjustment of the curvature of the crystalline lens of the eye to keep an image focused on the fovea.

[2]When two eyeballs focus on the same object, they converge by inward rotation towards the object. The angle of convergence is larger when the eye is fixating on nearby objects. Ocular convergence or simply the convergence is one of two binocular cues of visual depth perception, in which kinaesthetic information about the degree of ocular convergence indicates the distance of objects being fixated. Both the accommodation and convergence depth cues require the use of eye muscles and provide neural feedback about the depth of the object.

in Section 2.2.2. Monocular parallax is one of the monocular cues of visual depth perception. It depends on the fact that movement of the head produces relatively large apparent displacement of nearby objects in the opposite direction and relatively small displacement of distant objects in the same direction. Also called motion parallax or movement parallax. Autostereoscopic displays with suggested solution to both problems are classified as Super Multi View (SMV) displays implemented for autostereoscopic displays [42–44]. SMV displays provides a large number of images with narrow interval of parallax to overcome the mentioned problems [42, 45, 46]. Such architectures require large number of views and pixels, and architectures with great level of optical complexity and, therefore, had limited success. The mentioned limitations have not been addressed properly in the case of passive binoculars type stereoscopic displays, which are more common in practice, and visibly cost-effective.

This work introduces a new major milestone for binoculars type stereoscopic displays, where the accommodation-vergence conflict is improved to a large degree by introducing monocular parallax by way of simultaneously presenting multiple perspective images to a single eye. Our contribution is highlighted as in Figure 2.9.

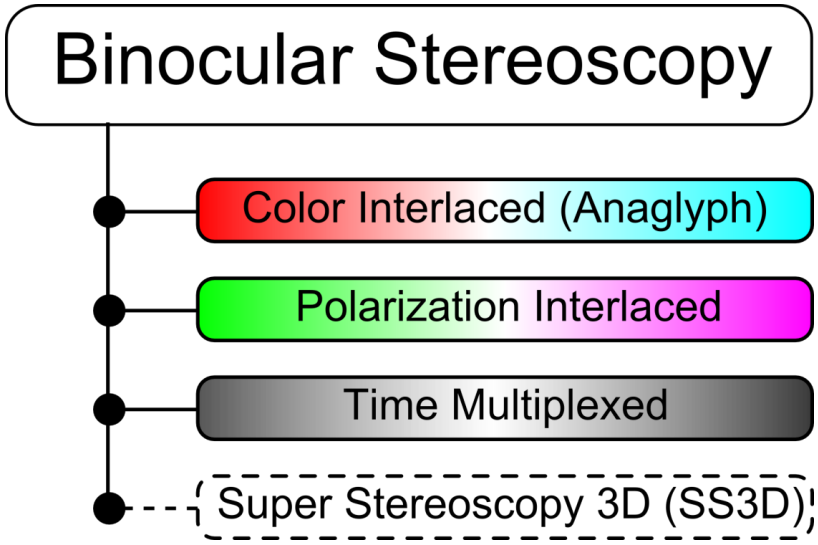


Figure 2.9: Sketch showing our contribution to the existing literature.

Our demonstrator satisfies SMV region criterion [42] by providing monocular

parallax at least in the horizontal axis. This new method is named as Super Stereoscopy 3D (SS3D) technique. Through subjective tests, we have shown that SS3D glasses can enhance 3D perception without visual fatigue and also extend the perceived depth of field. Due to the use of apertures in front of the eye, main limitations of our method are reduced brightness and limited field-of-view.

2.2.2 Method

SMV displays produce multiple viewing slits per eye by generating a narrower angular spread than a single eye pupil [42, 46]. SMV criterion is met when at least two perspective images enters from different regions of the pupil. If different views are rendered with about 2 mm wide vertical viewing slits across the viewer's face, about 50 views are required across the viewer's face. The same principle is valid for the case of stereoscopic displays, where a minimum of four images is required (two per eye).

Figure 2.10 illustrates the operation of conventional stereoscopic binoculars with the new SS3D binoculars. While the polarizers are used to separate the left and right eye images in both cases, two pinholes with color filters are used to separate the perspective images in the SS3D case. Note that there is a remarkable reduction in the volumetric pixel (voxel) size when Figure 2.10(c) and Figure 2.10(d) are compared. In other implementations, the perspective images can be selected using either spatial selectivity (color filters, polarization filters) or temporal selectivity (electronic shutters). A combination of filters and shutters can be used to increase the number of perspective views per eye. While circular pinholes are typically used, other shapes such as slits have been shown to be effective as well.

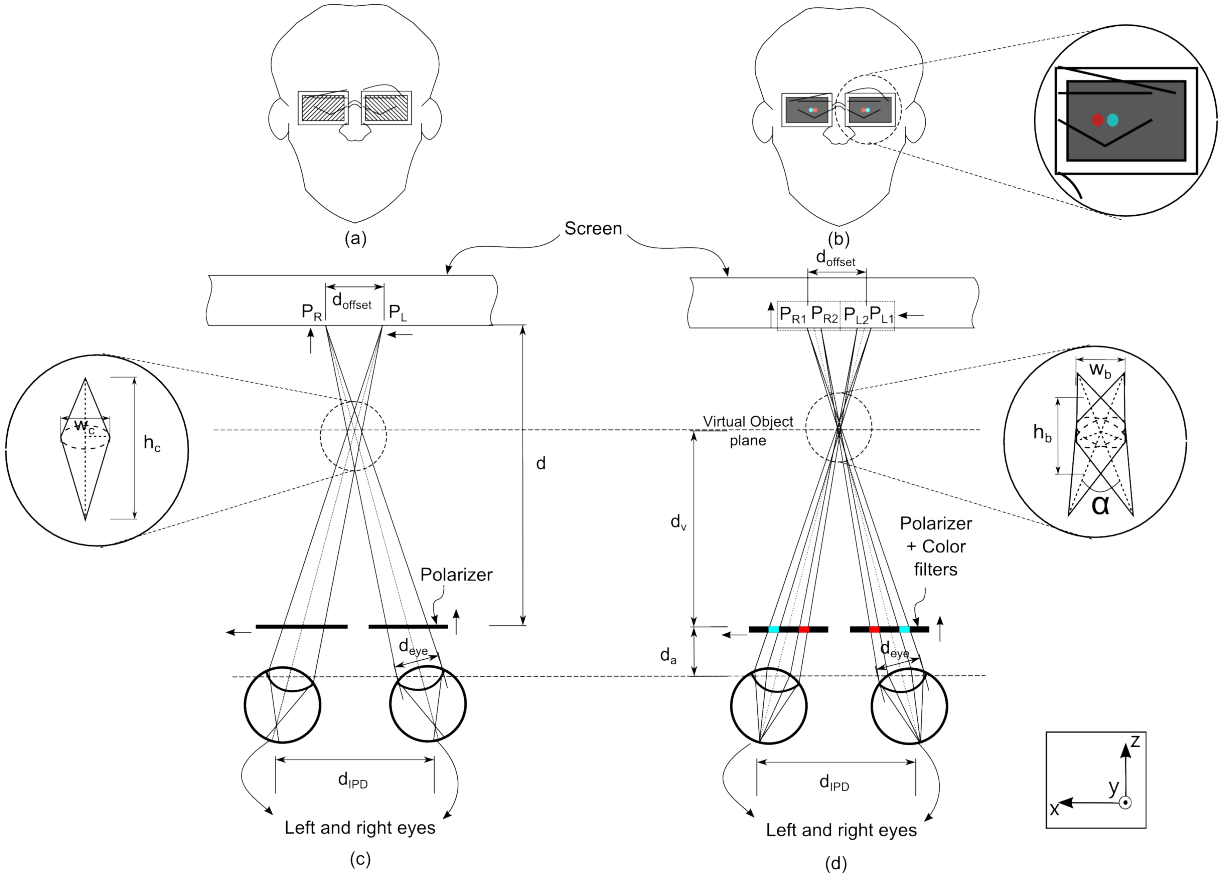


Figure 2.10: Sketch showing a user with (a) a conventional polarization based binoculars and (b) binoculars of our proposal. Note that in both conditions, the eyes are focused at the virtual object plane. Shape of a voxel in (c) a conventional polarization based stereoscopic display, and (d) our proposed system.

The limitations and the optimization of the pinholes based SS3D method in terms of voxel size, angular resolution, and field-of-view as a function binocular design and other display parameters are discussed in detail in Section 2.2.4.1.

2.2.3 Prototype

Multiple platforms, either a stereoscopic or an autostereoscopic display, may benefit from our proposed method. We had chosen to build a prototype to be used with our in-house built autostereoscopic display [K8] to highlight that our method is applicable to different platforms. An optimized pinhole design (details in subsequent sections)

with pinhole size $d_p = 0.6\ mm$, and pinhole spacing $d_k = 2\ mm$ is built as shown in Figure 2.11(a). The prototype is able to provide the best possible angular resolution with a Field-Of-View (FOV) of $\sim 30°$. Each pinhole is equipped with a color filter taken from conventional anaglyph glasses. Note that there are only two different color filters (red and cyan) in our prototype.

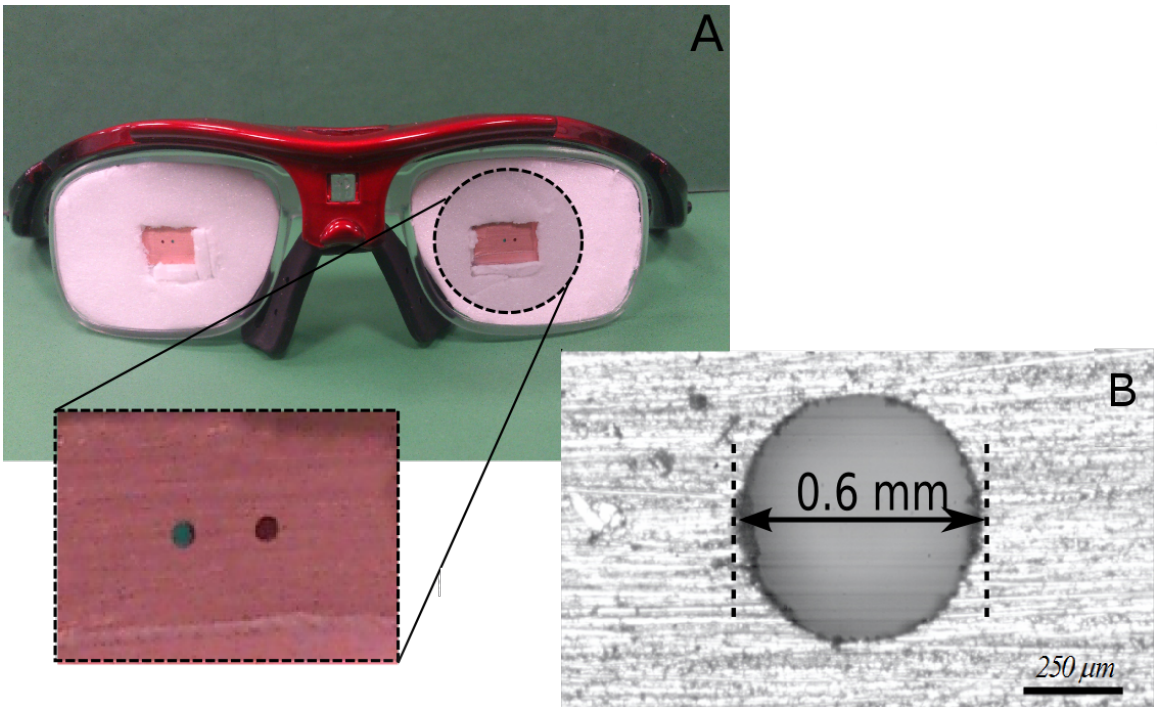


Figure 2.11: Photographs showing (a) double pinhole glasses with $d_p = 0.6\ mm$, and $d_k = 2\ mm$, and (b) a single pinhole under a microscope.

Figure 2.11(b) provides a photograph of a pinhole from our prototype under an optical microscope. Figure 2.12 shows the results of an experiment to show the monocular voxel formation using an image that should appear as a white vertical bar at the correct virtual image plane. A conventional LCD display is placed at 56 cm distance from a camera. The camera is focused at three different planes: virtual object plane at $\sim 25\ cm$, screen at $\sim 56\ cm$, and infinity. The content in the experiments contains two white bars with different disparities corresponding to virtual objects at $\sim 25\ cm$ (on the right) and ($\sim 20\ cm$) (on the left). Figure 2.12 provides different

sets of photographs captured with different focuses, when conventional method, a single pinhole, double pinhole without filters and double pinhole with filters are used. Single pinhole case and double pinhole without filters case are in Figure 2.12 to show the intermediate steps. Note that, without the pinholes, the width of the virtual object is substantially larger and blurred compared to the object width at the screen. The cases with single pinhole and two-pinholes without color filters are not expected to work well but included here in to illustrate the alternatives cases. For the two pinholes with filters case, the bar on the right appears focused and white while the bar on the left appears slightly colored due to different virtual image location, thus providing the monocular parallax formation with correct accommodation. Only a single pinhole can also produce a sharp image as it provides nearly infinite depth of focus. However, as our eyes are accustomed to a natural blurring effect with defocus, the resultant image is not natural looking. Using the SS3D glasses, image is blurred to non-overlapping color images when the accommodation is shifted away from the virtual object plane. Double image leads the eyes to correct virtual image plane. If the number of pinholes (and views) is increased, even more natural blurring effect can be achieved. At the extreme, this leads to continuous parallax as in the case of holographic vision. The number of views can be increased by using color, polarization, and temporal multiplexing simultaneously. The photographs in Figure 2.12 were captured using the same exposure. The dramatic decrease in luminous intensity with double pinholes and filters is visible through the photographs. When observed with the conventional method, brightness is adequate since the eye response is adapted, and the image quality is equally good in all the experiments. Increasing the number of pinholes would improve the brightness and field-of-view further but more complex filters and shutters would be needed on the eyeglass.

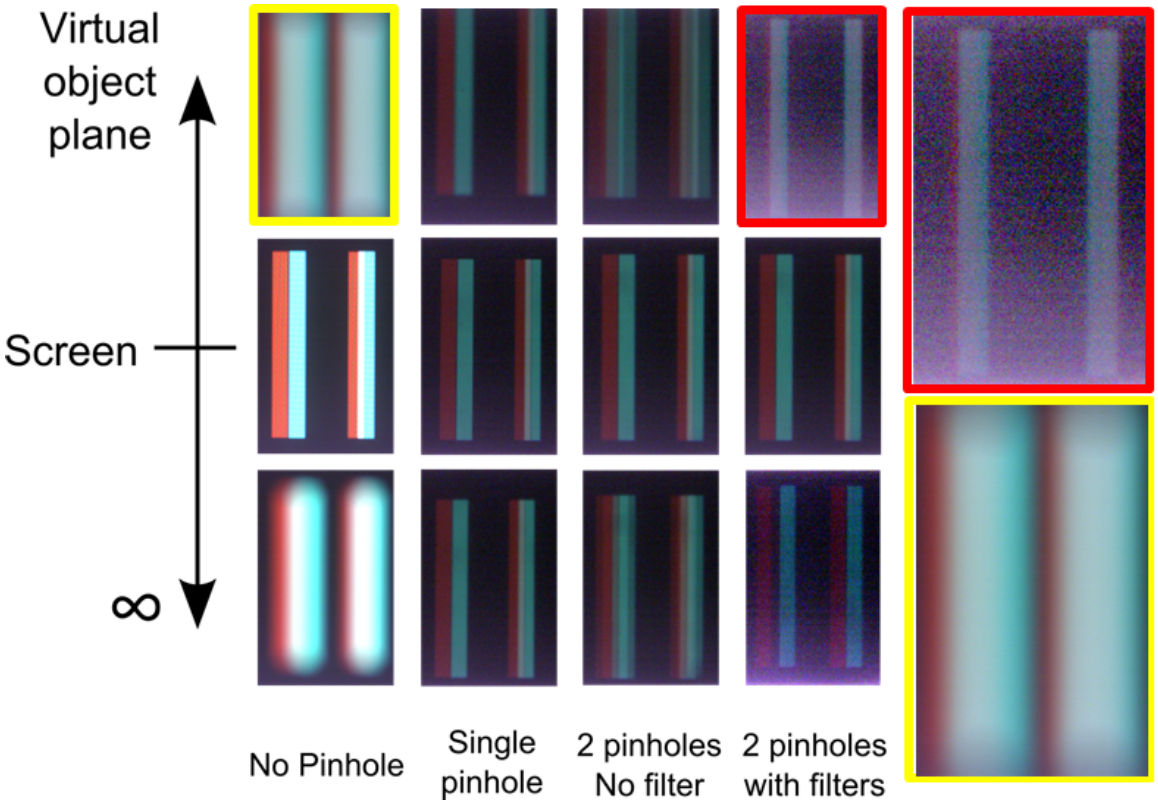


Figure 2.12: Photographs from a camera focused at the virtual object plane ($\sim 25 - 30\ cm$) for cases with pinhole, double pinhole, and double pinhole with color filters. Photographs are captured with the same exposure time.

2.2.4 Results and discussion

We have conducted a test to see the capabilities of the prototype with the real users using our in-house built autostereoscopic display among with SS3D binoculars. An important part of our trials is the requirement of correct content creation. Figure 2.13 illustrates the concept of the content creation applied for our tests, in which L represents contents for the left eye and R for the right eye. As shown in Figure 2.13, L1 and R1 are coded as red and cyan respectively. Color coded images are superimposed on top of each other to create two full color images as left and right. Overall two images were created at close distances ($d_v = 500\ mm - 550\ mm$), which can not be perceived as 3D image for an average viewer using conventional methods. Both of the created contents can be seen in Figure 2.14.

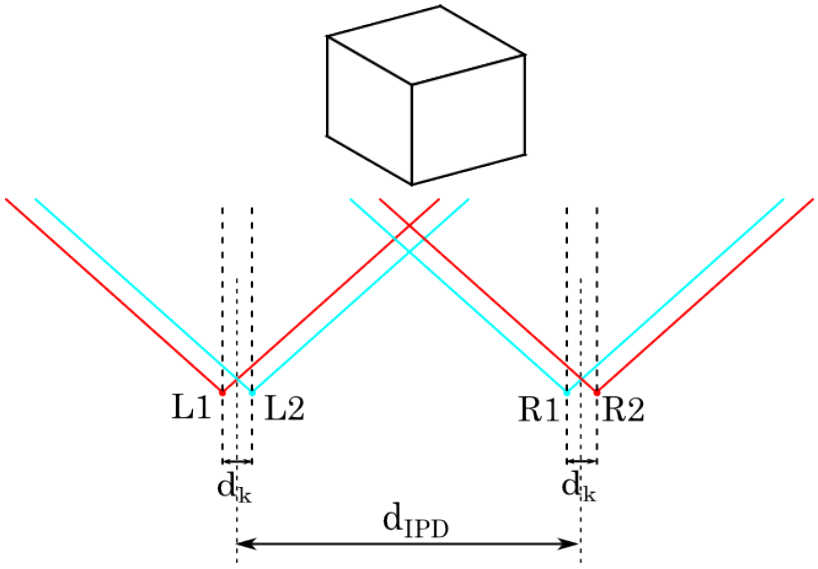


Figure 2.13: Sketch showing the used content capture geometry for our tests.

An autostereoscopic display prototype developed in our lab is used in the experiments with human subjects. 15 subjects were first tested using some 3D test images that appeared around $1m$. They were all able to see 3D without any eyeglasses in the demonstrator. Later, they were shown contents A and B and asked if they were able to perceive 3D (i) without glasses, and (ii) with double pinhole SS3D glasses. Their ability to see those images in 3D are summarized in Figure 2.14. Every subject indicated that their comfort and ability to perceive 3D increased dramatically with both types of SS3D glasses. While only 13 % can see 3D without glasses, 100 % of the subjects were able to see 3D with glasses. This is mainly because of the improved monocular parallax and the infinite depth of focus offered by the pinholes. Each tester commented that their vergence and accommodation have changed in a correct manner with the help of the SS3D glasses. On the other hand, subjects reported a blurred double vision when they wanted to see the content in 3D with bare eyes. During the trials none of the testers experienced an eye strain while using the glasses.

The testers on the other hand highlighted also some visible disadvantages. Testers found a dramatic decrease in perceived light intensity and perceived some speckle due to the pinholes and the laser illumination.

We believe that our proposal can be an easy and cost-effective modification on existing 3D TV and cinema. It is found to be very useful tool in the cases of objects closer than 50 *cm*. The monocular parallax can be improved further by using multiple pinholes with multiple light selective filters in both horizontal and vertical axis. Thus, an enhanced parallax can be presented using the same display hardware with modified SS3D glasses and modified content. We believe that images at different planes will look sharp, allowing accommodation-transient-free seeing and decreasing the eye's adaptation over time.

Even though the testing methods and the number of subjects were not sufficient to have conclusive results, the results of the preliminary experiments were encouraging and showed that SS3D glasses significantly improved the 3D vision and reduced the accommodation-vergence conflict.

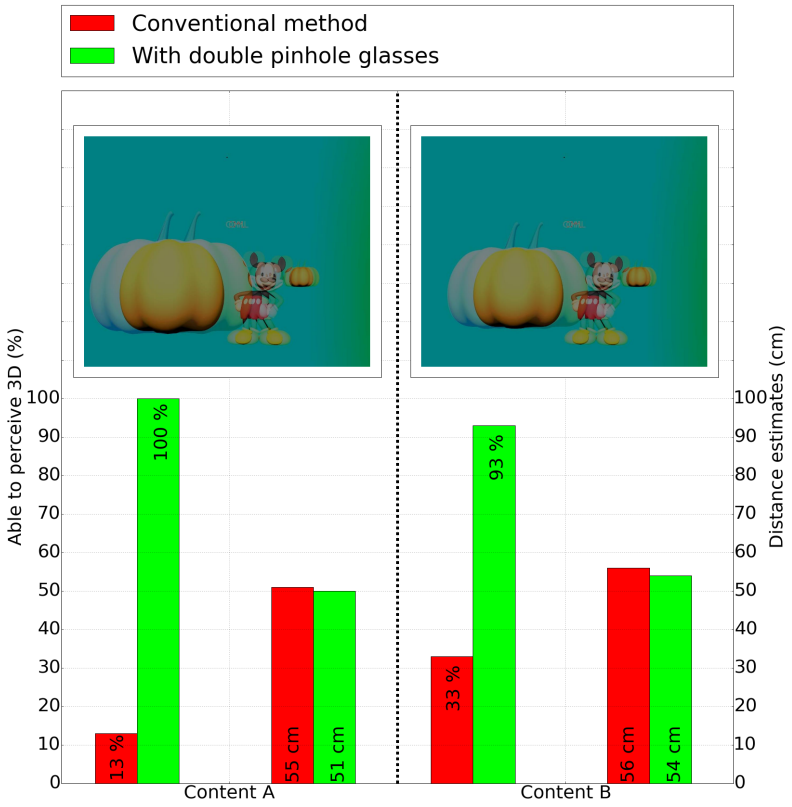


Figure 2.14: Sketch showing results of the subjective tests. At each graph the ability to perceive 3D is shown on the left hand side. On the right hand side, subjects' depth estimation is presented. Additionally, the used content for each test is shown.

2.2.4.1 Limitations of SS3D Technology

2.2.4.2 Volumetric pixel size

The voxel shapes and sizes in our system are different than those for a conventional stereoscopic system. Using ray-optics, a conventional stereoscopic display's voxel size, which is shown in Figure 2.10(c), can be found in terms of width w_c and height h_c as in:

$$w_c = \frac{(d - d_v) \times d_{eye}}{d}, \tag{2.2}$$

$$h_c = w_c \times \left(\frac{d - d_v}{d_{offset} - w_c} + \frac{d_v}{d_{IPD} + d_{eye} - w_c} \right), \tag{2.3}$$

respectively. Additionally, the required offset value in between image sources d_{offset} can be found using Equation:

$$d_{offset} = d_{IPD} \times \frac{d - d_v}{d_v}. \tag{2.4}$$

As an example, eye pupil diameter $d_{eye} = 5\ mm$, interpupillary distance $d_{IPD} = 65\ mm$, distance between the viewer and the screen $d = 1000\ mm$, distance between the viewer and the virtual object $d_v = 500\ mm$, offset in between image sources $d_{offset} = 65\ mm$ will lead to voxel width $w_c = 2.5\ mm$, and voxel height $h_c = 38\ mm$. Note that d_{eye} is chosen as $5\ mm$ to give a reasonable comparison between SS3D case and the conventional stereoscopy case. Note that the geometrical shape of the voxel created in this case is similar to a shape of a diamond with a sharp tip.

The overall voxel size of our proposal can be calculated in two phases: First the monocular voxels shown in Figure 2.15 has to be calculated in size, and than binocular voxel, which is the overall voxel, shown in Figure 2.10(d) can be determined approximately using monocular voxel dimensions.

The formation of monocular parallax using the pinhole binoculars are represented in detail in Figure 2.15, in which d represents the distance between the screen and the viewer, d_a is the distance between the binoculars and the eye, d_p is the diameter of the aperture, d_k is the separation between two apertures from center to center, w_m is the monocular voxel width, h_m is the monocular voxel height, and d_c is the separation between two image sources reserved for each eye (P_1 and P_2), which creates the voxel P. Correlation between the introduced variables can be expressed by means of ray-optics, voxel width and height are found as:

$$w_m = \frac{(d - d_v) \times d_p}{d}, \tag{2.5}$$

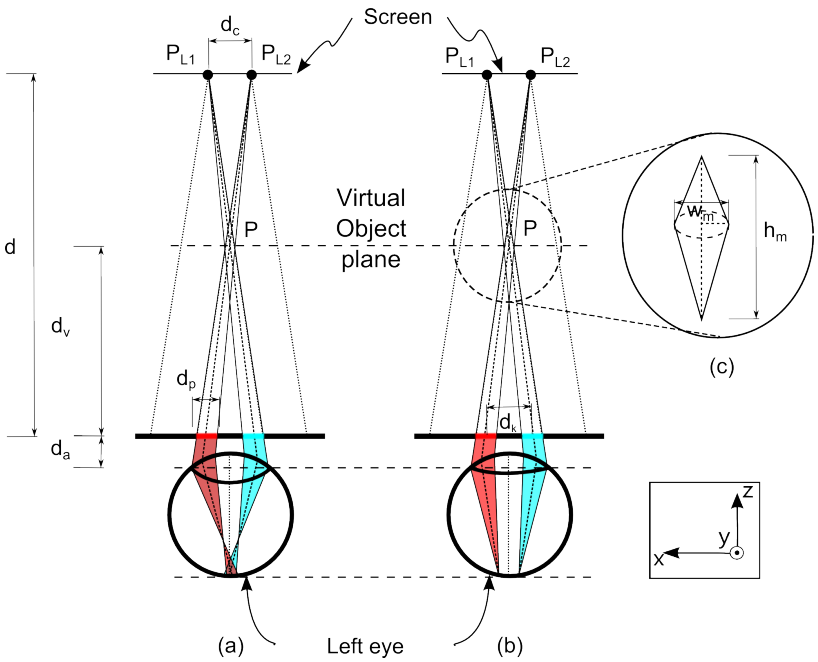


Figure 2.15: Sketch showing the left eye of a viewer focused at (a) the virtual object plane, (b) the screen. Part (c) demonstrates how a monocular voxel looks like when pin-hole apertures with color filters are used.

$$h_m = w_m \left(\frac{d_v}{d_p + d_k - w_m} + \frac{d - d_v}{d_c - w_m} \right), \tag{2.6}$$

respectively. Additionally, Equation:

$$d_v = \frac{d_k \times d}{d_c + d_k} \tag{2.7}$$

provides a binding relation in between distances. Effect of the change in different variables is also observed over the monocular voxel dimensions (h_m, w_m) through our simulations, found under Section 5.2. The simulations are carried by setting $d = 1000\ mm$, and $d_k = 2\ mm$. The results can be seen in Figure 2.16. Note that Figure 2.16 shows voxel width using ray-optics (continuous lines) and diffraction theory (dashed lines). Diffraction theory predicts a voxel width as:

$$w_m = tan(arcsin(2.44 \times \lambda / d_p)) \times d_v. \tag{2.8}$$

The optimum pinhole size d_p can be determined using the intersection points of diffraction calculations and ray-optics calculations for each virtual object plane ($d_v = 100 - 300 - 500 - 700\ mm$). In the case of our prototype, we have built a double pinhole structure with $d_p = 0.6\ mm$ to show voxels in between $d_v = 300 - 500\ mm$.

Monocular voxel width w_m is slightly larger in size than binocular voxel width w_b but for practical purposes they can be considered equal. On the other hand, with bigger values of α, h_b always tends to be larger than w_m, assuming it to be more or less equal to w_m is an acceptable approximation. As an example, $d_{eye} = 5\ mm$, $d_{IPD} = 65\ mm$, $d = 1000\ mm$, $d_v = 500\ mm$, $d_{offset} = 65\ mm$, $d_k = 2\ mm$, $d_c = 2\ mm$, $d_p = 0.6\ mm$ will lead to $w_b = 0.3\ mm$, and $h_b = 0.3\ mm$. Whereas in the case of the conventional stereoscopic displays, these values are $w_c = 2.5\ mm$, and $h_c = 38\ mm$. The geometric shape of the voxel created in our proposal is similar to a shape of a distorted sphere. The shape tends to become a ball shape with increasing offsets or increasing number of views. Thus, more realistic point source like voxels can be created using our system. Figure 2.17 shows different computer generated voxel shapes with different intersection angles α.

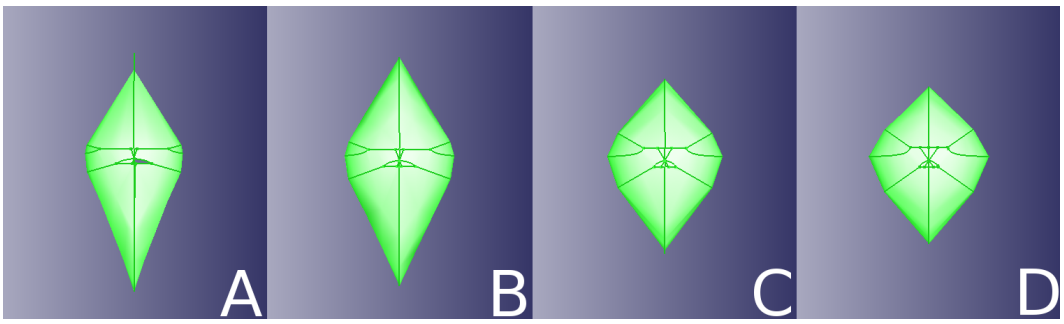


Figure 2.17: Computer generated representation of sample binocular voxel shapes when angle between monocular voxels (α) in Figure 2.10 is equal to (a) 30^o, (b) 40^o, (c) 60^o, (d) 70^o.

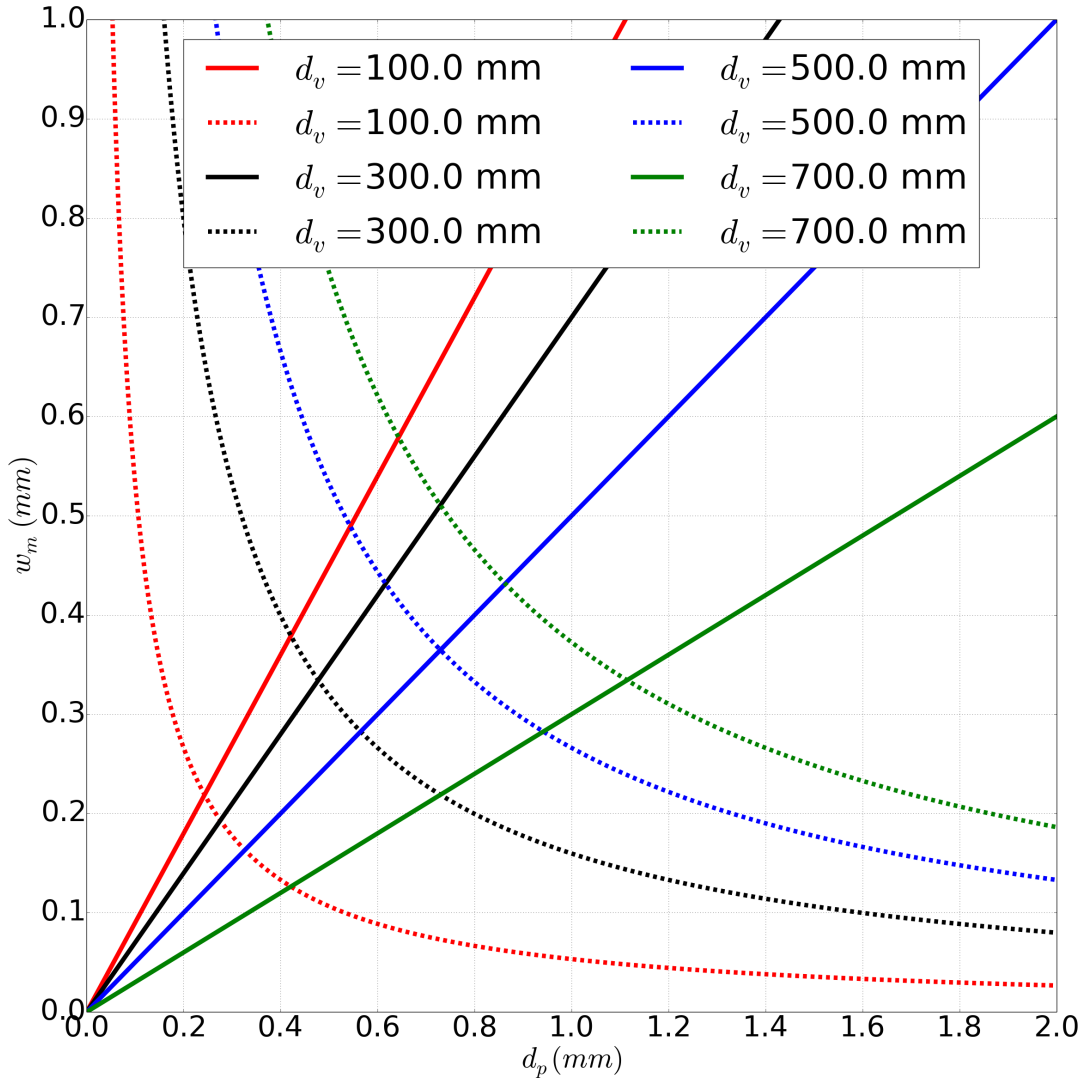


Figure 2.16: Simulation outputs showing Monocular Voxel Widths (w_m) with different pinhole diameter (d_p). The simulations are carried out using $d_a = 5\ mm$, $d = 1000\ mm$, and $d_k = 2\ mm$. Voxel widths are shown using ray-optics (continuous lines) and diffraction theory (dashed lines) for different virtual object planes ($d_v = 100 - 300 - 500 - 700\ mm$).

2.2.4.3 Field of View

A single eye of a human has a horizontal FOV of $95° - 115°$. To use special apertures in front of the eyes clearly limits FOV when Figure 2.18(a) is examined. The viewers are using only small portion of their FOV in real life scenarios. As an example, assume a viewer is standing in front of a display at $d = 1000\ mm$ and the display is a typical display found in an office environment with 500 mm diagonal size. Thus, the viewer in our example uses a FOV of $\theta_{FOV} \approx 30^o$ at most. FOV of a single aperture in front of the eye can be calculated using Equation:

$$\theta_{FOV} = 2 \times arctan(\frac{d_p + d_{eye}}{2 \times d_a}) \tag{2.9}$$

The case of using multiple apertures in front of a single eye is shown in Figure 2.18(b). Using multiple apertures, and the same formula, one can keep the same FOV with smaller aperture widths with a separation in between. But the region which could be used for SS3D can be found as:

$$\theta_{FOV_{SS3D}} = 2 \times arctan(\frac{d_{eye} - d_p - d_k}{2 \times d_a}), \tag{2.10}$$

$\theta_{FOV} \approx 30^o$ can be reached when $d_k = 2\ mm$, $d_{eye} = 5\ mm$, $d_a = 5\ mm$ and $d_p = 0.6\ mm$ are selected. In the case of multiple apertures, amount of the rays arriving at a certain point at retina decreases, so the viewer perceives an vignetting image with less intensity. This was not found to be a concern by the viewers wearing our prototypes. In case the proposed system is converted into a contact lens, d_a will have the smallest value possible, thus d_k will decrease as well.

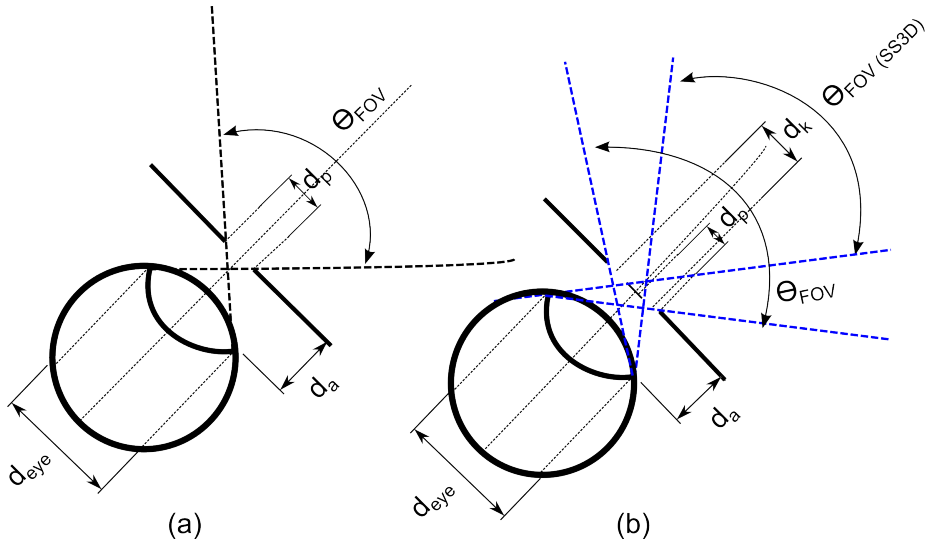


Figure 2.18: Sketch showing FOV (a) with a comparison in between a bare eye and an eye with a single pinhole, (b) of multiple apertures.

2.2.4.4 Angular Resolution

Another important variable of our proposal is the smallest angular resolution θ_{spot}, which should be almost equal to a single pixel appearing on the screen. Otherwise, a viewer will face problems resolving features on the screen. Due to the light selectivity nature of our proposal, optics theory predicts a relationship in between d_p, d_v and θ_{spot} as in Rayleigh angular resolution relationship: $\theta_{spot} = arcsin(2.44 \times \lambda/d_p)$. Smallest feature that has to be resolved should be around $\theta_{spot} = w_b/d_v$, when paraxial approximation is made. As an exemplary, smallest feature at $d_v = 500\ mm$, $\lambda = 532\ nm$ will lead to $d_p = 0.6\ mm$ to $1\ mm$ and $w_b = 0.5\ mm$ to $0.3\ mm$.

2.2.5 Conclusion

A new method entitled as Super Stereoscopy Technique is proposed to enhance the capabilities of stereoscopic displays and autostereoscopic displays. The method provides solution via monocular parallax formation to a common problem in stereoscopy field: accommodation-vergence conflict. The method was demonstrated using an in-house built 3D displays as the test bed. Our implementation provides two different

perspectives for each eye of every user by employing pinhole covered with color filters used in anaglyph stereoscopy method. Through human subjective experiments, the method was found to enhance the depth perception, and avoids accommodation-vergence conflict. In the most extreme case (virtual objects at $< 50\ cm$), subjects who could perceive 3D increased from 13 % to 100 % with the help of our double pinhole glasses. Major drawback of the method is a decrease in perceived light intensity, which can be fairly solved using multiple pinholes with multiple color filters. The overall system provides a promising modification to existing technologies.

2.3 Augmented Reality Display Application using head mounted Pico Projector

This section focuses on an augmented reality display application, which uses head mounted pico projectors. The idea was first raised in the weekly progress meetings in between Kaan Akşit and Hakan Urey. The idea was first tested by Kaan Akşit, Daniel Kade, Mehmet Kadıoğlu, and Ekin Akyürek . The text of this section was made possible through consensus in between Kaan Akşit, Daniel Kade, and Hakan Urey. This part of our research was made possible through the support of the FP7-PEOPLE-2012-IAPP research grant within the project "NaMoCAP - A mobility action in Interaction Design Research Focused on Natural Motion Capturing Process for Creative Industries" (Grant Agreement No: 324333).

2.3.1 Introduction

Entertainment industry products such as video games and films are deeply depending on Computer Generated Imagery (CGI). There are many cases where CGI characters' movements need to meet real world physics. One of the widely used ways is to capture motions of human actors in a dedicated vision based motion capture [47] studio as shown in Figure 2.19. Alternative motion capture techniques beside vision based techniques have also been investigated for daily-activities and professional motion capture [48, 49]. A typical set up of a motion capture studio is shown in Figure 2.19; a large space shoot area, motion capture cameras aiming at the shoot area, conventional cameras and sometimes projection displays, film cameras and pre-visualization (previs) cameras are typical components of such a studio.

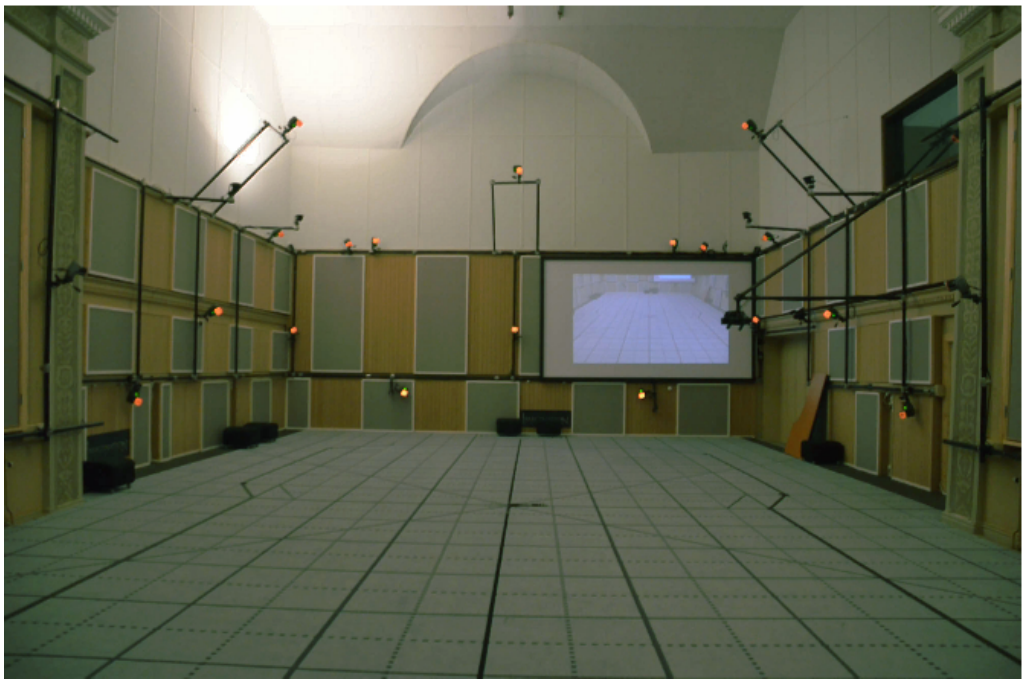


Figure 2.19: Motion capture studio of Imagination Studios located in Uppsala, Sweden. The studio is equipped with near infrared light source equipped motion capture cameras, and a projection display.

Actors are usually using a quite minimal scenery to resemble the environment they act in. Usually simple metal or wooden props are used to create the acting environment. Previous research has shown that actors can experience challenges to perform in such an environment and that the quite diversely skilled 'actors' would benefit from improvements in a motion capture shoot procedure [50, 51]. This is why this research addresses this issue and aims at providing a new interface for motion capture actors that is closer to an actual movie set and especially allows to give visual aid that can be used to act off.

To provide an application for a motion capture environment, such an application needs to comply with the specific needs of the environment. It is important for some motion capture shoots that the face and vision of an actor is not occluded by virtual reality glasses or other equipment. This is especially the case for facial motion capture shoots, stunts and when interactions with real world objects or other persons would

interfere with the worn hardware.

For some motion capture shoots, as depicted in Figure 2.19, a projection display showing the virtual environment from a single perspective is mounted and usable for motion capture shoots. During a motion capture session, in most cases, it is not possible for actors to see the computer generated virtual world around them, without performing unnatural movements like turning the head to look at the screen. This is especially an issue if this is not wanted. Therefore, it would be better for an actor to see the virtual environment while acting in a more natural and usable way through their own first person view, in real-time.

In this work, we describe our prototype combining a retro-reflective screen that covers the walls and a headband consisting of a laser scanning projector with a smartphone and an additional battery pack for the projector. The smartphone's built-in sensors allow real actors to explore the digital world freely. Our prototype is a stand-a-lone, lightweight, mobile system that fits the requirements of a motion capture environment. Additionally, we have presented data on our initial informal functionality tests, which we performed to get feedback on the potential and the usability of the developed prototype.

2.3.2 State-of-the-Art

The interest in head-worn display systems [52–54] is rising with the increasing number of products on the market, where they are being used excessively in entertainment industry. Most of the see-through products offer a near-eye solution with a limited field of view, a constant focus, single eye usage (no stereoscopy), and limited depth of the field. Most of the near-eye solutions come with a great deal of optical complexity in design, e.g., Google Glass [55], or [56], in which an additional specially made contact lens has to be used to see the content. Thus, the users have the problem of interacting naturally with the physical world due to these optical limitations. By simply disconnecting the user from the real world, the mentioned problem is avoided via opaque wearable stereoscopic head-worn displays on the market, e.g., Oculus

Rift [57] as virtual reality glasses. Nonetheless, the challenge remains the same for real-life use cases. Our focus is to address these challenges by enhancing the optical features such as larger field of view, larger depth of field or focusing at any depth that are vital in a real life scenario.

Although there are great advancements within the past years, head-worn display prototypes from different institutions are not yet in an affordable price range and real life applications do not meet the promised usability on everyday basis, even though there is a research result that proposed a projection display for daily use [58]. Usability is especially of importance as we intend to provide real-time video content shown from a game engine to motion capture actors while acting.

A laser scanning pico-projection display was an exciting development, because it does not require any optical components to focus on any surface, and the number of pixel size stays constant with the increasing distance between the projector and the screen. Additionally, it comes with a coin size light engine [32], where it is very obvious that it reserves room for miniaturization in head-worn display systems.

HMPD's and their image qualities as well as the use and evaluation of reflective materials, used as a screen, have been already discussed and introduced by others [59]. This technology was also already used in other research where digital content is projected to reflective surfaces [60] as well as non-reflective surfaces [61, 62] to allow interactions between users and the system. Even though this research uses a head-mounted projector setup that projects 2D video content onto a reflective surface and allows for interaction with the system, it differs to the approach we describe in this work.

According to our point of view, a laser scanning pico-projection display has some interesting aspects to be explored as it does not require any optical components to focus on any surface and the number of pixels stay constant with the increasing distance between the projector and the screen. Additionally, it comes with a coin size light engine [32], which allows build even smaller head-worn display systems. One of the valuable contributions of this work was embedding such a pico-projector into a

head-worn display, so that it can provide infinite depth of focus, enhanced color gamut through laser technology and serves as a light-weight mobile system without requiring fixed cabling to a stationary system. The system we provide is independent from any additional system, e.g. a tracking system or a serve processing the image, and carries all equipment on a head band.

Laser projectors, as we use it in our research have been used in other research as well [63]. There, a shoulder mounted projector was combined with a depth camera to project images to non-reflective surfaces and to allow gestural interactions. It is obvious that the use of their mentioned system is completely different, even though the laser projector is the same. For our purpose it was necessary to have a head-mouted system that allows to have the projector close to the eyes without blocking the users' vision. Furthermore, our system needs to record head movements and should not limit an actor in his movement capabilities. This also means a rigid setup that cannot move easily was needed in our setup, as in motion capture acting, stunts and athletic movements have to be considered in the requirements towards a HMPD. Furthermore, a more optimized projector setup was needed for our purposes. So we reduced the size of the projector and connected it to an external battery pack which allows to extend the uptime of the projector and allows to have a more minimal setup.

Unlike other systems like e.g. from CastAR [64] or other research [65] our system does not require multiple projectors. Thus, problems originated from using multiple projectors such as the keystone or image registration are not an issue in our system. Even though the CastAR glasses or even similar AR glasses are about the size of normal sport sunglasses, parts of the face such as the eyelids and eyebrows are still covered when acting for facial motion capture. In our system, only a minor part of the forehead is covered.

Another more practical issue and difference compared to our system lays within the fact that the CastAR system uses tracking markers with infrared LED's on their reflective display material. A reason to try to avoid this within motion capture is that the cameras would pick up the light and could get affected by it. Masking the

regions of those markers within the motion capture software so that these regions are simply ignored while acquiring motion capture data could be a solution for stationary reflective surfaces or objects but for dynamic setups it is rather unlikely to mask and recalibrate a motion capture system after every setup change; it is simply ineffective.

With our head-worn display, connected to a game engine which is running on a mobile phone, we approach to create a wearable augmented reality projection in a slightly different way than as literature has shown. Others have created an augmented reality application that provides additional information about certain real-world locations and provide navigational help using RFID technology. This information is then shown on wearable data glasses [66]. Such applications differ to our application scenario, especially in the sense that the purpose for our use case is to immerse users into the digital environment and allow them to explore it instead of providing information or navigational guidance. This includes that the digital environment should be superimposed onto the normal vision in a higher resolution showing digital content from a game engine.

Conventional off-the-shelf laser scanning pico-projectors can provide an illumination of $10 - 20\ lm$. On the other hand, conventional off-the-shelf projectors can provide up to $1000\ lm$. Thus, it is required to use the light in the most efficient way in a system equipped with such a pico-projector. Availability of different retro-reflective materials is another excitement for our purpose, since they provide high light gain when used as screen. In the past, retro-reflective surfaces were used as a part of head-worn displays as in the case of [60, 67, 68] as well. To combine a recently licensed stereoscopy method [19, 69] based on polarized glasses is also easily realizable method to provide 3D imagery.

2.3.3 Head-worn Projection Display

Head-worn display systems are investigated extensively among the optics design community, and they are believed to be the expected hardware upgrade for future virtual reality applications. In this section, we introduce a theoretical background on

our simplistic head-worn projection display architecture. The section also provides information on our virtual reality application implementation.

2.3.3.1 Hardware Description

Our head-worn projection display system, as shown in Figure 2.20 consists of several off-the-shelf hardware: 1) a stripped-down laser pico projector SHOWWX+ from Microvision, Inc., 2) a Sony Xperia S smart phone, 3) retro-reflective material from Reflexite (not in the figure), and 4) in-house 3D printed housing for the equipment.

Figure 2.20: A photograph of the hardware device, equipped with a smart phone, a battery and a pico projector. The housing for both items is 3D printed in-house.

2.3.3.1.1 Pico projector The white laser light from the pico projector is a combination of three different laser light sources (RGB: 643 *nm*, 530 *nm*, and 446 *nm* respectively). The laser spot scanned from the projector is a Gaussian beam [9]. The Gaussian laser spot beam waist and the best resolution appear at about 80 *cm* distance from the projector. However, using 1st order optical approximations, we can assert that the image is always in focus beyond that distance since the image size and the spot size both increase linearly with distance, i.e., a number of resolvable spots do not

change with distance. The laser light source does not require beam shaping optics as in conventional projectors. The content shown with the pico projector remains at focus even in case of large distance variations. In Figure 2.20, the pico projector acts as the light engine in our head-mounted display prototype, thus, a user wearing the prototype does not suffer from any key distortion effect, even when the surface used as a screen is totally distorted, as can be observed in Figure 2.21.

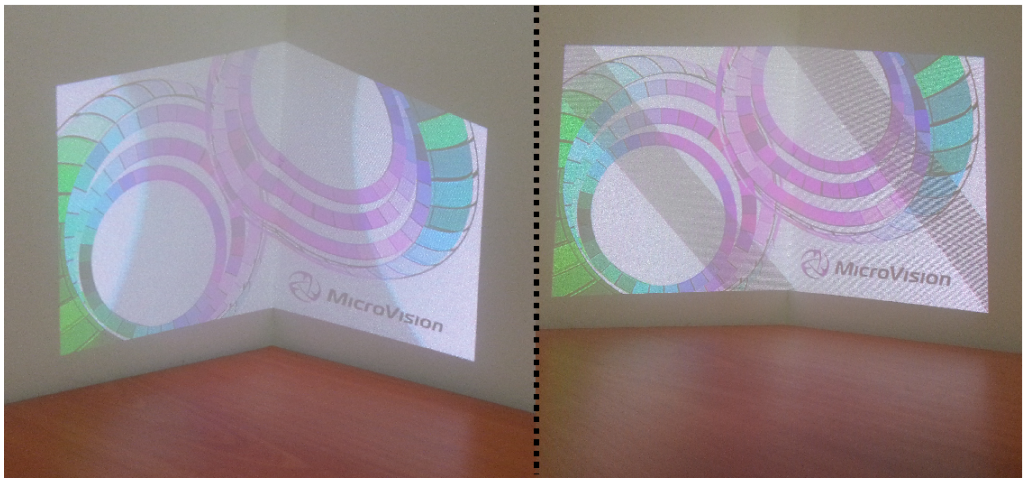


Figure 2.21: Left: A photograph of the visible key-distortion problem when the separation between the camera and the projector is high. Right: This photograph shows that there is no visible key-distortion effect when the camera is placed close to the pico projector, although the surface is curved.

2.3.3.1.2 Smartphone A conventional smartphone is equipped with different types of sensors (gyroscope, accelerometer, magnetometer, GPS sensor, ...) and can be hooked to the pico projector through an HDMI port or an MHL adapter. The maximum native resolution of the projector is $848px \times 480px$ @60 Hz *vsync* in size. Thus, a smartphone provides content at the very same resolution with the very same refresh rate. For our application, we address the sensors of the smartphone as means of controlling the digital environment.

2.3.3.1.3 Battery An off-the-shelf pico projector comes with a Lithium Ion battery with 3.7 V, and 1800 mAh, which corresponds to 6.7 Wh of energy. The mentioned

battery allows a pico projector to operate constantly around approximately $1,5$ hours. In our prototype, we replaced the parts of the projector's housing and the battery with two Lithium Polymer batteries ($2 \times 3,7\ V \times 2000\ mAh = 14,8\ Wh$). This modification reduced the size of the overall system, and increased the usage time of the pico projector. This modification also allows us to upgrade the battery by adding more battery cells to the prototype. It should be mentioned that the smartphone runs on its built-in battery. As an overall system, the expected uptime is $3 - 4$ hours.

2.3.3.1.4 Retro-reflective Screen The final part of our head-worn projection display is a high-gain screen, which has the property to retro-reflect the light to the source. Thus, a user standing close to the light source, in our case the projector, has high light gain; he or she basically sees the screen very bright. Through trials, it is observed that screen brightness is good enough for outdoor applications, e.g., sunny weather situations.

Retro-reflective material to be used as a screen can be in different forms: cloth/paint type [70], corner-cube (prismatic) type [71], and cat's eye type [72]. The one used for our prototype was a corner-cube type retro-reflective material. The corner-cube retro-reflective material can have the highest efficiency among the others, when they are placed in an hexagonal arrangement to have high-fill-factor. Typical corner-cube type retro-reflective material found on the market has a pitch size in between $0.1\ mm - 0.3\ mm$. The distance in between the pico projector and the screen is in the range of $1 - 10\ m$.

2.3.3.1.5 Use of Reflective Materials in Optical Motion Capture During our research we encounter an obvious issue: when using a projector-based solution that includes a retro-reflective material, an optical motion capture system will be affected by the light reflections of such a material. We tested different retro-reflective materials within a motion capture studio and found that all reflective foils were recognised by the motion capture system. In some situations the cameras were not functional for a short time anymore as the retro-reflective foil returned too much light to handle for

the camera.

As a test environment, we used a motion capture studio with 32 Eagle 4 cameras and the Cortex software from Motion Analysis. The foils that we tested were the Reflexite VC310 and the 3M 4090.

A4-sized samples of each foil were picked up by the motion capture cameras in an equal manner. Moving an A4-sized piece of the foils towards one camera led to a shutdown in about 2 m distance to the camera. A larger piece that was available from the Reflexite VC310 foil was then placed as intended on a wall of the shoot floor, behind the cameras. The sample sheet was 9 m x 0.775 m in size. Cameras on the other side of the shoot floor in about 8 m distance from the foil started to shut down immediately when running the motion capture software. Other cameras that were angled to pick up the reflections from the foil but located at the very end of the shoot floor were still affected by the returned light and sporadically shut down as well.

We found two solutions to the above explained problem:
1) Masking the foil in the motion capture software, so that it will not be considered.
2) Applying a notch filter to the retroreflective foil.

The first solution is only possible when the reflective materials are not large enough to affect the cameras and when the materials will not be moved. Otherwise re-masking would be necessary.

The second solution worked in our quick tests surprisingly well. To avoid this problem reflecting infra-red light that the motion capture cameras could pick up, a screen has to have a band-pass filter (a notch filter), which absorbed infra-red wavelength light, but allows visible light to pass. Initially, we tested a see-through plastic coating on top of the retro-reflective material. We observed that the see-through plastic coating seems to act as a notch filter for infra-red light. Nonetheless, this finding needs extended empirical testing and development.

2.3.3.1.6 Stereoscopy The hardware described so far can provide stereoscopic vision [16] through conventional stereoscopic methods using additional passive 3D glasses, such as the long known anaglyph method [73], or a common method in movie theaters: the polarized glasses method [74]. The active methods such as shutter glasses [75] causes noticable flicker with our prototype, due to the low vertical refresh rate of the pico projector. Previously, a new method [19] which solves the flicker problem and combines the benefits of a shutter glasses system with a polarized passive glasses type system has been invented. Although the method has an active component (a liquid crystal polarization rotator) mounted on the projector, it works without any noticeable flicker in pico projectors that have low refresh rates. The method requires the screen to be polarization maintaining. There are polarization maintaining retro-reflective foils on the market as well. To make our prototype stereoscopic using the mentioned technique, a polarization rotator has to be mounted on the photonics module of the pico projector. Additionally, the user has to wear polarized passive glasses or contact lenses. Currently, our system does not provide 3D imagery, but with the modifications mentioned it is possible to provide 3D imagery.

2.3.3.2 Software Description

As software development environment, the Unity 4 engine, which is widely used in industry and research, was chosen for our prototype. One requirement for the software was to be able to create and change digital environments that will be shown to actors in a quick way and by allowing to use file formats that are common in computer games creation. Another requirement to be considered was that the built-in sensors within our prototype such as gyroscope and accelerometer were meant to be used to control the digital environment. These decisions limited our choice to a few game engines that support mobile phone game development. As the Unity 4 game engine is a cross-platform and state-of-the-art game engine, our decision was to use it to develop our software. To develop the software and create the digital environment we used Unity 4.3.1f1 by compiling the software to an Android phone (tested phones:

Samsung Galaxy S4, Samsung Galaxy S4 mini, Sony Xperia S, Samsung Galaxy S3).

Furthermore, we implemented controls to make the environment explore able through using the gyroscope to react to the movements of the phone which can be mounted in different positions on the head of a user (back-head, top-head and side-head). The accelerometer was used to determine the steps and the direction of movement of a user. In places where there is limited space to walk, a solution based on walk-in-place [76] was also implemented as an option. Note that multiple users using multiple prototypes can run the same software independently but our software does not provide any synchronization between the users at this moment. Head rotations, independent from the phone's mounting position, and walk-in-place movements look accurate, so that exploring a digital environment is possible.

Moreover, we figured that a few features needed to be implemented so that testing the system, the software and the user experience was possible. One of those features was to be able to scale the character controller from the debug menu we added. This was especially useful to adjust the height of the character controller when different environments needed to be loaded or added contents were scaled differently. Scaling the view height allowed to adjust for on-the-fly changes while using the environment to make scenes and objects placed in the environment look more believable. Also, changing the orientation of the phone and the views of the screen was a helpful feature to comply with testing different mounting locations on the user's head. For testing purposes and also to allow switching scenarios and scenes quickly without reconfiguring or loading new digital environments we added the feature of switching to different locations or scenes within the environment so that e.g. for motion capture shoots, scenes can be switched in an instance and in real-time.

In Figure 2.22, the three locations that were implemented for testing and exploring our digital environment are depicted. To create the digital environment, a height map of an island was taken and modified to get the basic layout of the digital environment. Used content and textures to create the environment are freely accessible on the Internet or through the Unity Asset Store.

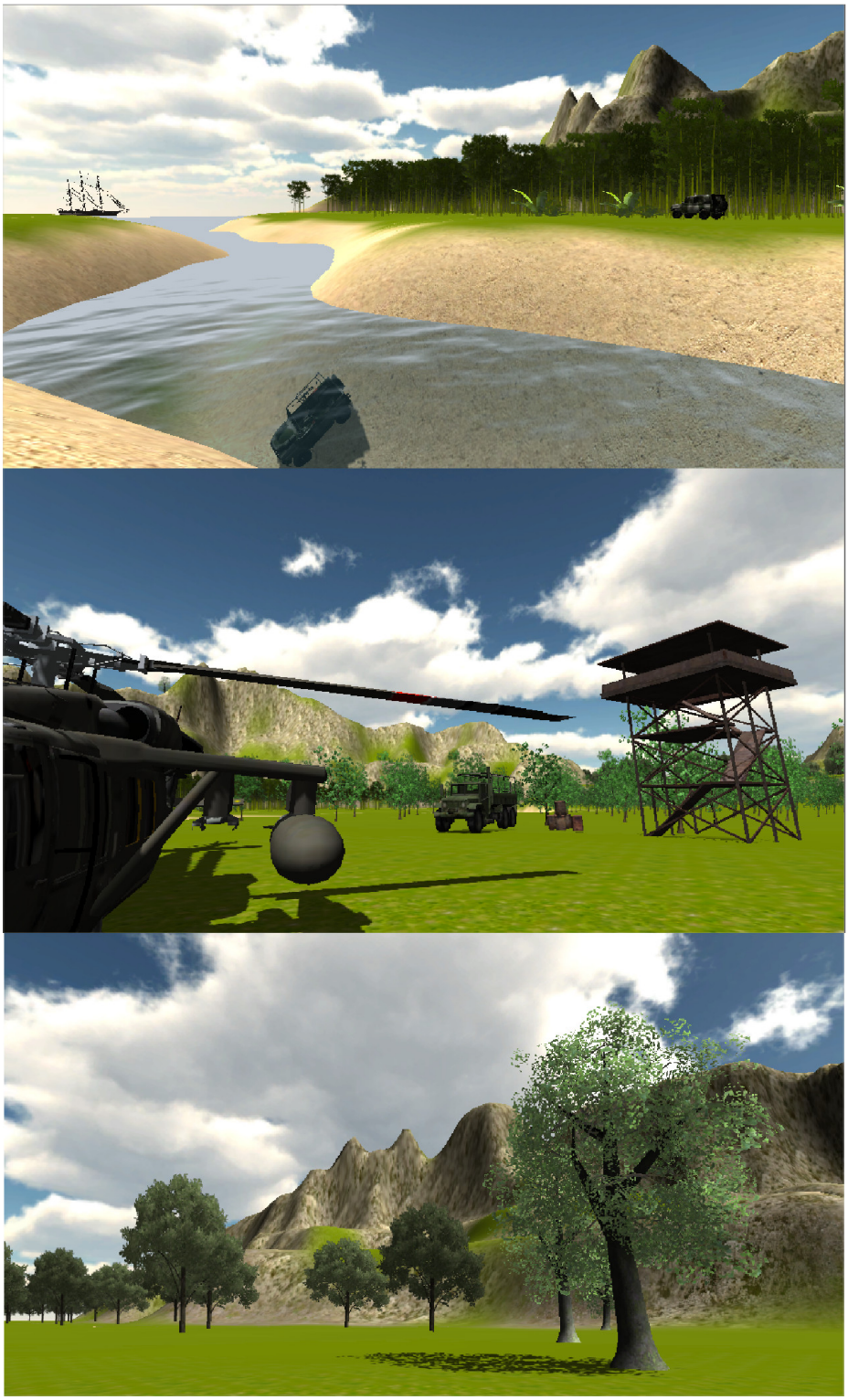


Figure 2.22: Screen-shots from the different locations in the digital environment.

The created scenarios depicted in Figure 2.22 were created to serve as a basis to test our prototype. To provide a slightly better perception of a more realistic environment, trees are animated through shadows and reactions to wind; water is animated through reoccurring wave cycles.

For the above-mentioned hardware and software prototype we decided to collect feedback from users that are not familiar with our prototype and the environment to get an initial understanding which parts need to be improved.

2.3.4 Functionality Test

For our informal functionality tests and to get a first impression from users about our prototype, we conducted a test with 10 users, three female and seven male testers. The testers were in the range of $20 - 35$ years and 9 out of 10 testers have not experienced a wearable projector or display before. Three out of ten testers have an acting background. Nonetheless, for our initial tests we neglected to test the environment as a motion capture acting aid because it was more important at this stage to prove the functionality and to see how users react to the application we built.

In Figure 2.23, the test set-up can be seen. The tester is wearing a strap which holds the projector on the forehead, and holds the phone on the back side of the head. Walls with a retro-reflective foil that reflect the image of the projector into the eyes of the users were placed in front of the testers.

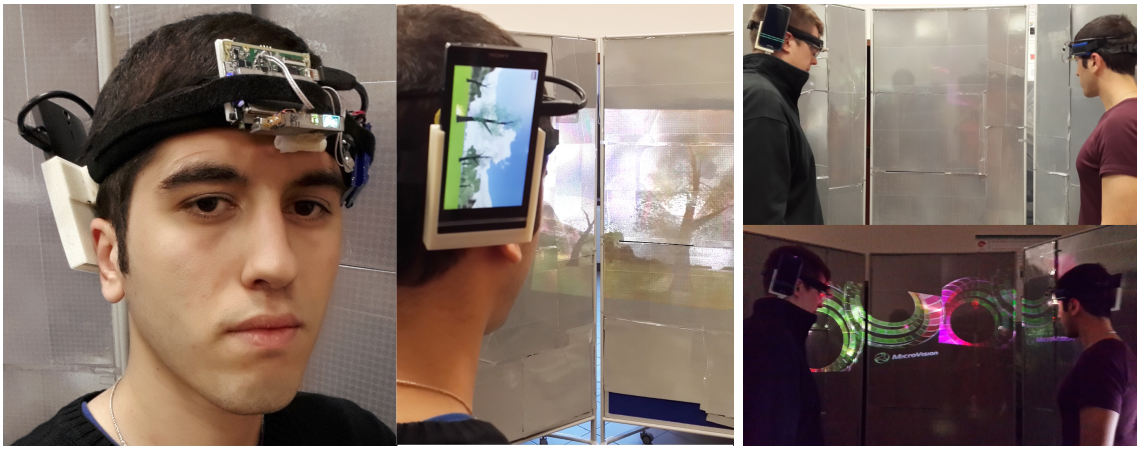


Figure 2.23: On the left, two different photographs of a user with our wearable augmented reality display are shown, the system is composed of a smartphone, an external battery and a pico projector with 3D printed housings. On the right, two photographs show a scenario where multiple users independently benefit from a passive retro-reflective screen without crosstalk.

When conducting the tests, each user was given an introduction to the prototype and its functionality. Then, the users were asked to explore the environment, and to test the prototype at their own pace. Functionality possibilities and hints were mentioned during the tests and a dialog using the think aloud method was performed to understand what the user experiences while testing the prototype. The user tests were also videotaped as reference material and as another source of data collection to evaluate the user reactions and comfort or discomfort while using the prototype. In addition, a questionnaire was filled out by the users to help evaluating their experiences with the prototype.

Evaluating the questionnaires and video recordings revealed that there was a mixed opinion amongst the testers in terms of how comfortable the prototype is to wear. Half of the testers mentioned that they only realized that they were wearing the prototype while performing the tests; the other half did not or rarely notice the worn hardware. When we asked if testers experienced any symptoms of nausea or discomfort while testing, none of the testers experienced any of the mentioned symptoms.

Qualitative feedback towards the question how immersed the testers felt into the

environment while testing was also positive. On a Likert-scale, five testers answered with fully "immersed" and five with "immersed". When we asked how realistic the environment felt, we need to report that the testers gave a fairly mediocre feedback. The reason for this lies within the fact that (i) the reflection area and explore able space needs to be enlarged and, (ii) the walking algorithm and the perception of movements in the digital environment need to be improved as well.

The general feedback from the tests we performed was positive and the testers showed interest in the application. When we asked the testers if they could imagine using this application for private entertainment or training purposes, 4 testers answered on a Likert-scale with "strongly agree", 3 testers answered with "agree" and 3 with "undecided". Our tests also indicated which parts of the application need to be improved to create an even better experience, especially as future work for entertainment applications.

2.3.5 Conclusion

In this work, we have demonstrated a light-weight, low-cost, mobile and ergonomic head-worn projection display using off-the-shelf equipment. The used equipment was modified to create a compact and longer lasting prototype ($3-4$ hours). Our prototype uses a single 15 lm pico projector, and does not require a cable connection to a stationary unit. Perceived images by the users are focus-free, bright and distortion-free. A perceived field-of-view of a viewer is 50^o in diagonal. The vision of the users is not occluded and no hardware is placed in the users field of vision. We have shown that exploring a virtual environment is possible with our prototype, and allows to create a fairly immersive experience. Multiple users can independently use the screen without experiencing as crosstalk. Initial functionality tests have been successfully performed. Furthermore, the feeling of nausea originated from long-time usage was not observed through our initial testing. The prototype can also be used as a 3D stereo system using the same hardware by additionally mounting polarized glasses and active polarization rotator, while maintaining all of the advantages listed above.

The primary utility in publishing will be to enable an inexpensive, lightweight and stand-alone AR HMD for use in research and for motion capture studios.

2.3.6 Future Improvements

As the goal is to use the prototype for motion capture acting aid, a few improvements and changes need to be done to allow the use of the prototype in this specific environment. Improving the controls of the prototype for example, as well as the walking algorithm and smoothing sensor data through filters or sensor fusion needs to be performed to receive a better output.

To prepare the prototype for motion capture acting, the digital environment needs to be enriched with scenarios that are more suitable to test and support common acting tasks in a motion capture environment. This could e.g. include to add more animations and trigger able events.

The next version of the software is expected to provide interaction in between multiple users sharing the same digital content and same screen. Alternatively, there can be multiple screens used and shared by multiple users in remote locations.

Testing and preparing the prototype's physical rigidity must also be performed to allow to wear the prototype for stunt motion capture shoots.

Our functionality tests conclude that different application areas are also exciting to be explored, apart from motion capture application. Especially, in gaming and in collaborative interactions for entertainment applications, we see a great potential for our prototype.

Chapter 3

AUTOSTEREOSCOPIC DISPLAYS

This chapter focuses on our scientific contribution to the domain of autostereoscopic displays, which do not require certain aid such as glasses, binocular, or spatial apertures. The chapter contains two different methods of autostereoscopy, where each one provides unique opportunities in their own use-case scenario.

3.1 Multi-view autostereoscopic projection display using rotating screen

This section introduces a single-viewer autostereoscopic display method, which was published in [K8]. The idea was first raised by Osman Eldeş. The idea was first tested with a collaboration in between Kaan Akşit and Osman Eldeş. The publication was made possible through consensus in between Kaan Akşit, Osman Eldeş, and Hakan Urey. This part of our research had Financial support from TÜBİTAK Project No: 111E183.

3.1.1 Introduction

Autostereoscopic displays form at least two exit pupils, through which a three-dimensional (3D) image is observed without glasses. The exit pupils can be either fixed, or can dynamically follow viewer's head position under the control of a head-tracker. A large viewing field for the viewer in an autostereoscopic projection display can be achieved by creating tens of exit pupils using the parallax barrier approach [77], or using an array of projectors [78, 79]. Alternatively, one can design a display tracking the viewer's eye using dynamic exit pupils [80, 81, K14]. The main advantage of tracking displays is that full resolution is achieved in each view, but they are typically limited to one or few viewers. Projection based autostereoscopic displays employ

various transfer screens to form an exit pupil, such as retro-reflective light diffusing screen [78, 80], double lenticular screen [79], or Fresnel lens in front of a light shaping diffuser [82, K14].

In this work, we propose a novel autostereoscopic projection display technique which employs a transfer screen to form a pair of dynamic vertical viewing slits aligned with the viewer's eyes. Any of the aforementioned transfer screens can be used for the proposed technique. The viewing slits track the viewer's eyes in a large viewing field by rotating the transfer screen in-plane. Advantages of the proposed technique are as follows: it requires only two projectors rather than an array of projectors; there is no image registration problem on the screen due to movement of projectors; the viewing slits always track the viewer, so the viewer never perceives discrete transitions between different perspectives; the technique can provide high-gain, and sufficient brightness even with a pair of mobile projectors. The main limitation of this technique is that it is suitable for only one viewer. Luminance, crosstalk, and dynamic viewing area analysis, and measurement results as well as a video demonstration of the system are presented.

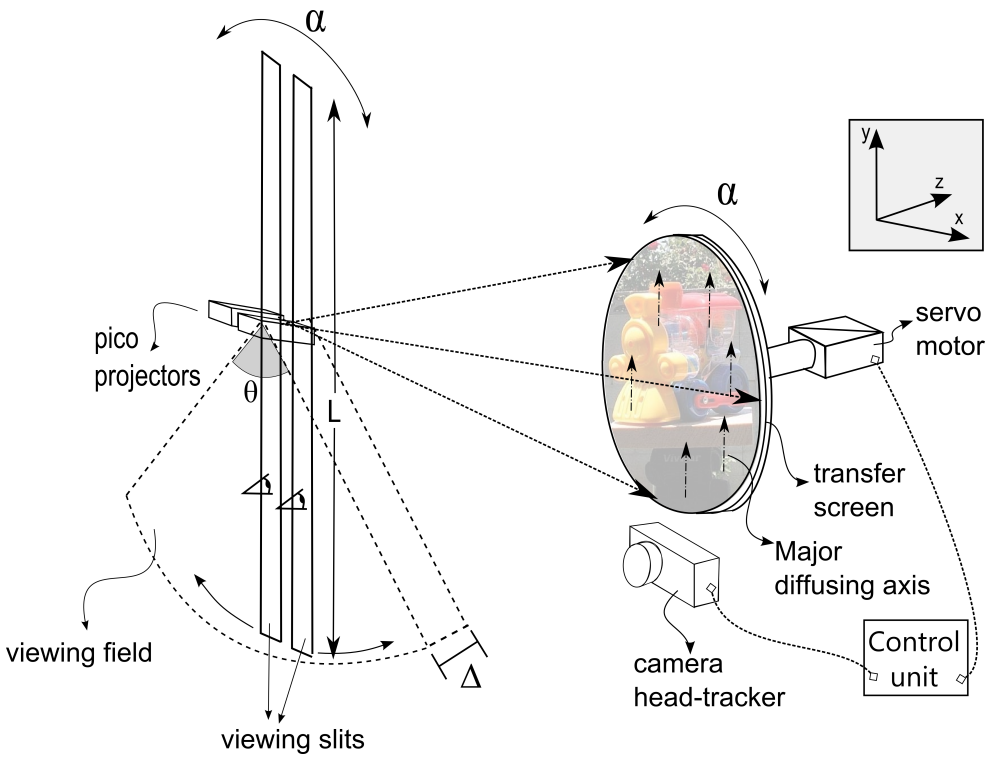


Figure 3.1: System sketch showing the elements and the created viewing field of the display.

3.1.2 Concept of the display

The system sketch depicted in Fig. 3.1 shows two pico projectors, a rotating transfer screen, a head-tracker unit, and a control unit. The stereo content is projected onto the transfer screen by two pico projectors which are placed horizontally apart from each other by average human interpupillary distance (IPD), 63 mm. Each projector is assigned to one eye of the viewer. One projector projects the content for the right eye perspective and the other projects the content for the left eye perspective. On the plane of projectors, as shown in Fig. 3.1, the transfer screen creates two viewing slits, each of which is parallel to the major diffusing axis of the single axis diffuser, and crosses over the position of corresponding pico projector's micro electromechanical system (MEMS) based scanner. Each viewing slit contains the image content projected by the corresponding pico projector. The creation of viewing slits by using a retro-reflective diffuser screen, as transfer screen, has been explained in [83]. A viewer, who is standing

in the plane of pico projector and looks through the viewing slits, perceives stereo images.

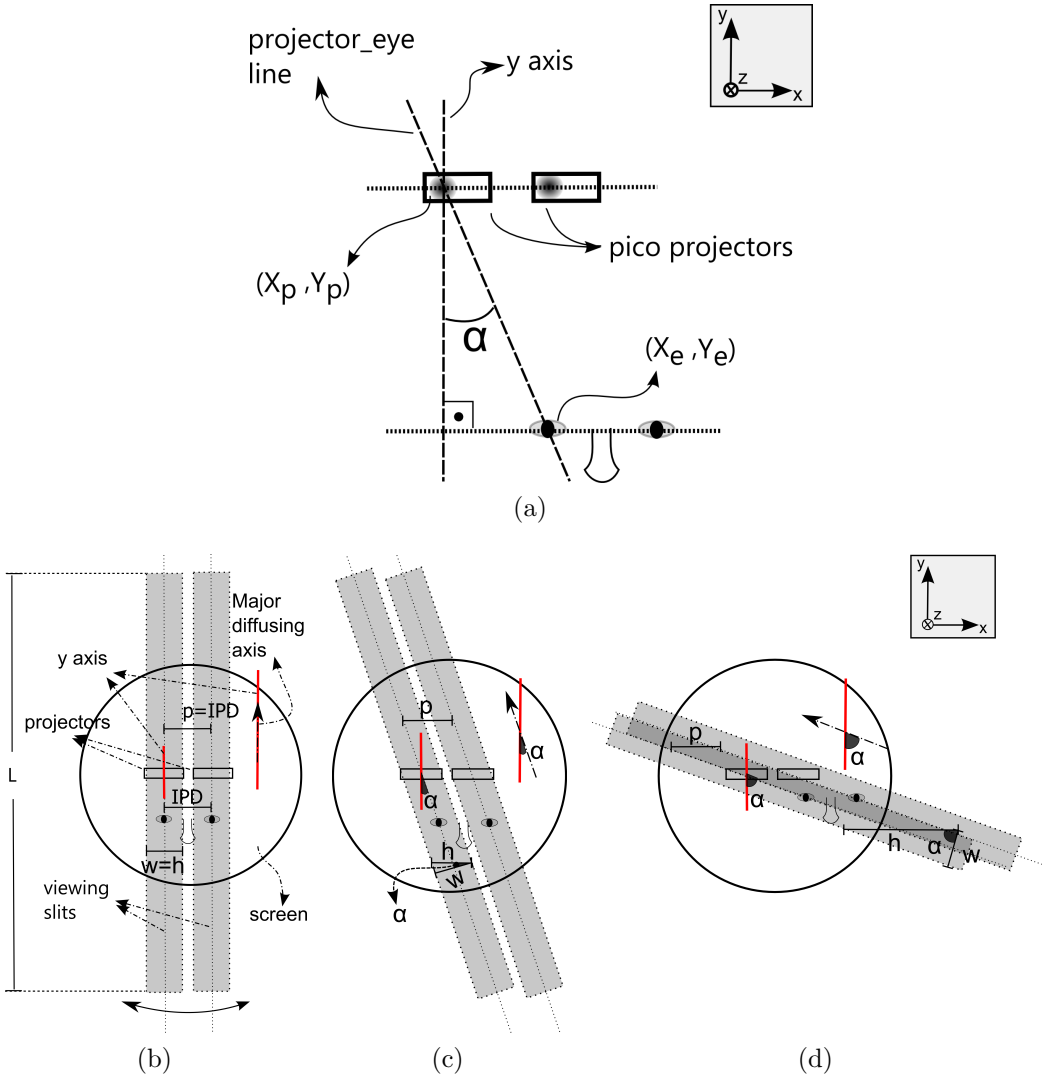


Figure 3.2: Viewing slits for different orientations of the transfer screen. (a) Relative angular position of viewer's eyes with respect to projectors (b) Rotation of transfer screen by 0^o of α and $w < 2 \times IPD$, thus $h = w < 2 \times IPD$ (c) Rotation of transfer screen by small α, thus $w < h < 2 \times IPD$ (d) Rotation of transfer screen by large α, thus $h > 2 \times IPD > w$, and there is crosstalk between viewing slits.

In order to change the position of viewing slits according to the position of viewer's eyes, a head-tracker unit tracks the position of the viewer, and sends the position information to the control unit. Using the formula :

$$\alpha = arctan(\frac{X_e - X_p}{Y_e - Y_p}), \tag{3.1}$$

the control unit calculates the angular position of the viewer, α, which is the angle between eye-projector line and y axis, as shown in Fig. 3.2a. Using the angle, α, a servo motor rotates the transfer screen in-plane such that the major diffusing axis of the diffuser makes the same angle with y axis as the eye-projector line does, as in Fig. 3.2. Thus, viewing slits dynamically track the viewer in a large viewing field, as depicted in Fig. 3.1. The motion of the viewing slits is analogous to the in-plane rotation of a beam around an anchor point. The MEMS scanner of the pico projector is the anchor point of the corresponding viewing slit.

By projecting different perspective images for the corresponding positions of the viewer, a multi-view 3D display is realized. The proposed technique provides multi-view stereo vision to a single viewer in a large viewing field, which is an arc of a circle with projectors at its center, as depicted in Fig. 3.1.

3.1.3 The prototype

A prototype was realised to demonstrate the capabilities of the proposed technique. The prototype consists of two MEMS based laser pico projectors from Microvision, Inc. [1], a retro-reflective diffuser screen as a transfer screen [83], an in-house made head-tracker unit [84], an in-house made rotating mechanism which rotates the transfer screen around its center, and a personal computer.

The transfer screen is a layered composition of a retro-reflective sheet [71] from Reflexite [85], and a single axis diffuser (large diffusing angle in major axis and small diffusing angle in the perpendicular minor axis). Two different types of single axis diffuser have been tested as the diffuser layer of the transfer screen. Diffuser 1 is a randomly patterned aperiodic diffuser from Luminit, LLC [86] with FWHM diffusing angle, ϕ, of 40^0 in major axis, and, ψ of 0.2^0 in minor axis. Diffuser 2 is a conventional periodic diffuser with FWHM diffusing angle, ϕ, of 20^0 in major axis, and, ψ of $\sim 0^0$ in minor axis. The retro-reflector sheet is a high fill factor 200 $mm \times 200$ mm corner

cube retro-reflector array with 0.254 mm pitch size. As the screen size is equal to the retro-reflector sheet size, it can be enlarged using an improved mechanical design to support a larger screen.

(a) (b)

Figure 3.3: (a) Created viewing slits at different rotation angles: 9 shots are superimposed in order to create the photograph. The two bright spots in the photograph are pico projectors. (b) A sample picture of viewer, showing viewing slits on his eyes' position.

It is enough to rotate only single axis diffuser in order to create dynamic viewing slits. However, in the prototype, rotating mechanism rotates both retro-reflector sheet and the single axis diffuser which makes the realization of the prototype easier. The created viewing field is illustrated in Fig. 3.3a by taking pictures of viewer space for different rotation angles of viewing slits. Figure 3.3b shows a sample picture of viewer while viewing slits appear on his eyes' position.

3.1.4 Results and discussions

As a quality measure, a crosstalk analysis of the realized prototype for both diffusers was conducted. The crosstalk has been quantized by:

$$crosstalk(\%) = \frac{leakage}{signal} \times 100, \tag{3.2}$$

where leakage is the maximum luminance of light that leaks from unintended channel to the intended channel, and 'signal' is maximum luminance of the intended channel, [87].

Thus, two luminance measurements of the screen are taken from each eye's position to calculate the crosstalk value of the corresponding eye. In the conducted experiments, only crosstalk values for the left eye has been measured, since the system is approximately symmetrical. For the first luminance measurement, which determines the leakage, full black image is projected by left-eye projector, and full white image is projected by right-eye projector. For the second measurement, which determines the signal, images for the first measurement are swapped. Luminance values have been measured by a calibrated camera from Radiant Imaging, which takes photometrically weighted photographs, and inserted into Eq. 3.2. By repeating the explained procedure above for different positions of camera in viewer's space, and by interpolating the measured values; crosstalk, and luminance maps of viewer's space were obtained for both transfer screen with diffuser 1, and transfer screen with diffuser 2.

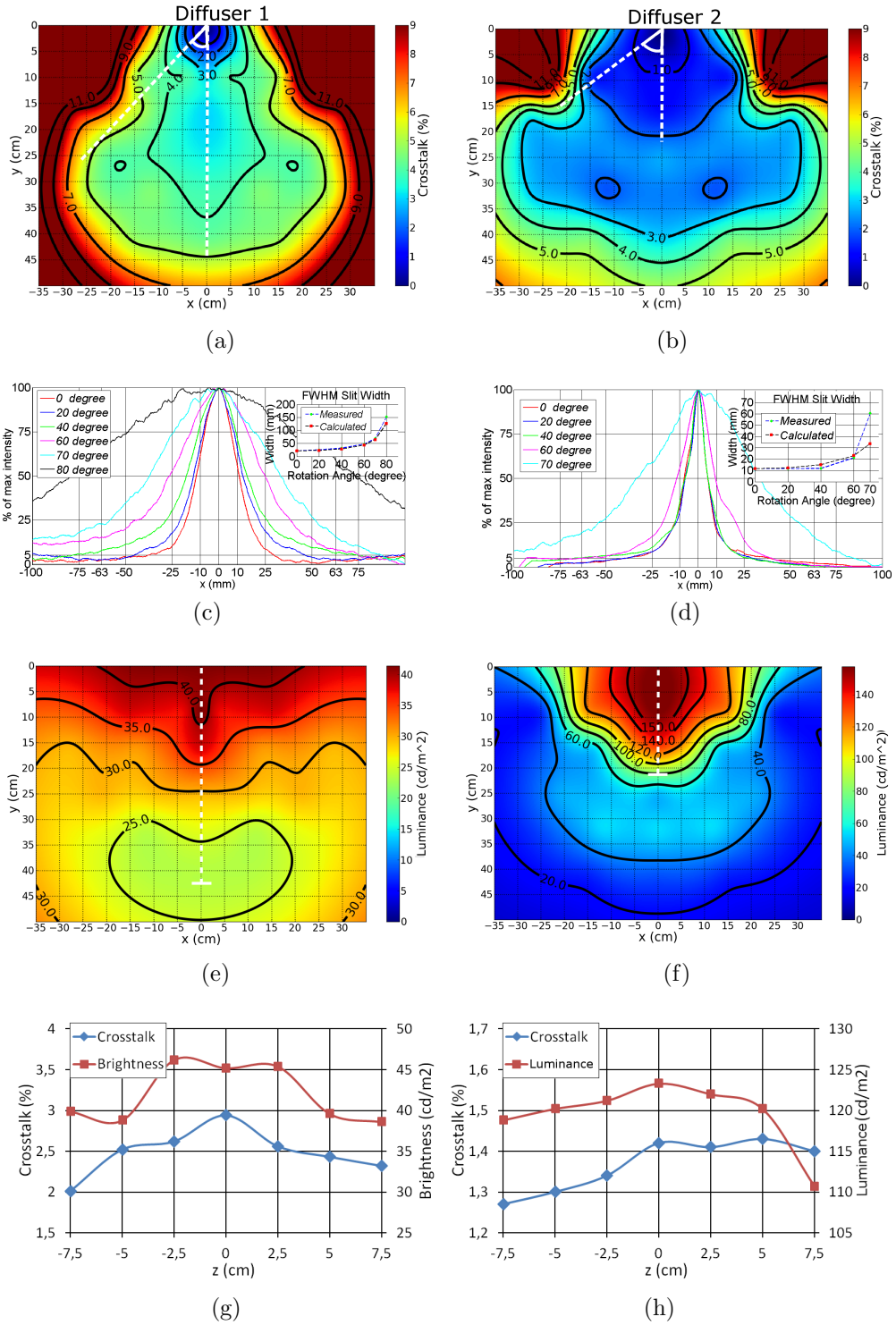


Figure 3.4: (a) and (b) Interpolated crosstalk maps of viewer's space at the projector plane for diffusers I and II. (c) and (d) Horizontal cross-sections of viewing slits for different rotation angle, a for diffusers I and II. (e) and (f) Interpolated luminance map of viewer's space at the projector plane for diffuser 1 and 2. (g) and (h) Crosstalk and luminance variations along the projection axis for diffuser 1 and 2 at the position (x,y) = (0,9)cm and z is variable in the measurements.

Figure 3.4a is the obtained crosstalk map of the viewer's space at the projector plane for screen with diffuser 1. According to [88], crosstalk should be less than 5 % in order to prevent reduced viewing comfort in half of the population. Thus the maximum rotation angle, θ, of viewing slits is 46^o, beyond which the crosstalk is more than 5 %, and results in inseparable stereo images. Figure 3.4c illustrates the horizontal cross-section of viewing slits for different rotation angles. The actual width, w, of viewing slits, which is the full width at 5 % of maximum intensity, for 0^o of rotation, is 75 mm. By placing the actual width, w, of viewing slit, the maximum rotation angle, θ, of viewing slits is calculated as 53^0, which validates the experimental result of 46^o.

Figure 3.4e is the luminance map of viewer's space in projector plane. The luminance of the display decreases to 50% of the maximum luminance at around 420 mm away from the center of viewing slits, which are located at (0,0) in the coordinate system. Thus, the FWHM length, L, of viewing slits is 840 mm.

Figure 3.4g shows the crosstalk, and luminance variations over the projection axis. The crosstalk of the display is below 5 % for ± 7.5 cm away from the projector plane. Thus, the viewer still perceives stereo images away from the projector plane. However, as the viewer moves away from the projector plane, perceived luminance varies over the transfer screen.

In order to increase the viewing field by increasing the maximum rotation angle, θ, of viewing slits, another transfer screen has been constructed by replacing diffuser 1, $40^o \times 0.2^o$ aperiodic single axis diffuser, with diffuser 2, $20^o \times 0^o$ periodic diffuser. Crosstalk, and luminance maps of the viewer's space have been obtained for the screen with diffuser 2, and presented in Figs. 3.4b, 3.4f, and 3.4h. The crosstalk of the screen with diffuser 2 is less than the diffuser 1, for the same area of viewer's space, as presented in Fig. 3.4a. The maximum rotation angle, θ, of viewing slits is 58^o for the prototype 2, and it is more than the maximum rotation angle, θ, of viewing slits for the diffuser 1, which is 46^o. Figure 3.4d illustrates the horizontal cross-section of viewing slits for different rotation angles. As seen in Fig. 3.4d, the actual width, w, of

viewing slits, which is the full width at 5 % of maximum intensity for 0^o of rotation, is around 60 mm. By placing the actual width, w, of viewing slit, the maximum rotation angle, θ, of viewing slits is calculated as 61^o, which validates the experimental result of 58^o.

Figure 3.4f is the luminance map of viewer's space in projector plane for the the screen with diffuser 2. As seen in Fig. 3.4f, the luminance of the display decreases to 50% of the maximum luminance at around 210 mm away from the center of viewing slits. Thus, the FWHM length, L, of viewing slits is 420 mm. By placing the prototype 2 parameters, which are the projection distance, d, of 1180 mm, and the FWHM diffusing angle, ϕ, of 20^o, the FWHM length, L, of viewing slits is calculated as 416 mm, which validates the experimental result of 420 mm.

Figure 3.4h shows the crosstalk, and luminance variations in projection axis for the screen with diffuser 2. The crosstalk for the diffuser 2 is less than the crosstalk for diffuser 1, for ± 7.5 cm away from the projector plane, as presented in Fig. 3.4g

As the screen in prototype is a high gain retro-reflective diffusing surface, all generated power is concentrated into the viewing slits, and luminance is always reasonably high in the viewing slits regardless of the projection distance. Retro-reflective surfaces can have gains of between $1k - 10k$ [89]. Thus, projection distance, d, and screen size, s, is not limited by the projector power. However, it should also be noted that, as the retro-reflector screens work under a specific acceptance angle limit, the projection angle, β can not exceed the acceptance angle limit. In the prototype, projection angle, β, is $9,68^o$, which is smaller than the acceptance angle, 30^o, of the retro-reflector.

3.1.5 Conclusion

A new type of multi-view autostereoscopic projection display, using a pair of MEMS scanner based pico projectors, a head-tracking camera, and a rotating retro-reflective diffuser screen is proposed and demonstrated. Analytical and experimental results for crosstalk and luminance are in good agreement. The screen with diffuser 1 achieved a

viewing field of 500 mm in horizontal axis, and 450 mm in vertical axis. The screen with diffuser 2 achieved a viewing field of 700 mm in horizontal axis, and 500 mm in vertical axis. The crosstalk of the display is below 5 % for both diffusers. The maximum luminance of the display is 50 cd/m^2 and 160 cd/m^2 for diffuser 1 and diffuser 2, respectively. The crosstalk remained below 5 % within ± 7.5 cm in the z-axis, i.e., user can move back and forth about 15 cm and the crosstalk remain at acceptable levels while the luminance dropped by 25 % and 10 % for Screen 1 and Screen 2. Beyond that range, the luminance at the edges of the transfer screen decreases and corners of the images are vignetted.

The technique introduced in this work provides a large-screen autostreoscopic projection display using a pair of low-lumen projectors for a single viewer. Full resolution of the display is maintained. Retro-reflective screen provides high brightness gain. The transition between different perspectives is smooth as viewing slits track the eyes real-time. Some of main applications are personal entertainment displays used in cars, planes, and simulators.

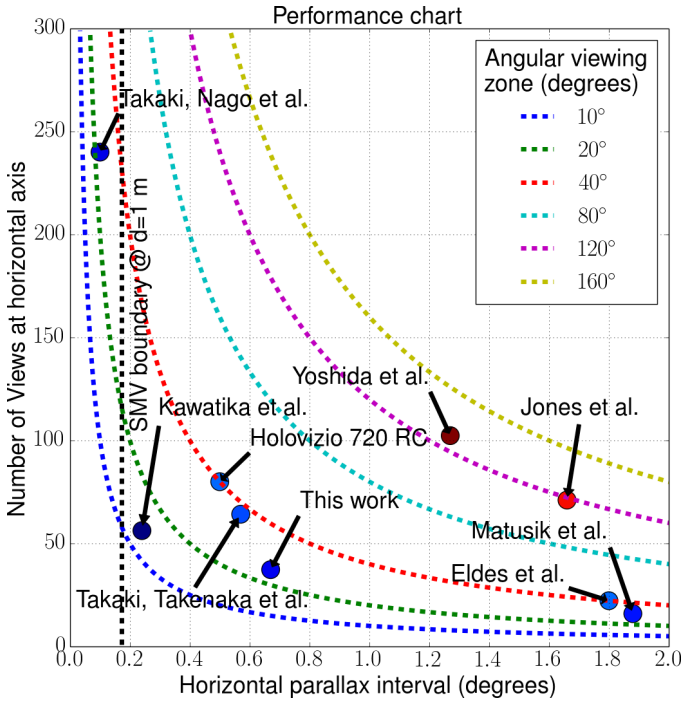
3.2 Modular Multi-view Autostereoscopic Display using MEMS Projector Array

This section introduces a modular multi-viewer autostereoscopic display method, which will be published in the near future. The idea was first raised by Hakan Urey. The idea was first tested with a prototype by Kaan Akşit, and Selim Ölçer. The text was made possible through consensus in between Kaan Akşit, and Hakan Urey. This part of our research had Financial support from TÜBİTAK Project No: 111E183.

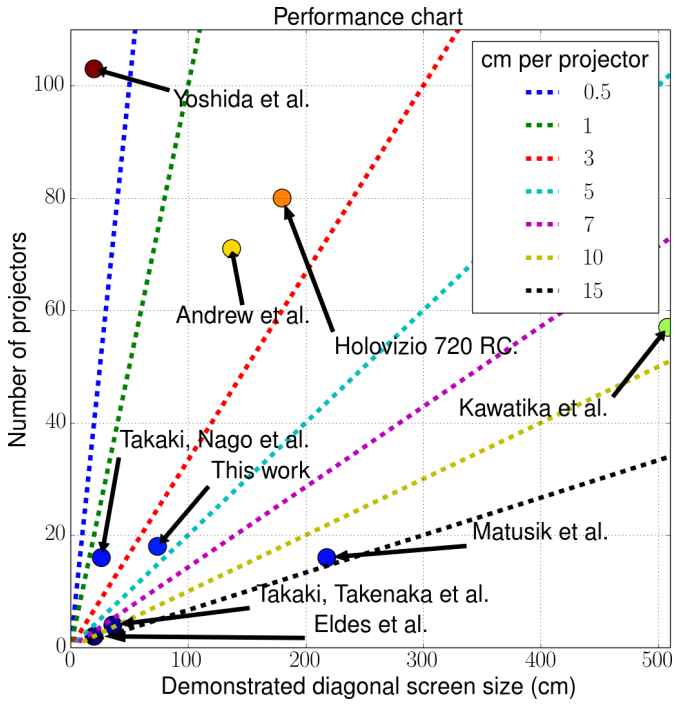
3.2.1 Introduction

A common way of building glasses free three dimensional (3D) displays [16] is based on using multiple projectors with a screen assembly. Projection based multi-view autostereoscopic displays [3–8, 90, K8] are capable of displaying different perspectives depending on the position of the viewers. There are two major performance parameters for such displays, which are the angular horizontal parallax interval, and the number of views at horizontal axis. The state of the art displays can provide also vertical parallaxes through pupil trackers [8]. We made two charts that are given in Figure 3.5 to show the differences among demonstrators found in the literature. For the chart (a) shown in 3.5, angular viewing zone in front of the screen can be back calculated by simply multiplying the number of views by the horizontal parallax interval. Different angular viewing zones are shown as curves in Figure 3.5. On the other hand, Super Multi-view (SMV) concept [42, 45, 46] proposes to provide more than one perspective to a single eye to improve the monocular parallax. The boundary to satisfy the SMV condition at 1 m viewing distance is highlighted in Figure 3.5, in which parallax intervals smaller than the boundary satisfies the SMV condition. An ultimate solution would have smallest horizontal parallax with maximum number of views. For the chart (b) found in Figure 3.5, an ultimate solution would have the smallest number of projectors with maximum diagonal screen size. Careful readers immediately recognizes for both charts, there is no certain winner. The better qualifications comes with certain trade offs. Our motivation is to provide easily scalable miniature size system

with the best possible parameters.



(a)



(b)

Figure 3.5: Performance chart showing different projection based multi-view autostereoscopic displays [3–8, K8] from the literature and our system. Chart (a) shows Super Multi-view (SMV) boundary when the viewer is 1 m away from the screen and has eye pupil size of 6 mm.

The display demonstrators compared in Figure 3.5 except our proposal have two-pieced screen assembly, where a vertical diffuser is combined either with a lenticular lens or a retro-reflector foil. An ultimate solution to the problem of building glasses free 3D displays would be to use a lens-free system built with a two-dimensional projector matrix [91]. In such a system, a seamless 3D experience can be presented without any gaps appearing on the screen with a center-to-center spacing of 2 mm in between projectors, when they are positioned in an arc geometry. Such geometric orders would consume too much volume behind the screen and limit the screen size, whereas in our case screen size scalability is preserved with a seamless 3D experience without any gaps through 9 mm center-to-center projector spacing and a vertical diffuser ($40° \times 0.2°$). Note that our proposal offers an extreme compact light engine behind the screen with multiviewer support, which was not demonstrated until now. A single pico projectors' photonics module requires only 35 $mm \times 20$ $mm \times 8$ mm volume. Our previous work [K8] did not provide multiviewer support, and occupied more space.

We have introduced a new modular multi-user multi-view autostereoscopic display structure based on off-the-shelf Microelectromechanical systems (MEMS) laser projectors, where no longer complex screen assembly is required. Unlike similar type of displays, the volume of the projection display's light engine decreased dramatically. The proposed architecture is highly scalable in screen size. The main disadvantage which we have observed in our prototype is the color match problem among the projectors. We have observed that each projector can differ in terms of output illumination power, which did not disturb the viewers during the informal prototype tests.

3.2.2 Method

Our proposal is sketched in Figure 3.6, in which an array of MEMS projectors with a vertical axis diffuser is shown. A single image projected by a projector contains different portions from different perspectives. Depending on the relative angular position of a viewer to that single projector, a different portion of the image will be

visible to the viewer. The vertical diffuser helps the viewers to perceive a complete image from the screen, and gives the freedom to see the full image as the viewers move in vertical axis.

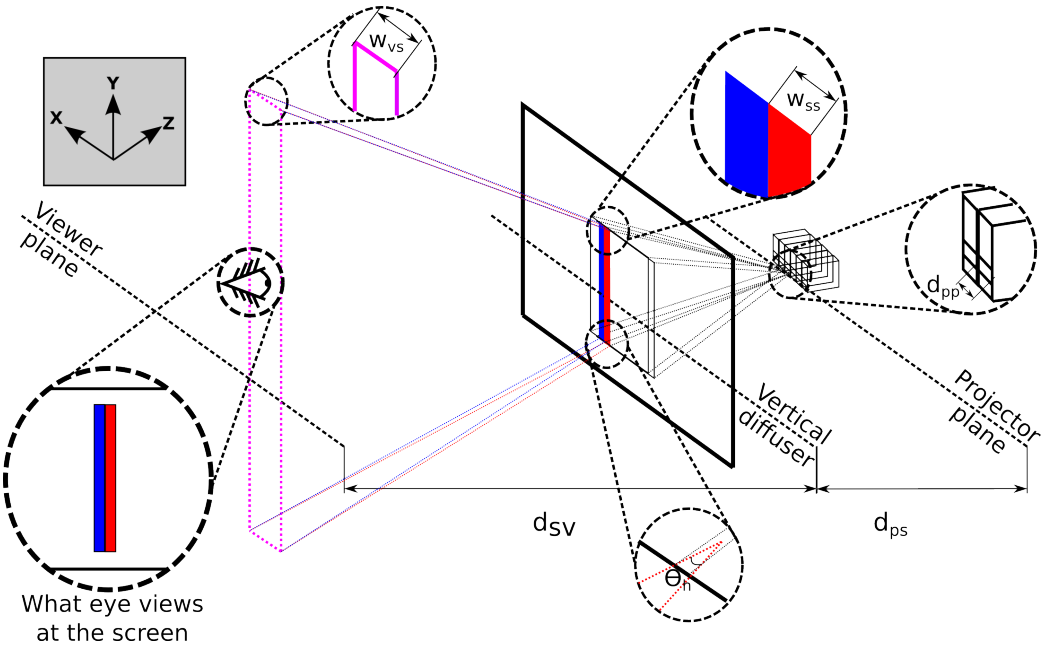


Figure 3.6: Sketch showing the overall system consisting of projector array with a vertical diffuser. The different rays from different pico projectors from different portions of the screen are arriving at a viewer's eye.

Figure 3.7 provides a top view of the setup, here d_{ps} represents the distance between the projector plane and the vertical diffuser, d_{sv} represents the distance between the viewers and the vertical diffuser, θ_h represents the diffusion angle of the vertical diffuser at horizontal axis, d_{pp} represents center-to-center spacing between projectors, and θ_{ph} represents the projection angle of the projector at horizontal axis. Note that single projector projects an image with an uneven geometry, this is due to the 90° rotated projectors. When only a single projector is operational, a single eyed viewer will see a single narrow slit on the screen. The width of the single slit at horizontal axis w_{ss} is a function of θ_h, d_{sv} and d_{eye}. The correlation in between mentioned parameters is formalized by using geometrical optics:

$$w_{ss} = 2 \times d_{sv} \times tan(\frac{\theta_h}{2}) + d_{eye}. \tag{3.3}$$

Note that a typical human eye has a pupil diameter d_{eye} of $2-8$ mm at horizontal axis. For example, w_{ss} can be calculated as 8.86 mm, when $d_{sv} = 1400$ mm, $\theta_h = 0.2°$ and $d_{eye} = 4$ mm are selected.

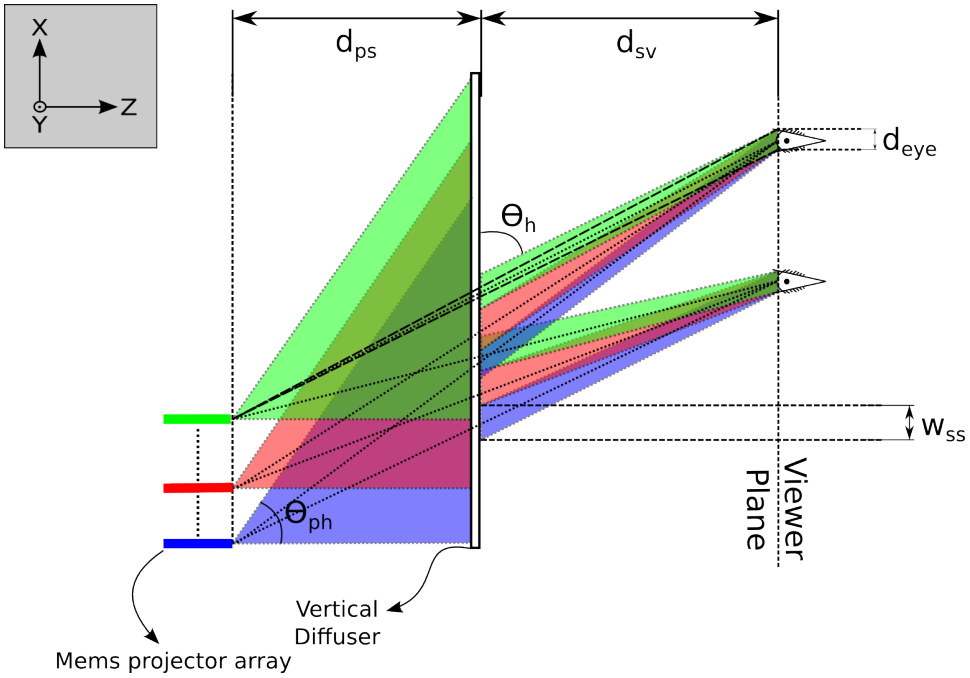


Figure 3.7: Sketch showing rays from different pico projectors from different portions of the screen are arriving at a viewer's eye, and forming a complete image on the screen.

Number of views n_v in the proposed system can be also formalized as in Equation:

$$n_v = \frac{tan\theta_{ph} \times d_{ps}}{w_{ss}}, \tag{3.4}$$

using w_{ss}, d_{ps} and θ_{ph}. For example, n_v can be calculated as 37 when $w_{ss} = 8.86$ mm, $\theta_{ph} = 22.5°$, and $d_{ps} = 800$ mm are set.

Another important parameter of the system is the screen width w_{sw}, which can be engineered by changing d_{ps} and the number of projectors n_p. Equation 3.5 gives w_{sw}

using geometric optics:

$$w_{sw} = \frac{tan\theta_{ph} \times d_{ps}}{n_v} \times n_p. \tag{3.5}$$

Note that number of the views n_v has no relation with the number of projectors n_p. The screen width w_{sw} can be calculated as 161 mm, when $n_p = 18$, $d_{ps} = 800$ mm and $\theta_{ph} = 43.5°$ are chosen. On the other hand, screen height h_{sh} depends on d_{ps} and the projection angle at vertical axis θ_{pv}. Equation 3.6 formalizes the correlation in between h_{sh}, θ_{pv}, and d_{ps}:

$$h_{sh} = tan\theta_{pv} \times d_{ps}. \tag{3.6}$$

For example, h_{sw} can be calculated as 720 mm when $\theta_{pv} = 42°$ and $d_{ps} = 800$ mm are chosen. The viewing zones created at the viewer plane are discrete zones as in the case of previous studies. Thus, transitions between different perspectives are visible to the viewers when they want to move from one perspective to an another. Additionally, the system does not provide a vertical parallax, and the depth information with fixed offsets changes as d_{sv} changes. Such problems can be solved easily through tracking [8].

3.2.3 Content creation

Existing multi-view content can not be directly fed into our proposed system. A proper conversion method is required, which we have built such a method in-house. Figure 3.8 provides a sketch for a scenario, where a system with three projectors are fed with two different perspectives. The procedure can be summarized as follows: the first slit from the first perspective is fed into the first projector's content, the second slit from the first perspective is fed into the second projector's content with a certain offset. The procedure goes on until all the projectors are fed with the first perspective image. Using the same offset, the other perspectives are also fed into the projectors. Note that offset information for all projectors is required to display the content properly on the screen.

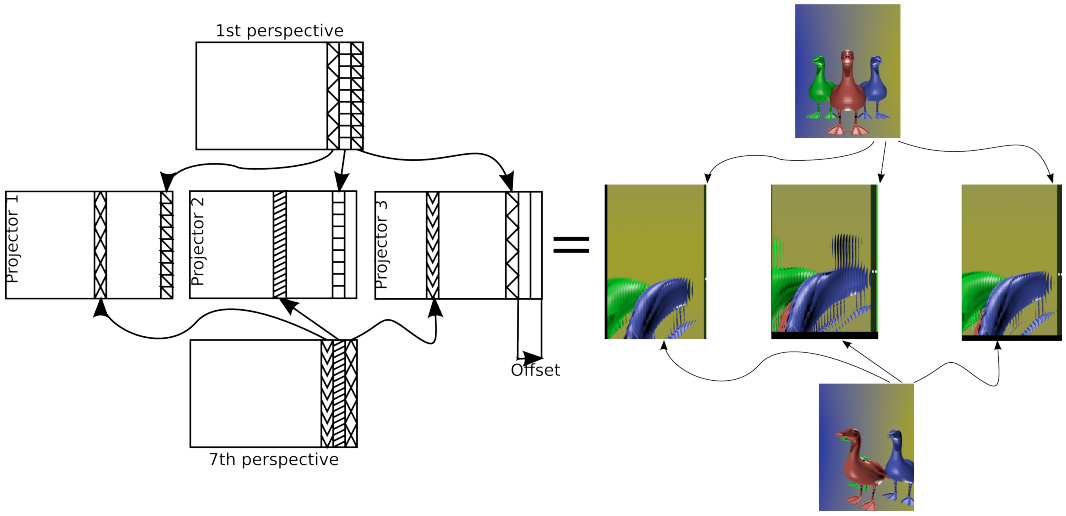


Figure 3.8: Sketch showing content creation, in which two perspective images are fed to three different projectors.

Offsets can be determined either manually, or in an automatic fashion. A single eyed observer watching the screen from the viewer plane can easily detect if each slit from each projector matches to form a single image. An image processing algorithm detecting the registration status of the image can actively change offset until it reaches to an optimum point, as we have done using our prototype, a smart phone, and in-house algorithm. On the other hand, single slit width in pixels w_p is determined by the number of views n_v and the amount of the pixels projected by a single projector at the horizontal axis n_{ph}. Single slit width $w_p = n_{ph}/n_v$ can be calculated as 13 px, when $n_v = 37$ and $n_{ph} = 480$ px are set.

3.2.4 Prototype

A prototype is constructed to demonstrate the capabilities of the proposal using off-the-shelf equipment. The specifications of the constructed prototype are listed in Table 3.1.

Table 3.1: Prototype Parameters

Parameter	Symbol	Value
Interpupillary distance	d_{IPD}	65 mm
Eye pupil diameter at horizontal axis	d_{eye}	4 mm
Vertical diffuser to viewer plane distance	d_{sv}	1400 mm
Projectors to vertical diffuser distance	d_{ps}	800 mm
Number of projectors	n_p	18
Center-to-center Spacing in between projectors	d_{pp}	9 mm
Projection angle of the projectors at horizontal axis	θ_{ph}	22.5°
Projection angle of the projectors at vertical axis	θ_{pv}	42°
Illumination from a single projector	I_p	15 lm
Diffusion angle of the vertical diffuser at horizontal axis	θ_h	0.2°
Diffusion angle of the vertical diffuser at vertical axis	θ_v	40°
Width of a single slit at horizontal axis	w_{ss}	8.86 mm
Single slit width in pixels at horizontal axis	w_p	13 px
Screen width	w_{sw}	161 mm
Screen height	h_{sw}	720 mm
Screen resolution at horizontal axis	X	234 px
Screen resolution at vertical axis	Y	848 px
Number of views	n_v	37
Angular viewing zone at horizontal axis	θ_{zone}	25°
Horizontal parallax interval	$\theta_{parallax}$	0.68°

There are three main components in our prototype. The first component is the vertical diffuser from Luminit with 40° of diffusion angle at vertical axis, and 0.2° at horizontal axis. The second component is the MEMS projector SHOWXX+™ from Microvision. We use 18 of them to form an array behind the vertical diffuser. Each projector's light engine and the control circuitry was taken from their case, and we had 3D printed a new piece in-house to stack the projectors as an array. Thus, we have achieved 9 mm spacing from center-to-center in between projectors. The third component of the prototype is the control unit, which drives the projector array with the necessary content. We used Rasberry PI brand mobile computers. A single Rasberry PI has the capability to drive a single projector through the HDMI port. We modified Rasberry PIs in terms of software so that a single one of them can drive up to 6 projectors through USB external graphics card. Thus, for our prototype, we

used only 3 Rasberry PIs, this leads to an extremely cost efficient modular system.

The in-house built control software has the capability to convert a multi-view content into a proper format, to control the content on each projector, and to provide remote control over the local area network. Figure 3.9(a-b) shows the overall prototype, and the projector array. Note that at Figure 3.9(c), although each projector has a limited illumination of 15 lm, the screen easily saturates a camera looking at it.

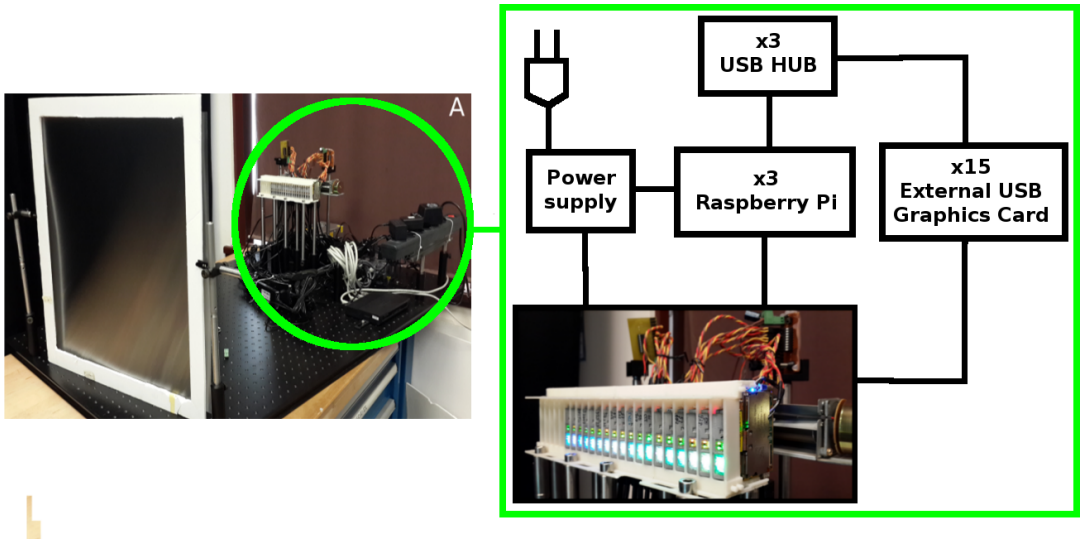


Figure 3.9: Photographs showing a front view of the over all prototype consisting of 18 projectors, 3 mobile computers, and a vertical diffuser. Additionally, 3D printed projector housing with 18 pico projectors and the unseen hardware pieces are shown on the right.

3.2.5 Results and discussion

Figure 3.10 shows photographs of the different perspectives captured from our prototype, when an observer changes position at horizontal axis on the viewer plane. Although the angular viewing zone θ_{zone} covers 25°, we choose to use exaggerated motion parallax of the content so that the effect of moving in front of the screen on the horizontal axis would be well noticeable by the viewers.

Each MEMS projector provides an illumination of 15 lm. When used as a stand-alone projector, the illumination power is fairly enough to project an image with the

size of an A4 paper. But in our system, due to the high number of projectors and the vertical diffuser, image looks very bright even in sizes $\times 4$ larger than A4. We have measured the luminance level of the display as 200 cd/m^2 using a calibrated camera from Sphere Optics with $F\# = 8$, and exposure time of 10 ms are used. This represents a worst case scenario for the human eye, where eye pupil diameter is around the minimum possible level of 2 mm. Luminance level is found to be homogeneous across different perspectives.

The system provides a high level of expandability in screen size due to its modular structure. It is possible to switch to a different vertical diffuser, distances and higher number of projectors. Thus, the proposal is a good candidate for large screen applications.

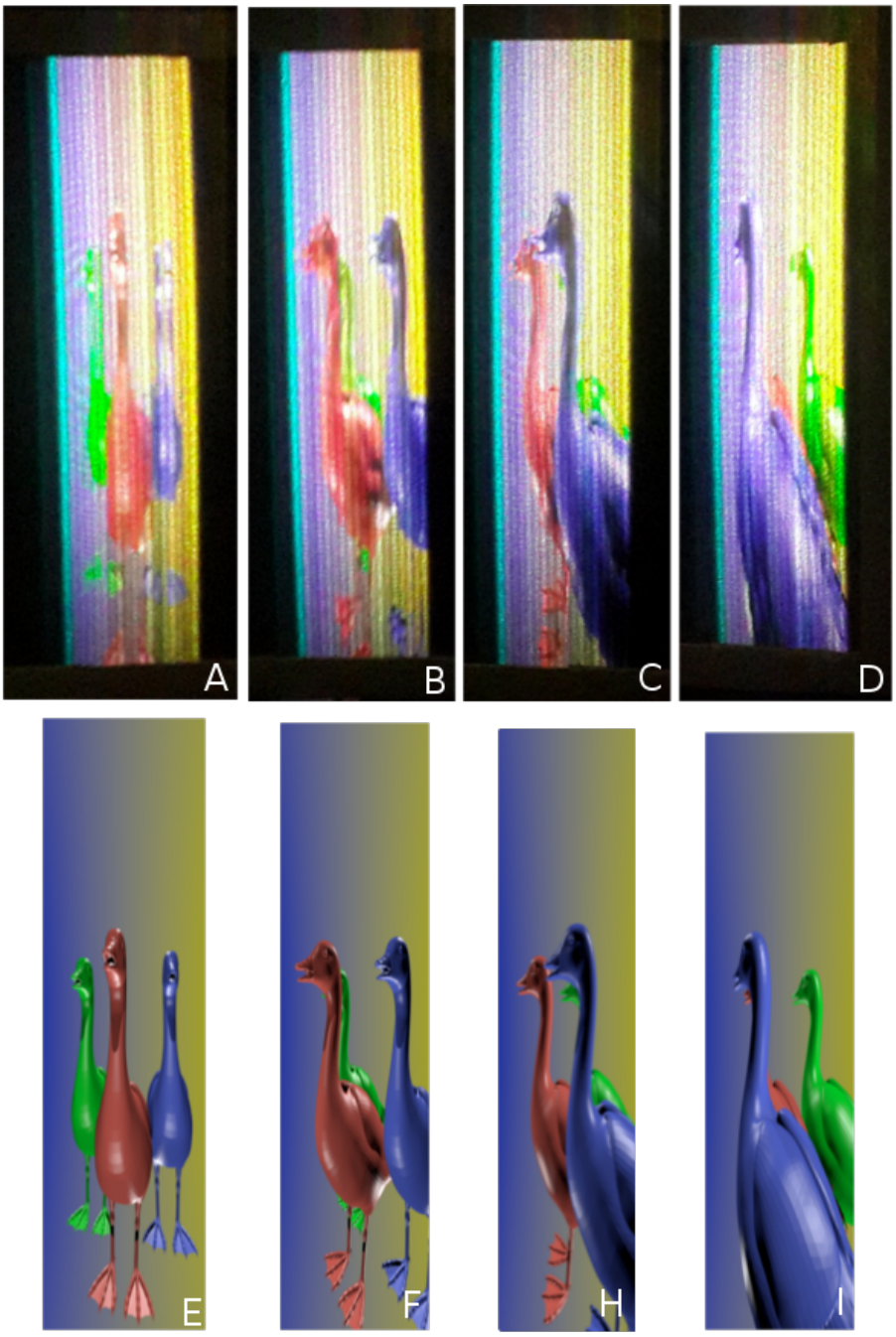


Figure 3.10: Photographs showing (a-d) different perspectives when the camera is placed at a different positions at horizontal axis on the viewer plane.

An observed issue is the visible wave pattern on the screen at the horizontal axis. The problem is originated from the variance in the total illumination power of each

projector. Another issue that has to be raised is the lack of color balance between slits appearing on the screen, which is highly visible in Figure 3.10. This behavior is originated from the light source of the projectors. Each projector acts differently in this sense. Through informal subjective tests in our laboratory, the viewers found the 3D experience nice and satisfactory.

3.2.6 Conclusion

A new modular multi-user multi-view autostereoscopic display architecture based on an array of MEMS projectors and a vertical diffuser is presented with a demonstrator. Unlike similar type of displays, the screen assembly consists of only one vertical diffuser. Our proposal has the capability to provide horizontal expansion in the screen size, and an increase in the number of different perspectives as well. The proposal is a good candidate for huge screen applications.

The in-house built demonstrator with an array of reverse engineered 18 projectors with 9 mm projector center-to-center spacing displays an image with a resolution of 234×848, and 37 different perspectives. The size of the image at the screen is $16\ cm \times 72\ cm$. The control hardware of the demonstrator is fairly simple and cost efficient.

Chapter 4

CONCLUSION

The 3D displays promise an immersive experience and provide more information to the end-users when compared with the conventional 2D displays. One of our main goals was to develop new types of 3D displays with enhanced capabilities. We believe that enhanced capabilities combined with easily adaptable techniques can help renew the interest of consumers and consumer electronics companies in 3D displays. We mainly focused on developing new applications using MEMS scanner based pico projector devices, which proved to be an outstanding candidate technology for 3D.

The overall research presented in this work has introduced four different 3D display methods using off-the-shelf laser scanning pico projectors. Two of the new methods require eyeglasses and the other two are glasses-free. The technical contribution and the impact of each method and the thesis overall are detailed below.

Mixed Polarization Technique: The first stereoscopy method entitled as Mixed Polarization proposes a novel glasses-type stereoscopic pico projector based projection display with an active polarization rotator. The proposal had came to life with a prototype, which operates on source 3D content with a display refresh rate value of 60 Hz without any noticeable flicker effect. It was also shown that with a little creativity, one can find suitable surfaces for polarization-based 3D display in everyday environments. The existing content is usable with the prototype through an online simple conversion algorithm. The prototype was benchmarked in terms of average cross talk level, and found out that crosstalk was 4.3% for the left eye and 7.5% for the right eye. Viewers generally agreed that the current level of crosstalk did not degrade the 3D experience. Before this work, displaying 3D contents without flicker at 60 Hz was not possible. As long as the display device is suitable to use this technique (having

initially polarized light sources), 3D displays with glasses do not require high refresh rates or complex hardware. Existing suitable display can have 3D capability through our technique. We believe also this new approach opens a reliable path for displaying and sharing 3D content with mobile devices. Glasses based systems offer the best cost-effective solution in the 3D display domain; with our contribution costs to realize a mobile 3D display with glasses went down dramatically. The technique requires some hardware modifications to existing pico-projectors. The targeted end-user space is basically the wide spectrum of all mobile users. We believe such an enhancement can be coupled to smart phones through mobile projectors and it can provide more opportunities in terms of social sharing scenarios.

Super Stereoscopic 3D Technique: The second stereoscopy method entitled as Super Stereoscopy Technique proposes an enhancement for the capabilities of stereoscopic displays and autostereoscopic displays. The method provides solution to a long known common problem in 3D displays: accommodation-vergence conflict. The method was demonstrated using an in-house built 3D display introduced in Section 3.1 as the test bed. Our implementation provides two different perspectives to each eye of a user through pinholes equipped with color filters. The pinholes provide images that are sharp regardless of accommodation distance of the eyes. As a result, accommodation of the eyes is driven by binocular disparity and convergence, but not by image blur, leading to the correction of accommodation response and elimination of A/C conflict. Providing two parallax images per eye provides a more natural blurring effect for defocused objects. Through subjective tests, the method was found to enhance the depth perception, and avoids accommodation-vergence conflict (allowing more life-like images). For a virtual object as close as 50 cm, subjects who could perceive 3D increased from 13 % to 100 % with the help of our two pinhole glasses. Major drawback of the method is a decrease in perceived light intensity, which can be solved using multiple pinholes with multiple color filters or polarizers. The overall system provides a quite promising modification to existing technologies. Before our work, 3D displays with glasses did not have the capability to provide 3D images at

close distances. With a small amount of hardware modification, depth perception is enhanced, and the technique is highly adaptable to existing hardware at low cost. People requiring to use a 3D display with glasses for design and engineering purposes are generally viewing their displays at distances such as $50 - 100\ mm$. This makes those people a good target in terms of end-user space. People with similar requirements can finally enjoy enhanced depth information without requiring a SMV display. The work presented is highly replicable, even end-users can replicate the work to enhance existing hardware at hand.

Rotating 3D display: The third method proposes a new type of multi-view autostereoscopic projection display, using a pair pico projectors, a head-tracking camera, and a rotating retro-reflective diffuser screen. The method was demonstrated with a prototype. Analytical and experimental results for crosstalk and luminance are in good agreement. Two different diffuser types were used for the screen: one periodic and one aperiodic. The screen with periodic diffuser achieved a viewing field of 500 mm in horizontal axis, and 450 mm in vertical axis at the distance of 1 m between the user and the screen. The screen with aperiodic diffuser achieved a viewing field of 700 mm in horizontal axis, and 500 mm in vertical axis. The crosstalk of the display is below 5 % for both diffusers. The maximum luminance of the display is 50 cd/m^2 and 160 cd/m^2 for periodic diffuser and aperiodic diffuser, respectively. The crosstalk remained below 5 % within $\pm 7.5\ cm$ in the z-axis, i.e., user can move back and forth about 15 cm and the crosstalk remain at acceptable levels while the luminance dropped by 25 % and 10 % for screen with periodic diffuser and screen with aperiodic diffuser. The transition between different perspectives is smooth as viewing slits track the eyes real-time. The overall system provides an extremely cheap solution using off-the-shelf equipments, and it is high feasible for single user scenarios in daily life such as computer screens, design tables, personal simulators and privacy screens. Previously, there wasn't any glasses free display dedicated to a single moving viewer in front of the screen. Requirement of complex hardware were avoided. The work presented is highly replicable, even end-users can replicate the work to enhance

existing hardware at hand.

Multiview 3D display: The last method proposes a new modular multi-user multi-view autostereoscopic display architecture based on an array of MEMS projectors and a vertical diffuser. A demonstrator is built to show the capabilities of the proposal. Unlike similar type of displays, the screen assembly is consisting only a vertical diffuser. Our proposal has the capability to provide horizontal expansion in the screen size, and an increase in the number of different perspectives as well. The in-house built demonstrator with an array of reverse engineered 18 projectors with 9 mm projector center-to-center spacing displays an image with a resolution of 234×848, and 37 different perspectives. The size of the image at the screen is $16\ cm \times 72\ cm$. The control hardware of the demonstrator is fairly simplistic and cost efficient. The proposal is a good candidate for huge screen applications. Before our implementation, such displays were bulky in volume, we have decreased the volumes dramatically to a level where users can carry around. Our main target in this development is the manufacturers, we believe we have presented a good example of a relatively simple and scalable display architecture. The work presented requires engineering capabilities in terms of replication, it can be only replicated by the manufacturers.

Head-worn projection display: Additionally, as an exemplary display application, we have demonstrated a light-weight, low-cost, mobile and ergonomic head-worn projection display using off-the-shelf equipment. The used equipment was modified to create a compact and longer lasting prototype ($3 - 4$ hours). Our prototype uses a single 15 lm pico projector, and does not require a cable connection to a stationary unit. Perceived images by the users are focus-free, bright and distortion-free. A perceived field-of-view of a viewer is 50^o in diagonal. The vision of the users is not occluded and no hardware is placed in the users field of vision. We have shown that exploring a virtual environment is possible with our prototype, and allows to create a fairly immersive experience. Multiple users can independently use the screen without experiencing crosstalk. Initial functionality tests have been successfully performed. Furthermore, the feeling of nausea originated from long-time usage was not observed

through our initial testing. The prototype can also be used as a 3D stereo system using the same hardware by additionally mounting polarized glasses and active polarization rotator, while maintaining all of the advantages listed above. The primary utility in publishing will be to enable an inexpensive, lightweight and stand-alone AR HMD for use in research and for motion capture studios. The work presented is highly replicable, even end-users can replicate the work to enhance existing hardware at hand.

The four new methods and the HMPD application demonstrated that laser scanning MEMS projection is a very promising enabling technology for 3D. The small volume of the projectors enables miniaturization of the display, and provides true mobility with flexibility in terms of design. We believe there are different solutions to different end-user spaces, and each method introduced provides different opportunities to different end-user spaces. In terms of replication, our methods are fairly simplistic when compared with their alternatives. Some of these techniques are even replicable with the existing equipment at hand. Our research contributed to multiple domains and created a valuable piece of information to provide cost-effective 3D displays using off-the-shelf devices. We see our implementations as building blocks for next generation 3D displays. We believe a motivation boost on pursuing the ultimate goal of presenting life-like images through displays was given through our research both for industrial and academic communities.

Chapter 5

APPENDIX

5.1 APPENDIX A: Conversion of a standard content for Mixed polarization method

Content conversion routine:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Programmed by Kaan Akşit

#avconv -i sample.mp4 -s 480x848 sample.avi

import cv,sys,cv2
from numpy import *

# Hazır bir kaynaktan görüntü alıp işlemek için ilgili sınıf
class vidyo:
    def __init__(self,hedef):
        self.kaynak     = cv.CaptureFromFile(hedef)
        self.yukseklik = cv.GetCaptureProperty(self.kaynak,
           cv.CV_CAP_PROP_FRAME_HEIGHT)
        self.genislik   = cv.GetCaptureProperty(self.kaynak,
           cv.CV_CAP_PROP_FRAME_WIDTH)
        self.kareno     = cv.GetCaptureProperty(self.kaynak,
           cv.CV_CAP_PROP_FPS)
```

```
        self.cc          = cv.GetCaptureProperty(self.kaynak,
            cv.CV_CAP_PROP_FOURCC)
        return
    def yakala(self):
        self.resim    = cv.QueryFrame(self.kaynak)
        if self.resim == None:
            sys.exit()
        cv.CvtColor(self.resim, self.resim, cv.CV_BGR2RGB)
        return self.resim

def main():
    cv.NamedWindow('Görsel',cv.CV_WINDOW_AUTOSIZE)
    vidyo1     = vidyo('./sample.avi')
    eskikopya = vidyo1.yakala()
    r1         = cv.CreateImage((eskikopya.width/2,eskikopya.
        height), 8, 1)
    b1         = cv.CreateImage((eskikopya.width/2,eskikopya.
        height), 8, 1)
    g1         = cv.CreateImage((eskikopya.width/2,eskikopya.
        height), 8, 1)
    r2         = cv.CreateImage((eskikopya.width/2,eskikopya.
        height), 8, 1)
    b2         = cv.CreateImage((eskikopya.width/2,eskikopya.
        height), 8, 1)
    g2         = cv.CreateImage((eskikopya.width/2,eskikopya.
        height), 8, 1)
    dosya1     = cv2.VideoWriter('./cikti.avi',int(vidyo1.cc)
        ,60,(cv.GetSize(r1)[0],cv.GetSize(r1)[1]),1)
    while True:
```

```
        resim        = vidyo1.yakala()
        ikiresim1    = cv.CloneImage(resim)
        ikiresim2    = cv.CloneImage(resim)
        ilkresim     = ikiresim1[0:ikiresim1.height, 0:
            ikiresim1.width/2]
        ikinciresim = ikiresim2[0:ikiresim2.height, ikiresim2.
            width/2:ikiresim2.width]
        cv.Split(ilkresim,r1,g1,b1,None)
        cv.Split(ikinciresim,r2,g2,b2,None)
        cv.Merge(r1,g2,b1,None,ilkresim)
        cv.Merge(r2,g1,b2,None,ikinciresim)
        for i in xrange(0,19):
            for j in xrange(0,19):
                ilkresim[j,i]     = (255,255,255)
                ikinciresim[j,i] = (0,0,0)
        dosya1.write(asarray(ilkresim))
        cv.ShowImage('Görsel', ilkresim)
        cv.WaitKey(10)
        dosya1.write(asarray(ikinciresim))
        cv.ShowImage('Görsel', ikinciresim)
        cv.WaitKey(10)
    return True

if __name__ == '__main__':
    sys.exit(main())
```

5.2 Appendix B: Volumetric pixel size simulation using optical ray tracing

Voxel size detection routine:

```
 #!/usr/bin/python
# -*- coding: utf-8 -*-

import sys,odak,math,csv
import matplotlib.pyplot as plt
from math import cos, sin

__author__ = ('Kaan Akşit')
# This script solves the question how stereoscopy is improved
    with two pinholes infront of an eye.

# Main function where variables are set, and the solution
   method is called.
def main():
    # REMEMBER TO ALWAYS ASSIGN FLOATING NUMBERS TO THE
       VARIABLES!
    # Distance between pinholes and the point sources (mm).
    ds          = 1000.0
    # Distance between center to center in between pinholes (
       mm).
    dhp         = 2.
    # Radius of the pinholes (mm).
    r           = 0.4
    # Distance between the point sources (mm).
    dpp         = 4.0
    # Half of the aperture of the eye (mm), and also
       determines equivalent ball lens focal.
    dea         = 9.0
```

```
    # Diameter of the whole eye, according to http://www.
        opticampus.com/files/introduction_to_ophthalmic_optics
        .pdf ,which employs Gullstrand-Emsley model.
    dwe         = 12.0
    # Half of the thickness of the eye lens (mm).
    tel         = 0.2
    # Refractive index of the cristal eye lens (ball lens in
        our case).
    nel         = 1.41
    # Refractive index of inside the lens.
    nie         = 1.33
    # Refractive index of the outside envorinment.
    nair        = 1.
    # Distance between the pinholes and the lens (mm).
    dpl         = -10.
    # X and Y positions of the lens.
    xel         = 0
    yel         = 0
    # Detector position in 3D space determined by three
        points from the surface.
    DetPos      = [
                  (-30,30,-50),
                  (30,-30,-50),
                  (30,30,-50)
                  ]
    # Solve the problem according to the given values.
#     Solve(ds,dhp,r,dpp,dea,dwe,tel,nel,nie,nair,dpl,xel,yel,
    DetPos,True)
    # Open a csv file to store the output of the simulation.
```

```
filename = 'results.csv'
out      = csv.writer(open(filename,"w"), delimiter=',',
   quoting=csv.QUOTE_ALL)
# Iterative solution for plotting dependency on aperture
   size, separation between point sources, and separation
    between two pinholes.
VoxelHeights = []
VoxelWidths  = []
X            = []
Y            = []
Maxr         = 0.6
Maxdpp       = 5.0
step         = 0.1
# Iterate the aperture size.
for r in xrange(1,int(Maxr/step),1):
    r *= step
    # Iterate the seperation of point sources.
    for dpp in xrange(1,int(Maxdpp/step),1):
        dpp *= step
        # Solve the given case.
        VoxelHeight, VoxelWidth = Solve(ds,dhp,r,dpp,dea,
            dwe,tel,nel,nie,nair,dpl,xel,yel,DetPos,False)
        # Voxel dimensions stored in a list.
        VoxelHeights.append(VoxelHeight)
        VoxelWidths.append(VoxelWidth)
        X.append(r*2)
        Y.append(dpp)
        # Values stored in a list.
        Values = (r,dpp,VoxelHeight,VoxelWidth)
```

```
            # Show current the latest solution.
            print Values
            # Write the solution to the file.
            out.writerow(Values)

    # Font size in the plots.
    FontSize =  22
    # Font weight in the plots.
    FontWeight = 'normal'
    # Font type in the plots.
    FontType = 'sans'

    # Call ray tracing library to plot the data as a 3D
        surface.
#     Fig = odak.raytracing()
    # Setting font properties in the figure.
#     Fig.SetPlotFontSize(FontType,FontWeight,FontSize)


    # Call ray tracing library to plot the data as a 3D
        surface.
#     Fig1 = odak.raytracing()
    # Setting font properties in the figure.
#     Fig1.SetPlotFontSize(FontType,FontWeight,FontSize)
    # Plot the data as 3D surface.
#     Fig1.PlotData(X,Y,VoxelHeights,'g')
    # Show the plotted data.
#     Fig1.showplot('$h$ (mm)','$d_p$ (mm)', '$d_c$ (mm)', '
    VoxelHeightRay.png')
```

```
    # Call ray tracing library to plot the data as a 3D
        surface.
#     Fig2 = odak.raytracing()
    # Setting font properties in the figure.
#     Fig2.SetPlotFontSize(FontType,FontWeight,FontSize)
    # Plot the data as 3D surface.
#     Fig2.PlotData(X,Y,VoxelWidths,'g')
    # Show the plotted data.
#     Fig2.showplot('$w$ (mm)','$d_p$ (mm)', '$d_c$ (mm)', '
    VoxelWidthRay.png')

    # Contour presentation of the data.
    # Necessary imports for the contour plot.
    import matplotlib.pyplot as plt
    import matplotlib
    from matplotlib import cm
    from scipy.interpolate import Rbf
    from numpy import *

    # Definition to set the font type, size and weight in
        plots.
    font = {'family' : FontType,
            'weight' : FontWeight,
            'size'   : FontSize}
    matplotlib.rc('font', **font)
    # Enables Latex support in the texts.
    matplotlib.rc('text', usetex=True)
```

```
    # Voxel height and voxel width figure created.
    for Z in [VoxelWidths,VoxelHeights]:
        rbf    = Rbf(X, Y, Z, epsilon=2)
        t1     = linspace(amin(X), amax(X), 300)
        t2     = linspace(amin(Y), amax(Y), 300)
        XI, YI = meshgrid(t1, t2)
        ZI     = rbf(XI, YI)

        # Plot the 2D surface shape.
        FigContour1 = plt.figure(figsize=(15,9),dpi=300)
        ax1         = FigContour1.gca()
        plt.pcolor(XI, YI, ZI, cmap=cm.jet)

        # Add colorbars and the labels to the figure.
        cb = plt.colorbar(orientation='vertical')
        cl = plt.getp(cb.ax, 'ymajorticklabels')
        plt.setp(cl, fontsize=20)
#         cb.ax.set_ylabel(etiket,fontsize=20)

        # Regions are labeled with the contours.
        levels = linspace(amin(ZI), amax(ZI), 10)
        CS     = plt.contour(XI, YI, ZI, levels, linewidths
           =5, colors='k', linestyle='-')
        plt.clabel(CS, inline=1, fontsize=FontSize)
        if Z == VoxelWidths:
            name = 'VoxelWidthRay'
            plt.title('$w_m (mm)$',fontsize=FontSize)
        else:
```

```
            name = 'VoxelHeightRay'
            plt.title('$h_m (mm)$',fontsize=FontSize)
        plt.xlabel('$d_p (mm)$',fontsize=FontSize)
        plt.ylabel('$d_c (mm)$',fontsize=FontSize)
        plt.savefig('%s.png' % name)

    return True

def Solve(ds,dhp,r,dpp,dea,dwe,tel,nel,nie,nair,dpl,xel,yel,
    DetPos,ShowPlot=False):
    # Z position of the lens is calculated.
    zel         = dpl-tel
    # Define the center of the first circular pinhole in 3D
       space.
    HoleCenter1 = (-dhp/2,0,0)
    # Define the radius of the first circular pinhole in mm.
    Radius1     = r
    # Define the center of the second circular pinhole in 3D
       space.
    HoleCenter2 = (dhp/2,0,0)
    # Define the radius of the second circular pinhole in mm.
    Radius2     = r
    # Call the library to trace.
    ray         = odak.raytracing()
    # Position of the first point source.
    Point1      = (dpp/2,0,ds)
    # Position of the second point source.
    Point2      = (-dpp/2,0,ds)
```

```
    # First section of the solution for first point source
       and first pinhole.
    # Exterior points on the pinholes.
    PointList1 = [HoleCenter1]
    PointList2 = [HoleCenter2]
    SourceList = [Point1,
                  Point2
                 ]
    # Add additional points on the exterior of the pinholes.
    for k in xrange(0,360,60):
        # Converting degrees to radians.
        k = ray.DegreesToRadians(k)
        # Appending new item to the point lists.
        PointList1.append((HoleCenter1[0]+Radius1*cos(k),
            HoleCenter1[1]+Radius1*sin(k),HoleCenter1[2]))
        PointList2.append((HoleCenter2[0]+Radius2*cos(k),
            HoleCenter2[1]+Radius2*sin(k),HoleCenter2[2]))
    # Plot a spherical lens.
#     AsphericalLens = ray.plotasphericallens(xel,yel,zel,dea,
    dea,tel,nel,'b')
    # Create the whole eye ball.
    EyeBall  = (xel,yel,zel-dwe,dwe)
    # Plot the eye ball.
    SphericalLens = ray.plotsphericallens(EyeBall[0],EyeBall
        [1],EyeBall[2],EyeBall[3],'y',0.1)
    # Create a dummy spherical lens to find the intersection
       point of the incoming ray with aspherical lens.
    DummySL1 = (xel,yel,zel-dea,dea)
    # Plot a spherical lens.
```

```
    SphericalLens = ray.plotsphericallens(DummySL1[0],
        DummySL1[1],DummySL1[2],DummySL1[3],'g',0.1)
    # Calculate and print power of the spherical lens.
    D = (nel-nair)*(pow(dea,-1)-pow(-dea,-1)+(nel-nair)*2*dea
        /nel/(-pow(dea,2)))
#     print 'Power of the spherical lens: ', 1000.*D
#     print 'Focal length of the spherical lens (mm): ', 1./D
    # Calculate the effective focal length of a ball lens.
    # See http://www.edmundoptics.com/technical-resources-
        center/optics/understanding-ball-lenses/
    EFL = nel*2*dea/4/(nel-nair)
#     print 'Effective focal length of the ball lens (mm):',
    EFL
    # Arrays to save rays by their source of origin.
    rays1     = []
    rays2     = []
    # Plot the pinholes.
    ray.PlotCircle(HoleCenter1,Radius1)
    ray.PlotCircle(HoleCenter2,Radius2)
    # Iterate ray tracing for every point source.
    for origin in SourceList:
        # Make destination choice according to the source,
            color the rays accordingly.
        if origin == Point1:
            DestinationList = PointList1
            RayColor        = 'r'
        elif origin == Point2:
            DestinationList = PointList2
            RayColor        = 'b'
```

```
# Iterate ray tracing for every destination.
for item in DestinationList:
    # Finding the angle in between the origin and the
        destination point.
    angles = ray.findangles(origin,item)
    # Creating the new vector in the direction of
        destination using the previous angles
        calculated.
    NewRay = ray.createvector(origin,angles)
    # Chiefray color selection.
    if item == DestinationList[0]:
        color = RayColor + '+-'
    else:
        color = RayColor + '+'
    # Intersect the vector with dummy sphere.
    distance,normvec = ray.findinterspher(NewRay,
        DummySL1)
    # Plot the ray from origin to the destination.
    if distance != 0:
        ray.plotvector(NewRay,distance,color)
        # Storing ray to the rays vector by their
            source of origin.
        if origin == Point1:
            rays1.append(NewRay)
        else:
            rays2.append(NewRay)
        # Refracting into the eye.
        RefractRay = ray.snell(NewRay,normvec,nair,
            nel)
```

```
                    # The ray travels inside the eye ball.
                    distance,normvec = ray.findinterspher(
                        RefractRay,DummySL1)
                    # Plot the refracting ray.
                    ray.plotvector(RefractRay,distance,color)
                    # Refracting to outside of the eye.
                    RefractOutsideRay = ray.snell(RefractRay,
                        normvec,nel,nie)
                    # Shine rays to the retina.
                    distance,normvec = ray.findinterspher(
                        RefractOutsideRay,EyeBall)
                    # Find the intersection of the refracted ray
                        with the detector surface.
#                   distance,normvec = ray.findintersurface(
    RefractOutsideRay,(DetPos[0],DetPos[1],DetPos[2]))
#                    # Plot the refracting ray.
                    ray.plotvector(RefractOutsideRay,distance,
                        color)
    # Loop to find intersection of the plotted rays.
    InterPoints = []
    for RaySource1 in rays1:
        for RaySource2 in rays2:
            # Find the intersection of two rays.
            intersection, distances = ray.
                CalculateIntersectionOfTwoVectors(RaySource1,
                RaySource2)
            # Check if the calculated distance value has a
                logical value.
            CheckValue = "%.2f" % abs(distances[0])
```

```
                if CheckValue != '0.00' and float(CheckValue) <
                    100000:
                    ray.PlotPoint(intersection,'go',False,True)
                    # Storing intersection point in a list.
                    InterPoints.append(intersection)
        # Transpose of the InterPoints 2D list.
        l  = map(list,map(None,*InterPoints))
        # Finding Voxel height.
        m1           = max(l[2])
        m2           = min(l[2])
        VoxelHeight = abs(m1-m2)
        # Finding Voxel width.
        m3           = max(l[0])
        m4           = min(l[0])
        VoxelWidth = abs(m3-m4)
        # Show the ray tracing envorinment in three-dimensional
            space.
        if ShowPlot == True:
            limit = 0.8*ds
            ray.defineplotshape((-limit,limit),(-limit,limit),(-
                limit,limit))
            ray.showplot()
        else:
            # Otherwise destroy figure.
            ray.CloseFigure()
        return VoxelHeight,VoxelWidth

if __name__ == '__main__':
    sys.exit(main())
```

Ray tracing, Fraunhofer and Fresnel diffraction, Jones Calculus library:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Programmed by Kaan Akşit

import sys,matplotlib,scipy
import matplotlib.pyplot
import mpl_toolkits.mplot3d.art3d as art3d
import scipy.linalg
from matplotlib.mlab import griddata
from matplotlib import cm
from matplotlib.patches import Circle, PathPatch
from mpl_toolkits.mplot3d import axes3d
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from numpy import *
from numpy.fft import *

__author__ = ('Kaan Akşit')

class paraxialmatrix():
    def __init__(self):
        # See "Laser beams and resonators" from Kogelnik and
            Li for the theoratical explanation
        self.plt = matplotlib.pyplot
        self.fig = self.plt.figure()
        return
    def createvector(self,x,angle):
        # Creates a paraxial ray, angle is in degrees, x is
            the distance of the point to the plane of
```

```
            direction of propagation
        angle  = radians(angle)
        vector = array([[x],[angle],[1]])
        return vector
    def freespace(self,vector,distance,deltax=0,deltafi=0):
        # Ray transfer matrix of free space propagation,
            distance is in meters
        # deltax is the spatial shift, deltafi is the angular
             shift
        space  = array([[1,distance,deltax],[0,1,deltafi
            ],[0,0,1]])
        vector = dot(space,vector)
        return vector
    def plotvector(self,startvector,stopvector):
        # Method to plot paraxial vectors in 2D space
        self.plt.plot([startvector[0]/startvector[1],
            stopvector[0]/stopvector[1]],[startvector[0],
            stopvector[0]],'o-')
        self.plt.show()
        return True

class raytracing():
    def __init__(self):
        # See "General Ray tracing procedure" from G.H.
            Spencerand M.V.R.K Murty for the theoratical
            explanation
        self.plt = matplotlib.pyplot
        # New figure created.
        self.fig = self.plt.figure(figsize=(17, 13))
```

```
        # 3D projection is enabled.
        self.ax  = self.fig.gca(projection='3d')
        # Enabling the grid in the figure.
        self.ax.grid(True, color='k', linewidth=2)
        return
    def SetPlotFontSize(self,family='normal',weight='normal',
        size='22'):
        # Definition to set the font type, size and weight in
            plots.
        font = {'family' : family,
                'weight' : weight,
                'size'   : size}
        matplotlib.rc('font', **font)
        # Enables Latex support in the texts.
        matplotlib.rc('text', usetex=True)
        # Set the ticks font propoerties as well to be on the
            safe side.
        self.ax.xaxis.label.set_fontsize(size)
        self.ax.yaxis.label.set_fontsize(size)
        self.ax.zaxis.label.set_fontsize(size)
        self.ax.xaxis.label.set_family(family)
        self.ax.yaxis.label.set_family(family)
        self.ax.zaxis.label.set_family(family)
        self.ax.xaxis.label.set_fontweight(weight)
        self.ax.yaxis.label.set_fontweight(weight)
        self.ax.zaxis.label.set_fontweight(weight)
        return True
    def DegreesToRadians(self,angle):
        # Function to convert degrees to radians.
```

```
        return radians(angle)
    def findangles(self,Point1,Point2):
        # Function to find angle between two points if there
            was a line intersecting at both of them.
        # Vector to hold angles is created.
        angles    = []
        # Find the distance in between points.
        distance = self.finddistancebetweentwopoints(Point1,
            Point2)
        # X axis rotation is calculated.
        angles.append(degrees(arccos( (Point2[0]-Point1[0])/
            distance )))
        # Y axis rotation is calculated.
        angles.append(degrees(arccos( (Point2[1]-Point1[1])/
            distance )))
        # Z axis rotation is calculated.
        angles.append(degrees(arccos( (Point2[2]-Point1[2])/
            distance )))
        # Angles are returned.
        return angles
    def finddistancebetweentwopoints(self,Point1,Point2):
        # Function to find the distance between two points if
            there was a line intersecting at both of them.
        distancex = Point1[0]-Point2[0]
        distancey = Point1[1]-Point2[1]
        distancez = Point1[2]-Point2[2]
        distance  = sqrt(pow(distancex,2)+pow(distancey,2)+
            pow(distancez,2))
        return distance
```

```
def createvector(self,(x0,y0,z0),(alpha,beta,gamma)):
    # Create a vector with the given points and angles in
        each direction
    point = array([[x0],[y0],[z0]])
    alpha = cos(radians(alpha))
    beta  = cos(radians(beta))
    gamma = cos(radians(gamma))
    # Cosines vector
    cosin = array([[alpha],[beta],[gamma]])
    return array([point,cosin])
def CalculateIntersectionOfTwoVectors(self,vector1,
   vector2):
    # Method to calculate the intersection of two vectors
        .
    A = array([
                [vector1[1][0][0], vector2[1][0][0] ],
                [vector1[1][1][0], vector2[1][1][0] ],
                [vector1[1][2][0], vector2[1][2][0] ]
               ])
    B = array(
               [vector1[0][0]-vector2[0][0],
                vector1[0][1]-vector2[0][1],
                vector1[0][2]-vector2[0][2]
               ])
    # LU decomposition solution.
    distances = scipy.linalg.solve(A[:][0:2], B[:][0:2])
    # Check if the given solution matches the initial
        condition at the third equation.
```

```
        if int(abs(dot(A[:][2],distances))) != int(abs(B
            [:][2])):
            distances[0] = 0
            distances[1] = 0
        # Point vector created.
        Point      = []
        # Intersection point at X axis.
        Point.append((vector1[0][0][0] - distances[0]*vector1
            [1][0][0])[0])
        # Intersection point at Y axis.
        Point.append((vector1[0][1][0] - distances[0]*vector1
            [1][1][0])[0])
        # Intersection point at Z axis.
        Point.append((vector1[0][2][0] - distances[0]*vector1
            [1][2][0])[0])
        return Point,distances
    def createvectorfromtwopoints(self,(x0,y0,z0),(x1,y1,z1))
        :
        # Create a vector from two given points
        point = array([[x0],[y0],[z0]])
        # Distance between two points
        s     = sqrt( pow((x0-x1),2) + pow((y0-y1),2) + pow((
            z0-z1),2) )
        alpha = (x1-x0)/s
        beta  = (y1-y0)/s
        gamma = (z1-z0)/s
        # Cosines vector
        cosin = array([[alpha],[beta],[gamma]])
        return array([point,cosin])
```

```
    def multiplytwovectors(self,vector1,vector2):
        # Multiply two vectors and return the resultant
            vector
        # Used method described under:
        # Cross-product: http://en.wikipedia.org/wiki/
            Cross_product
        angle = cross(vector1[1].transpose()[0],vector2[1].
            transpose()[0])
        return array([vector1[0],[[angle[0]],[angle[1]],[
            angle[2]]]])
    def anglebetweentwovector(self,vector0,vector1):
        # Finds angle between two vectors
        # Used method described under: http://www.wikihow.com
            /Find-the-Angle-Between-Two-Vectors
        angle = vector0[1]*vector1[1]
        angle = angle[0]+angle[1]+angle[2]
        s1    = sqrt(vector0[1][0]**2+vector0[1][1]**2+
            vector0[1][2]**2)
        s2    = sqrt(vector1[1][0]**2+vector1[1][1]**2+
            vector1[1][2]**2)
        angle = degrees(arccos(angle/(s1*s2)))
        return angle
    def isitontriangle(self,pointtocheck,point0,point1,point2
        ,error=0.1):
        # Check if the given point is insight the triangle
            which represented
        # by three corners of the triangle.
        # Used method described under: http://www.blackpawn.
            com/texts/pointinpoly/default.html
```

```
        # point0, point1 and point2 are the corners of the
            triangle
        # point is the point to check
        vector1 = self.createvectorfromtwopoints(pointtocheck
            ,point0)
        vector2 = self.createvectorfromtwopoints(pointtocheck
            ,point1)
        vector3 = self.createvectorfromtwopoints(pointtocheck
            ,point2)
        angle0  = self.anglebetweentwovector(vector1,vector2)
        angle1  = self.anglebetweentwovector(vector2,vector3)
        angle2  = self.anglebetweentwovector(vector3,vector1)
        sum     = angle0+angle1+angle2
        if sum <= 360+error and sum >= 360-error:
            return True
        else:
            return False
    def transform(self,input,(alpha,beta,gamma),(x0,y0,z0)):
        # alpha; rotation angle (euler) of x axis
        # beta; rotation angle (euler) of y axis
        # gamma; rotation angle (euler) of z axis
        # x0; x coordinate of origin measured in the
            reference system
        # y0; y coordinate of origin measured in the
            reference system
        # z0; z coordinate of origin measured in the
            reference system
        alpha  = radians(alpha)
        beta   = radians(beta)
```

```
        gamma  = radians(gamma)
        R1     = array([[cos(gamma),-sin(gamma) ,0],[sin(gamma
            ),cos(gamma) ,0],[0 ,0 ,1]])
        R2     = array([[1 ,0 ,0],[0 ,cos(beta),-sin(beta)],[0 ,
            sin(beta),cos(beta)]])
        R3     = array([[cos(alpha),0,-sin(alpha)],[0 ,1 ,0],[
            sin(alpha),0,cos(alpha)]])
        R      = dot(dot(R1,R2),R3)
        output = dot(R,input-array([[x0],[y0],[z0]]))
        return output
    def reflect(self,vector,normvector):
        # Used method described in G.H. Spencer and M.V.R.K.
            Murty, "General Ray-Tracing Procedure", 1961
        mu = 1
        div = pow(normvector[1,0],2) + pow(normvector[1,1],2)
             + pow(normvector[1,2],2)
        a = mu* (vector[1,0]*normvector[1,0] + vector[1,1]*
            normvector[1,1] + vector[1,2]*normvector[1,2]) /
            div
        VectorOutput       = vector.copy()
        VectorOutput[0,0] = normvector[0,0]
        VectorOutput[0,1] = normvector[0,1]
        VectorOutput[0,2] = normvector[0,2]
        VectorOutput[1,0] = vector[1,0] - 2*a*normvector[1,0]
        VectorOutput[1,1] = vector[1,1] - 2*a*normvector[1,1]
        VectorOutput[1,2] = vector[1,2] - 2*a*normvector[1,2]
        return VectorOutput
    def snell(self,vector,normvector,n1,n2,error=0.01):
        # Method for Snell's law
```

```
# n1 refractive index of the medium which vector is
    coming from
# n2 refractive index of the medium which vector
    tries to go into
mu      = n1/n2
div     = pow(normvector[1,0],2)  + pow(normvector
    [1,1],2) + pow(normvector[1,2],2)
a       = mu* (vector[1,0]*normvector[1,0] + vector
    [1,1]*normvector[1,1] + vector[1,2]*normvector
    [1,2]) / div
b       = (pow(mu,2)-1) / div
to      = -b*0.5/a
num     = 0
eps     = error*2
# Newton-Raphson method to find the correct root
while eps > error:
    num   += 1
    oldto  = to
    v      = pow(to,2) + 2*a*to + b
    deltav = 2*(to+a)
    to     = to - v /deltav
    eps    = abs(oldto-to)
    # Iteration notifier
    #print 'Iteration number: %s, Error: %s' % (num,
        error)
    # Iteration limiter
    if num > 5000:
        return vector
VectorOutput        = vector.copy()
```

```
        VectorOutput[0,0] = normvector[0,0]
        VectorOutput[0,1] = normvector[0,1]
        VectorOutput[0,2] = normvector[0,2]
        VectorOutput[1,0] = mu*vector[1,0] + to*normvector
            [1,0]
        VectorOutput[1,1] = mu*vector[1,1] + to*normvector
            [1,1]
        VectorOutput[1,2] = mu*vector[1,2] + to*normvector
            [1,2]
        return VectorOutput
    def findinterspher(self,vector,sphere,error=0.00000001,
        numiter=1000,iternotify='no'):
        # Method for finding intersection in between a vector
            and a spherical surface
        # There are things to be done to fix wrong root
            convergence
        number   = 0
        distance = 1
        olddist  = 0
        shift    = 0
        epsilon  = error*2
        k        = vector[0,0,0]
        l        = vector[0,1,0]
        m        = vector[0,2,0]
        FXYZ     = pow(k-sphere[0],2) + pow(l-sphere[1],2) +
            pow(m-sphere[2],2) - pow(sphere[3],2)
        if abs(FXYZ) < 0.01:
            shift = 1.5 * sphere[3]
            k     = shift * vector[1,0] + k
```

```
        l       = shift * vector[1,1] + l
        m       = shift * vector[1,2] + m
    while epsilon > error:
        number  += 1
        x       = olddist * vector[1,0] + k
        y       = olddist * vector[1,1] + l
        z       = olddist * vector[1,2] + m
        oldFXYZ = pow(x-sphere[0],2) + pow(y-sphere
            [1],2) + pow(z-sphere[2],2) - pow(sphere[3],2)
        x       = distance * vector[1,0] + k
        y       = distance * vector[1,1] + l
        z       = distance * vector[1,2] + m
        FXYZ    = pow(x-sphere[0],2) + pow(y-sphere
            [1],2) + pow(z-sphere[2],2) - pow(sphere[3],2)
        # Secant method is calculated, see wikipedia
            article of the method for more
        newdist = distance - FXYZ*(distance-olddist)/(
            FXYZ-oldFXYZ)
        epsilon = abs(newdist-distance)
        oldFXYZ = FXYZ
        olddist = distance
        distance = newdist
        normang = array([[(sphere[0]-x)/sphere[3]],[(
            sphere[1]-y)/sphere[3]],[(sphere[2]-z)/sphere
            [3]]])
        normpnt = array([x,y,z])
        normvec = array([normpnt,normang])
        # Iteration reminder
        if iternotify == 'yes':
```

```
                print 'Iteration number: %s, Calculated
                    distance: %s, Error: %s, Points: %s %s %s,
                     Function: %s' % (number,distance,epsilon
                    ,x,y,z,FXYZ)
            # Check if the number of iterations are too much
            if number > numiter:
                return 0,normvec
        return distance+shift,normvec
    def findintersurface(self,vector,(point0,point1,point2),
        error=0.00001,numiter=100,iternotify='no'):
        # Method to find intersection point inbetween a
            surface and a vector
        # See http://www.jtaylor1142001.net/calcjat/Solutions
            /VPlanes/VP3Pts.htm
        vector1  = self.createvectorfromtwopoints(point0,
            point1)
        vector2  = self.createvectorfromtwopoints(point0,
            point2)
        normvec  = self.multiplytwovectors(vector1,vector2)
        k        = vector[0,0,0]
        l        = vector[0,1,0]
        m        = vector[0,2,0]
        # See http://en.wikipedia.org/wiki/Normal_%28geometry
            %29
        a            = normvec[1][0]
        b            = normvec[1][1]
        c            = normvec[1][2]
        d            = -normvec[1][0]*normvec[0][0]-normvec
            [1][1]*normvec[0][1]-normvec[1][2]*normvec[0][2]
```

```
distance     = 1
number       = 0
olddistance = 0
epsilon      = error*2
while epsilon > error:
    number      += 1
    x1           = distance * vector[1,0] + k
    y1           = distance * vector[1,1] + l
    z1           = distance * vector[1,2] + m
    x2           = olddistance * vector[1,0] + k
    y2           = olddistance * vector[1,1] + l
    z2           = olddistance * vector[1,2] + m
    F1           = a*x1+b*y1+c*z1+d
    F2           = a*x2+b*y2+c*z2+d
    # Secant method: http://en.wikipedia.org/wiki/
      Secant_method
    newdistance = distance - F1*(distance-olddistance
      )/(F1-F2)
    epsilon      = abs(distance-olddistance)
    olddistance = distance
    distance     = newdistance
    normvec[0]   = array([x1,y1,z2])
    # Iteration reminder
    if iternotify == 'yes':
        print 'Iteration number: %s, Calculated
          distance: %s, Error: %s, F1: %s, F2: %s,
          Old distance: %s ' % (number,distance,
          error,F1,F2,olddistance)
    if number > numiter:
```

```
                return 0,normvec
        return olddistance, normvec
    def plotvector(self,vector,distance,color='g'):
        # Method to plot rays
        x = array([vector[0,0,0], distance * vector[1,0] +
            vector[0,0,0]])
        y = array([vector[0,1,0], distance * vector[1,1] +
            vector[0,1,0]])
        z = array([vector[0,2,0], distance * vector[1,2] +
            vector[0,2,0]])
        self.ax.plot(x,y,z,color)
        return True
    def PlotPoint(self,point,color='g*',contour=False,marker=
        False):
        # Method to plot a single spot.
        self.ax.plot(array([point[0]]),array([point[1]]),
            array([point[2]]),color)
        # Plotting contour on 3-axes.
        if contour == True:
            self.ax.contour(array([[-1,0,1],
                                   [-1,0,1],
                                   [-1,0,1]
                                   ]),
                            array([[-1,0,1],
                                   [-1,0,1],
                                   [-1,0,1]
                                   ]),
                            array([[2,4,2],
                                   [1,4,5],
```

```
                                        [3,3,3]
                                        ]),
                                zdir='z', offset=-100, cmap=cm.
                                    coolwarm)
        # Add text near to the point.
        if marker == True:
            label = '(%.1f, %.1f, %.1f)' % (point[0], point
                [1], point[2])
            self.ax.text(point[0], point[1], point[2], label)
        return True
    def plotsphericallens(self,cx=0,cy=0,cz=0,r=10,c='none',a
        =0.3):
        # Method to plot surfaces
        sampleno = 100
        v        = linspace(0, pi, sampleno)
        u        = linspace(0,2*pi,sampleno)
        x        = r * outer(cos(u), sin(v)) + cx
        y        = r * outer(sin(u), sin(v)) + cy
        z        = r * outer(ones(size(u)), cos(v)) + cz
        self.ax.plot_surface(x, y, z, rstride=8, cstride=8,
            alpha=a, color=c, antialiased=True)
        return array([cx,cy,cz,r])
    def CalculateFocal(self,rx,ry,rz,n,ShowFocal=False):
        # Method to calculate the focal length of the lens in
            different axes.
        for a in [rx,ry]:
            R          = (pow(a,2)+pow(rz,2))/(2*rz)
            LensMaker = (n-1)*(-2/R+(n-1)*rz*2/(n*R))
            f          = pow(LensMaker,-1)
```

```
            if ShowFocal == True:
                print 'Focal length of the lens: ',f
        return True
    def plotasphericallens(self,cx=0,cy=0,cz=0,rx=10,ry=10,rz
       =10,n=1.51,c='none'):
        # Method to plot surfaces
        sampleno = 50
        v        = linspace(0, pi, sampleno)
        u        = linspace(0,2*pi,sampleno)
        x        = rx * outer(cos(u), sin(v)) + cx
        y        = ry * outer(sin(u), sin(v)) + cy
        z        = rz * outer(ones(size(u)), cos(v)) + cz
        self.ax.plot_surface(x, y, z, rstride=6, cstride=6,
           color=c)
        # Calculate the focal length of the plotted lens.
        self.CalculateFocal(rx,ry,rz,n)
        return array([cx,cy,cz,rx,ry,rz])
    def PlotCircle(self,center,r,c='none'):
        # Method to plot circle.
        circle = Circle((center[0], center[1]), r, facecolor=
           c, edgecolor=(0,0,0), linewidth=4, alpha=1)
        self.ax.add_patch(circle)
        art3d.pathpatch_2d_to_3d(circle, z=center[2], zdir='z
           ')
        return array([center,r])
    def PlotData(self,X,Y,Z,c='none'):
        # Method to plot the given data.
        # Gridding the data.
        xi = linspace(min(X), max(X))
```

```
        yi = linspace(min(Y), max(Y))
        xim, yim = meshgrid(xi, yi)
        zi = griddata(X,Y,Z,xi,yi,interp='nn')
        # Plot the resultant figure.
        self.ax  = self.fig.gca()
        self.ax.plot_surface(xim, yim, zi, rstride=2, cstride
            =2, cmap=cm.jet, alpha=0.3, color=c)
        return True
    def plottriangle(self,point0,point1,point2):
        # Method to plot triangular surface
        x = array([ point0[0], point1[0], point2[0]])
        y = array([ point0[1], point1[1], point2[1]])
        z = array([ point0[2], point1[2], point2[2]])
        verts = [zip(x, y,z)]
        self.ax.add_collection3d(Poly3DCollection(verts))
        return array([point0,point1,point2])
    def plotcornercube(self,centerx,centery,centerz,pitch,
        revert=False):
        # Method to plot a single cornercube
        point00 = array([ centerx, centery, centerz])
        point10 = array([ centerx, centery, centerz])
        point20 = array([ centerx, centery, centerz])
        if revert == False:
            point01 = array([ centerx, centery-2*pitch/3,
                centerz+sqrt(2)*pitch/3 ])
            point11 = array([ centerx, centery-2*pitch/3,
                centerz+sqrt(2)*pitch/3 ])
            point21 = array([ centerx-pitch/sqrt(3), centery+
                pitch/3, centerz+sqrt(2)*pitch/3 ])
```

```
            point02 = array([ centerx-pitch/sqrt(3), centery+
                pitch/3, centerz+sqrt(2)*pitch/3 ])
            point12 = array([ centerx+pitch/sqrt(3), centery+
                pitch/3, centerz+sqrt(2)*pitch/3 ])
            point22 = array([ centerx+pitch/sqrt(3), centery+
                pitch/3, centerz+sqrt(2)*pitch/3 ])
        else:
            point01 = array([ centerx, centery+2*pitch/3,
                centerz+sqrt(2)*pitch/3 ])
            point11 = array([ centerx, centery+2*pitch/3,
                centerz+sqrt(2)*pitch/3 ])
            point21 = array([ centerx+pitch/sqrt(3), centery-
                pitch/3, centerz+sqrt(2)*pitch/3 ])
            point02 = array([ centerx+pitch/sqrt(3), centery-
                pitch/3, centerz+sqrt(2)*pitch/3 ])
            point12 = array([ centerx-pitch/sqrt(3), centery-
                pitch/3, centerz+sqrt(2)*pitch/3 ])
            point22 = array([ centerx-pitch/sqrt(3), centery-
                pitch/3, centerz+sqrt(2)*pitch/3 ])
        self.plottriangle(point00,point01,point02)
        self.plottriangle(point10,point11,point12)
        self.plottriangle(point20,point21,point22)
        return array([point00,point01,point02]),array([
            point10,point11,point12]),array([point20,point21,
            point22])
    def defineplotshape(self,(xmin,xmax),(ymin,ymax),(zmin,
        zmax)):
        # Method to define plot shape.
        self.ax.set_xlim3d(xmin,xmax)
```

```
        self.ax.set_ylim3d(ymin,ymax)
        self.ax.set_zlim3d(zmin,zmax)
        return True
    def SavePlot(self,filename):
        # Definition to save the plotted figure. One should
            call it after showplot.
        self.plt.savefig(filename,bbox_inches='tight')
        return True
    def showplot(self,title=None,LabelX=None,LabelY=None,
       filename=None):
        # Shows the prepared plot
        if title != None or LabelX != None or LabelY!= None:
            self.plt.title(title)
            self.plt.xlabel(LabelX)
            self.plt.ylabel(LabelY)
        self.ax.view_init(10.0, -135.0)
        if filename != None:
            self.SavePlot(filename)
        self.plt.show()
        self.CloseFigure()
        return True
    def CloseFigure(self):
        # Method to close the last figure.
        self.plt.close()
        return True
    def CloseAllFigures(self):
        # Method to close all figures.
        self.plt.close('all')
        return True
```

```
class jonescalculus():
    def __init__(self):
        return
    def linearpolarizer(self,input,rotation=0):
        # Linear polarizer, rotation is in degrees and it is
            counter clockwise
        rotation        = radians(rotation)
        rotmat          = array([[cos(rotation),sin(rotation)
            ],[-sin(rotation),cos(rotation)]])
        linearpolarizer = array([[1,0],[0,0]])
        linearpolarizer = dot(rotmat.transpose(),dot(
            linearpolarizer,rotmat))
        return dot(linearpolarizer,input)
    def circullarpolarizer(self,input,type='lefthanded'):
        # Circullar polarizer
        if type == 'lefthanded':
            circullarpolarizer = array([[0.5,-0.5j],[0.5j
                ,0.5]])
        if type == 'righthanded':
            circullarpolarizer = array([[0.5,0.5j],[-0.5j
                ,0.5]])
        return dot(circullarpolarizer,input)
    def quarterwaveplate(self,input,rotation=0):
        # Quarter wave plate, type determines the placing of
            the fast axis
        rotation = radians(rotation)
        rotmat   = array([[cos(rotation),sin(rotation)],[-sin
            (rotation),cos(rotation)]])
```

```
        qwp = array([[1,0],[0,-1j]])
        qwp = dot(rotmat.transpose(),dot(qwp,rotmat))
        return dot(qwp,input)
    def halfwaveplate(self,input,rotation=0):
        # Half wave plate
        rotation = radians(rotation)
        rotmat   = array([[cos(rotation),sin(rotation)],[-sin
            (rotation),cos(rotation)]])
        hwp      = array([[1,0],[0,-1]])
        hwp      = dot(rotmat.transpose(),dot(hwp,rotmat))
        return dot(hwp,input)
    def birefringentplate(self,input,nx,ny,d,wavelength,
        rotation=0):
        # Birefringent plate, d thickness, nx and ny
            refractive indices
        rotation = radians(rotation)
        rotmat   = array([[cos(rotation),sin(rotation)],[-sin
            (rotation),cos(rotation)]])
        delta    = 2*pi*(nx-ny)*d/wavelength
        bfp      = array([[1,0],[0,exp(-1j*delta)]])
        bfp      = dot(rotmat.transpose(),dot(bfp,rotmat))
        return dot(bfp,input)
    def nematicliquidcrystal(self,input,alpha,ne,n0,d,
        wavelength,rotation=0):
        # Nematic liquid crystal, d cell thickness,
            extraordinary refrative index ne, ordinary
            refractive index n0,
        # alpha helical twist per meter in right-hand sense
            along the direction of wave propagation
```

```
        # alpha is calculated is by dividing cell thickness
            to 1 meter length
        # alpha     =  1 /d
        rotation = radians(rotation)
        rotmat   = array([[cos(rotation),sin(rotation)],[-sin
            (rotation),cos(rotation)]])
        beta     = 2*pi*(ne-n0)/wavelength
        lrot     = array([[cos(alpha*d),-sin(alpha*d)],[sin(
            alpha*d),cos(alpha*d)]])
        lretard  = array([[1,0],[0,exp(-1j*beta*d)]])
        lc       = dot(lrot,lretard)
        lc       = dot(rotmat.transpose(),dot(lc,rotmat))
        return dot(lc,input)
    def ferroliquidcrystal(self,input,tetat,ne,n0,d,
        wavelength,fieldsign='+',rotation=0):
        # Ferroelectric liquid crystal, d cell thickness,
            extraordinary refrative index ne, ordinary
            refractive index n0
        # Applied field sign determines the rotation angle
        rotation = radians(rotation)
        tetat    = radians(tetat)
        rotmat   = array([[cos(rotation),sin(rotation)],[-sin
            (rotation),cos(rotation)]])
        beta     = 2*pi*(ne-n0)/wavelength
        lrot1    = array([[cos(tetat),-sin(tetat)],[sin(tetat
            ),cos(tetat)]])
        lrot2    = array([[cos(tetat),sin(tetat)],[-sin(tetat
            ),cos(tetat)]])
        lretard  = array([[1,0],[0,exp(-1j*beta*d)]])
```

```
        if fieldsign == '+':
            lc = dot(dot(lrot1,lretard),lrot2)
        elif fieldsign == '-':
            lc = dot(dot(lrot2,lretard),lrot1)
        lc      = dot(rotmat.transpose(),dot(lc,rotmat))
        return dot(lc,input)
    def electricfield(self,a1,a2):
        return array([[a1],[a2]])

class aperture():
    def __init__(self):
        return
    def twoslits(self,nx,ny,X,Y,delta):
        # Creates a matrix that contains two slits
        obj=zeros((nx,ny),dtype=complex)
        for i in range(int(nx/2-X/2),int(nx/2+X/2)):
            for j in range(int(ny/2+delta/2-Y/2),int(ny/2+
                delta/2+Y/2)):
                obj[ny/2-abs(ny/2-j),i] = 1
                obj[j,i] = 1
        return obj
    def rectangle(self,nx,ny,side):
        # Creates a matrix that contains rectangle
        obj=zeros((nx,ny),dtype=complex)
        for i in range(int(nx/2-side/2),int(nx/2+side/2)):
            for j in range(int(ny/2-side/2),int(ny/2+side/2))
                :
                obj[j,i] = 1
        return obj
```

```
def circle(self,nx,ny,radius):
    # Creates a matrix that contains circle
    obj=zeros((nx,ny),dtype=complex)
    for i in range(int(nx/2-radius/2),int(nx/2+radius/2))
       :
        for j in range(int(ny/2-radius/2),int(ny/2+radius
           /2)):
            if (abs(i-nx/2)**2+abs(j-ny/2)**2)**(0.5)<
               radius/2:
                obj[j,i] = 1
    return obj
def sinamgrating(self,nx,ny,grating):
    # Creates a sinuosidal grating matrix
    obj=zeros((nx,ny),dtype=complex)
    for i in xrange(nx):
        for j in xrange(ny):
            obj[i,j] = 0.5+0.5*cos(2*pi*j/grating)
    return obj
def lens(self,nx,ny,focal,wavelength):
    # Creates a lens matrix
    obj = zeros((nx,ny),dtype=complex)
    k   = 2*pi/wavelength
    for i in xrange(nx):
        for j in xrange(ny):
            obj[i,j] = exp(-1j*k*(pow(i,2)+pow(j,2))/2/
               focal)
    return obj
def gaussian(self,nx,ny,sigma):
    # Creates a 2D gaussian matrix
```

```
        obj = zeros((nx,ny),dtype=complex)
        for i in xrange(nx):
            for j in xrange(ny):
                obj[i,j] = 1/pi/pow(sigma,2)*exp(-float(pow(i
                    -nx/2,2)+pow(j-ny/2,2))/2/pow(sigma,2))
        return obj
    def retroreflector(self,nx,ny,wavelength,pitch,type='
        normal'):
        if nx != ny:
            nx = max([nx,ny])
            ny = nx
        part  = zeros((pitch,int(pitch/2)))
        for i in xrange(int(sqrt(3)*pitch/6)):
            for j in xrange(int(pitch/2)):
                if float(j)/(int(sqrt(3)*pitch/6)-i) < sqrt
                    (3):
                    part[i,j]          = int(sqrt(3)*pitch/6)-i
                    part[pitch-i-1,int(pitch/2)-j-1] = part[i
                        ,j]
        for i in xrange(int(pitch)):
            for j in xrange(int(pitch/2)):
                if j != 0:
                    if float(j)/(int(pitch)-i) < 0.5 and (int
                        (sqrt(3)*pitch/6)-i)/float(j) < 1./
                        sqrt(3):
                        # Distance to plane determines the
                            level of the amplitude
                        # Plane as a line y = slope*x+ pitch
```

```
                    # Perpendicula line  y = -(1/slope)*x
                      +n
                    slope      = -0.5
                    n          = j + (1/slope) * (i)
                    x1         = (n - pitch/2)/(slope+1/
                      slope)
                    y1         = -(1/slope)*x1+n
                    part[i,j] = int(sqrt(3)*pitch/6) -
                      sqrt( pow(i-x1,2) + pow(j-y1,2) )
                    part[pitch-i-1,int(pitch/2)-j-1] =
                      part[i,j]
            else:
                if i > int(sqrt(3)*pitch/6):
                    slope      = -0.5
                    n          = j + (1/slope) * (i)
                    x1         = (n - pitch/2)/(slope+1/
                      slope)
                    y1         = -(1/slope)*x1+n
                    part[i,j] = int(sqrt(3)*pitch/6) -
                      sqrt( pow(i-x1,2) + pow(j-y1,2) )
                    part[pitch-i-1,int(pitch/2)-j-1] =
                      part[i,j]
left  = part
right = part[::-1]
part  = append(left,right,axis=1)
obj   = tile(part,(nx/pitch,ny/pitch))
for i in xrange(nx/pitch/2):
    obj[(2*i+1)*pitch:(2*i+1)*pitch+pitch,:] = roll(
      obj[(2*i+1)*pitch:(2*i+1)*pitch+pitch,:],pitch
```

```
            /2)
        k       = 2*pi/wavelength
        D       = 5
        obj     = pow(obj,3)*exp(1j*k*obj)
        return obj
    def show(self,obj,pixeltom,wavelength,title='Detector',
       type='normal',filename=None,xlabel=None,ylabel=None):
        # Plots a detector showing the given object
        self.plt.figure(),self.plt.title(title)
        nx,ny = obj.shape
        # Number of the ticks to be shown both in x and y
           axes
        if type == 'normal':
            obj = abs(obj)
        elif type == 'log':
            obj = log(abs(obj))
        img = self.plt.imshow(obj,cmap=matplotlib.cm.jet,
           origin='lower')
        self.plt.colorbar(img,orientation='vertical')
        self.plt.xlabel(xlabel)
        self.plt.ylabel(ylabel)
        if filename != None:
            self.plt.savefig(filename)
        self.plt.show()
        return True
    def show3d(self,obj):
        nx,ny   = obj.shape
        fig     = self.plt.figure()
        ax      = fig.gca(projection='3d')
```

```
        X,Y      = mgrid[0:nx,0:ny]
        surf     = ax.plot_surface(X,Y,abs(obj), rstride=1,
            cstride=1, cmap=matplotlib.cm.jet,linewidth=0,
            antialiased=False)
        fig.colorbar(surf, shrink=0.5, aspect=5)
        self.plt.show()
        return True
    def showrow(self,obj,wavelength,pixeltom,distance):
        # Plots row crosssection of the given object
        nx,ny = obj.shape
        a     = 5
        self.plt.figure()
        self.plt.plot(arange(-nx/2,nx/2)*pixeltom,abs(obj[nx
            /2,:]))
        self.plt.show()
        return True

class beams():
    def __init(self):
        return
    def spherical(self,nx,ny,distance,wavelength,pixeltom,
        focal,amplitude=1):
        # Spherical wave
        distance = abs(focal-distance)
        k        = 2*pi/wavelength
        X,Y      = mgrid[-nx/2:nx/2,-ny/2:ny/2]*pixeltom
        r        = sqrt(pow(X,2)+pow(Y,2)+pow(distance,2))
        U        = amplitude/r*exp(-1j*k*r)
        return U
```

```
    def gaussian(self,nx,ny,distance,wavelength,pixeltom,
        amplitude,waistsize,focal=0):
        # Gaussian beam
        distance = abs(distance-focal)
        X,Y      = mgrid[-nx/2:nx/2,-ny/2:ny/2]*pixeltom
        ro       = sqrt(pow(X,2)+pow(Y,2))
        z0       = pow(waistsize,2)*pi/wavelength
        A0       = amplitude/(1j*z0)
        if distance == 0:
            U = A0*exp(-pow(ro,2)/pow(waistsize,2))
            return U
        k        = 2*pi/wavelength
        R        = distance*(1+pow(z0/distance,2))
        W        = waistsize*sqrt(1+pow(distance/z0,2))
        ksi      = 1./arctan(distance/z0)
        U        = A0*waistsize/W*exp(-pow(ro,2)/pow(W,2))*
            exp(-1j*k*distance-1j*pow(ro,2)/2/R+1j*ksi)
        return U

class diffractions():
    def __init__(self):
        return
    def fft(self,obj):
        return fftshift(fft2(obj))
    def fresnelfraunhofer(self,wave,wavelength,distance,
        pixeltom,aperturesize):
        nu,nv    = wave.shape
        k        = 2*pi/wavelength
        X,Y      = mgrid[-nu/2:nu/2,-nv/2:nv/2]*pixeltom
```

```
            Z       = pow(X,2)+pow(Y,2)
            distancecritical = pow(aperturesize*pixeltom,2)*2/
                wavelength
            print 'Critical distance of the system is %s m.
                Distance of the detector is %s m.' % (
                distancecritical,distance)
            # Convolution kernel for free space
            h       = exp(1j*k*distance)/sqrt(1j*wavelength*
                distance)*exp(1j*k*0.5/distance*Z)
            qpf     = exp(-1j*k*0.5/distance*Z)
            if distancecritical < distance:
                wave = wave*qpf
            result = fftshift(ifft2(fft2(wave)*fft2(h)))
            return result
        def fresnelnumber(self,aperturesize,pixeltom,wavelength,
            distance):
            return  pow(aperturesize*pixeltom,2)/wavelength/
                distance
        def intensity(self,obj,pixeltom):
            return abs(pow(obj,2))*pow(pixeltom,2)*0.5*8.854*pow
                (10,-12)*299792458

def main():
    print 'Odak by %s' % __author__
    return True

if __name__ == '__main__':
    sys.exit(main())
```

5.3 Appendix C: Multi-view display control using Linux under ARM processors

Screens update routine

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

__author__ = ('Kaan Akşit')

import sys,os,time

# Function to update screens.
def UpdateScreen(tty=6,ImagePath='./Content/blank.png'):
    command = 'fbi -T %s %s --autozoom --noverbose' % (tty,
        ImagePath)
    os.system(command)
    return True

# Function to assign TTYs to found framebuffer devices.
def AssignTtysToScreens(ScreenList):
    TtyList = []
    del TtyList[:]
    for i in xrange(0,len(ScreenList)):
        tty = '%s' % (i+10)
        TtyList.append(tty)
        command = './con2fb %s %s' % (ScreenList[i],TtyList[i
            ])
        print command
        os.system(command)
```

```
    return TtyList

# Finds the framebuffer devices attached to the system.
def FindScreens():
    ScreenList = []
    del ScreenList[:]
    pipe       = os.popen('ls /dev/fb*')
    for line in pipe.readlines():
        ScreenList.append(line.replace("\n",""))
    return ScreenList

# Main function started when the script is initiated directly
    .
def main():
    # Printing author information.
    print 'PI3B script, Author(s): %s' % __author__
    # Check for flags to know which screens to update.
    UpdateList = []
    for SelectedScreens in sys.argv:
        if len(SelectedScreens) < 2:
            if int(SelectedScreens) < 6 and int(
               SelectedScreens) > -1:
                UpdateList.append(int(SelectedScreens))
    if len(sys.argv) == 1:
        UpdateList = [0,1,2,3,4,5]
    print "Screens to be update:", UpdateList
    # Finding the screens connected to the device.
    print '\nFramebuffer devices found:'
    screens = FindScreens()
```

```
    print screens
    # Assigning TTY to each framebuffer device.
    print '\nAssigned TTY values for found Framebuffer
        devices:'
    ttys    = AssignTtysToScreens(screens)
    print ttys
    # Test pattern is displayed on the each display.
    test(ttys,screens,"no")
    # If flag to clear all screen is iniated, this block is
        valid.
    if len(sys.argv) > 1:
        if int(sys.argv[1]) == 666:
            print 'All screens are cleared!'
            sys.exit()
            return True
    # Each screen is being update with the necessary content.
    # Change the tdelay (secs) to adjust the time in between
        each screen update.
    tdelay  = 1
    for i in UpdateList:
        UpdateScreen(ttys[i],'./Content/samplescreen%d.png' %
            i)
        print 'Updating %s on %s' % (screens[i],ttys[i])
        time.sleep(tdelay)
    return True

# Display test pattern at each framebuffer device.
def test(ttys,screens,wait="no"):
```

```
    print '\nSample Update of screens make sure all of them
       is displaying logo!'
    for i in xrange(0,len(screens)):
        UpdateScreen(ttys[i])
        print 'Updating %s on %s' % (screens[i],ttys[i])
        if wait == "yes":
            time.sleep(1)
    return True

if __name__ == '__main__':
    sys.exit(main())
```

Content modification routine

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

__author__ = ('Kaan Akşit')

import sys,os,time,pygame,socket,csv
from pygame.locals import *

# CSV reader to read offset positions of the pico projectors.
def ReadCSV(filename,BlockNumber):
    # Open a socket to read CSV.
    ifile  = open(filename, "rb")
    reader = csv.reader(ifile)
    offsets = []
    # Skip header
    next(reader)
    # Read the CSV row by row.
```

```
    for row in reader:
        # Avoid empty lines in CSV
        if len(row) > 0:
            # Strip white spaces from CSV file.
            row = ([element.strip() for element in row])
            # Match the block number to get the related data
                into the array.
            if row[0] == BlockNumber:
                offsets.append(row)
    # Close the socket.
    ifile.close()
    return offsets

# If ShowImage is set to yes, the content is created using
    sample input under Content folder.
# BlockNumber determines the position of the six pico
    projector.
def main(ShowImage='yes'):
    # Setting ShowImage through shell.
    if len(sys.argv) > 1:
        if sys.argv[1] == 'no':
            ShowImage = 'no'
    # Number of views.
    NumberOfViews = 6
    # Recognize Raspberry PI.
    NameOfTheHost = socket.gethostname()
    if NameOfTheHost == 'PI3B01':
        BlockNumber           = 'a1'
        ImageCounterConstant = 0
```

```
elif NameOfTheHost == 'PI3B02':
    BlockNumber           = 'a2'
    ImageCounterConstant = NumberOfViews
elif NameOfTheHost == 'PI3B03':
    BlockNumber           = 'a3'
    ImageCounterConstant = 2*NumberOfViews
# Reading offsets.csv to retrieve the offset values.
offsets = ReadCSV("offsets.csv",BlockNumber)
# Width, and height of the desired image.
width          = 848
height         = 480
# Colors are defined.
colors      = [
              (255,0,0),
              (0,255,0),
              (0,0,255),
              (255,255,0),
              (255,0,255),
              (0,255,255),
              (100,20,50),
              (255,255,255),
              (127,100,0),
              (0,255,100),
              (0,100,127),
              (127,127,0),
              (127,255,127),
              (0,127,127),
              (200,0,50),
              (50,255,0),
```

```
(100,50,100),
(150,150,0),
(0,150,150),
(100,200,0),
(0,100,200),
(200,0,100),
(100,0,200),
(200,100,0),
(50,0,255),
(100,150,50),
(0,255,150),
(50,100,70),
(0,30,200),
(100,100,0),
(40,10,23),
(70,80,90),
(138,56,190),
(123,23,10),
(234,123,34),
(4,4,233),
(23,56,12),
(43,34,34),
(43,255,34),
(65,45,23),
(90,90,56),
(45,122,56),
(230,120,42),
(45,45,200),
(80,0,255),
```

```
                (100,200,67),
                (200,0,34),
                (100,250,230),
                (45,12,32),
                (12,99,200),
                (234,234,53),
                (43,89,75),
                ]
# Slit is defined geometrically.
SlitHeight      = 13
SlitSize        = [0, SlitHeight]
# Load multiview images and slice them into pieces.
ImageSlices     = []
ImageCounter    = []
# List of image files to be used.
MultiViewImages = []
hup             = 36
for no in xrange(0,hup):
    MultiViewImages.append('./Blender/image%s.png' % no)
# Reverse order the images for correct registration on
   the screen.
MultiViewImages = reversed(MultiViewImages)
# Create Image Slices to be displayed by each pico
   projector.
for ImageName in MultiViewImages:
    # Adding a new slice to the slices matrix.
    ImageSlices.append(LoadImage(ImageName, SlitHeight))
# Number of slits calculated. +10 to avoid missing slit.
NumberOfSlits = height / SlitSize[1] + 10
```

```
# ImageCounter vector is built.
for z in xrange(0,NumberOfSlits):
    # ImageCounter used in displaying right images.
   ImageCounter.append(ImageCounterConstant)
# Loop to create each view.
for j in xrange(0,NumberOfViews):
    # Setting offset
    OffsetLeft   = int(offsets[j][2])
    OffsetTop    = int(offsets[j][3])
    SlitSize[0] = int(offsets[j][4])
    # Creating the new surface.
    NewSurface = pygame.Surface((width, height))
    # Loop to create each slit.
    for i in xrange(0,NumberOfSlits):
        if (i*SlitSize[1] + OffsetTop) > 0:
            slit        = pygame.Rect((OffsetLeft,(i*
                SlitSize[1] + OffsetTop)), SlitSize)
        else:
            slit        = pygame.Rect((OffsetLeft,((
                NumberOfSlits + i)*SlitSize[1] + OffsetTop
                )), SlitSize)
        pygame.draw.rect(NewSurface, colors[i], slit, 0)
        # Check if the OffsetLeft is calibrated by
            drawing a line at the center.
        pygame.draw.rect(NewSurface, (255,255,255),
            pygame.Rect(slit.centerx-100,slit.centery,
            SlitHeight,slit.height) ,0)
        # If image display is desired, this if loop takes
            on.
```

```
            if ShowImage == 'yes':
                # Specify which color is replaced with an
                    image.
                a = i
                # Choosing the specific slices in the image
                    for correct image registration.
                for c in xrange(0,hup):
                    if colors[i] == colors[c]:
                        ChosenImage = ImageSlices[c]
                        # Adjusting image according to the
                            image height.
                        ChosenImage[ImageCounter[a]] = pygame
                            .transform.scale(ChosenImage[
                            ImageCounter[a]],(SlitSize[0],
                            SlitSize[1]))
                        # Necessary slice is being place
                            accordingly.
                        NewSurface.blit(ChosenImage[
                            ImageCounter[a]], slit)
                        # Increasing the image counter to
                            take right slice in the next step.
                        ImageCounter[a] += 1
        pygame.image.save(NewSurface, './Content/samplescreen
            %d.png' % j)
    os.system("mv ./Content/samplescreen1.png ./Content/
        samplescreen12.png")
    os.system("mv ./Content/samplescreen5.png ./Content/
        samplescreen1.png")
```

```
        os.system("mv ./Content/samplescreen2.png ./Content/
            samplescreen13.png")
        os.system("mv ./Content/samplescreen3.png ./Content/
            samplescreen14.png")
        os.system("mv ./Content/samplescreen4.png ./Content/
            samplescreen15.png")
        os.system("mv ./Content/samplescreen12.png ./Content/
            samplescreen2.png")
        os.system("mv ./Content/samplescreen13.png ./Content/
            samplescreen3.png")
        os.system("mv ./Content/samplescreen14.png ./Content/
            samplescreen4.png")
        os.system("mv ./Content/samplescreen15.png ./Content/
            samplescreen5.png")
        return True

# Function to load image and slice it
def LoadImage(path,SlitHeight=20,reverse=0,width=234,height
    =848,rotate='yes'):
        # Image load takes place.
        Image    = pygame.image.load(path)
        # Rotate Image.
        Image    = pygame.transform.rotate(Image, -90)
        # Mirror image horizontally.
        Image    = pygame.transform.flip(Image,True,False)
        # Transform the image into usable format.
        Image    = pygame.transform.scale(Image,(width, height))
        # Image properties are saved.
        ImgH     = Image.get_height()
```

```
    ImgW     = Image.get_width()
    Cropped = []
    # Producing the slices.
    for i in xrange(0,ImgW/SlitHeight):
        # Cropping takes place.
        Cropped.append(pygame.Surface((SlitHeight,ImgH)))
        Cropped[i].blit(Image, (0,0), (i*SlitHeight,0,
            SlitHeight,ImgH))
        # Rotating the each slice with 90.
        Cropped[i] = pygame.transform.rotate(Cropped[i], -90)
    # Reverse ordering slices.
    if reverse == 1:
        Cropped = Cropped[::-1]
    return Cropped


if __name__ == '__main__':
    sys.exit(main())
```

Offset look up table of the prototype

```
Block,ProjectorNumber,OffsetLeft,OffsetTop,SlitSize
a1,0,0,0,848
a1,1,30,-12,848
a1,2,40,-22,848
a1,3,43,-19,848
a1,4,30,-25,848
a1,5,25,-23,848
a2,0,0,-33,848
a2,1,37,-34,848
a2,2,48,-30,848
a2,3,48,-30,848
```

a2,4,35,−53,848

a2,5,12,−60,848

a3,0,15,−45,848

a3,1,40,−40,848

a3,2,43,−53,848

a3,3,21,−53,848

a3,4,40,−55,848

a3,5,45,−60,848

PUBLICATION RECORD

[K1] K. Akşit, , O. Eldes, S. Viswanathen, M. Freeman, and H. Urey, "Portable 3D Laser Projector using Mixed Polarization Technique," Journal of Display Technology **8**, 582–589 (2012).

[K2] K. Akşit, O. Eldes, S. Viswanathen, M. Freeman, and H. Urey, "Mixed Polarization 3D Technique for Scanned Laser Pico Projector Displays," in "IMID2012: The 12th International Meeting on Information Display, August," (SID, 2012).

[K3] H. Urey, K. Aksit, and O. Eldes, "Novel 3d displays using micro-optics and mems," in "International Conference on Fibre Optics and Photonics," (Optical Society of America, 2012).

[K4] H. Urey, S. Holmstrom, U. Baran, K. Aksit, M. Hedili, and O. Eldes, "Mems scanners and emerging 3d and interactive augmented reality display applications," (Invited talk in The 17th International Conference on Solid-State Sensors, Actuators and Microsystems, 2013).

[K5] K. Akşit, A. H. Ghanbari Niaki, and H. Urey, "Super Stereoscopic 3D Glasses for More Realistic 3D Vision," in "3DTV Conference: In pursuit of next generation 3D Displays(3DTV-CON), Hungary, 2014," (IEEE, 2014).

[K6] K. Akşit, A. H. Ghanbari Niaki, and H. Urey, "Improved 3D with Super Stereoscopy Technique," in "SID DISPLAYWEEK 2014: San Diego, California, June," (SID, 2014).

[K7] K. Akşit, M. Kadioglu, and H. Urey, "Head-worn Augmented Reality Projection Display," in "SID-ME 2014: Mid-Europe Chapter Sprint Meeting at Istanbul, April," (SID, 2014).

[K8] O. Eldes, K. Akşit, and H. Urey, "Multi-view autostereoscopic projection display using rotating screen," Optics Express **21**, 29043–29054 (2013).

[K9] O. Eldes, K. Akşit, and H. Urey, "Paper No 17.4: 3D Auto-stereoscopic display using pico-projectors and rotating screen," in "EURODISPLAY2013: 33rd International Display Research Conference, September," (SID, 2013).

[K10] O. Eldes, K. Akşit, and H. Urey, "Auto-Stereoscopic Projection Display using Head-Tracker and Rotating Screen," in "SID-ME 2014: Mid-Europe Chapter Sprint Meeting at Istanbul, April," (SID, 2014).

[K11] K. Akşit, S. Olcer, and H. Urey, "Modular Multi-projection Multi-View Autostereoscopic Display using MEMS Laser Projectors," in "SID-ME 2014: Mid-Europe Chapter Sprint Meeting at Istanbul, April," (SID, 2014).

[K12] K. Akşit, S. Olcer, and H. Urey, "Modular Multi-projection Multi-View Autostereoscopic Display using MEMS Laser Projectors," in "SID DISPLAYWEEK 2014: San Diego, California, June," (SID, 2014).

[K13] K. Akşit, , H. Baghsiahi, S. Olçer, E. Wilmann, S. E. Day, H., D. Selviah, P. Surman, and H. Urey, "Dynamic Exit Pupil Trackers for Autostereoscopic Displays," Optics Express **21**, 14331–14341 (2013).

[K14] K. Akşit, S. Olçer, E. Erden, V. Kishore, H. Urey, E. Willman, H. Baghsiahi, S. Day, D. Selviah, F. Aníbal Fernández *et al.*, "Light engine and optics for HELIUM3D auto-stereoscopic laser scanning display," in "3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), Turkey, 2011," (IEEE, 2011), pp. 1–4.

[K15] H. Baghsiahi, D. Selviah, E. Willman, A. Fernández, S. Day, K. Akşit, S. Ölçer, A. Mostafazadeh, E. Erden, V. Kishore *et al.*, "48.4: Beam Forming for a Laser Based Auto-stereoscopic Multi-Viewer Display," (SID, 2011).

[K16] P. Surman, B. Day, B. Boby, H. Chen, K. Akşit, and H. Urey, "Paper No 15.2: Head-Tracked Retroreflecting 3D Display," in "EURODISPLAY2013: 33rd International Display Research Conference, September," (SID, 2013).

[K17] P. Surman, S. Day, K. Aksit, H. Urey, J. Benjamin, K. Jain, and H. Chen, "Single and multi-user head tracked glasses-free 3d displays," in "3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2013," (2013), pp. 1–4.

[K18] K. Akşit, O. Eldes, and H. Urey, "Multiple Body Tracking for Interactive Mobile Projectors," in "IMID2012: The 12th International Meeting on Information Display, August," (SID, 2012).

[K19] Y. Cakmak, K. Akşit, S. Olcer, and H. Urey, "A biomedical device to alliviate the parkinson symptoms," (2013). Europe PCT/IB2013/055327 WIPO (SUBMITTED).

[K20] G. Corbellini, K. Akşit, S. Mangold, S. Schmid, and T. R. Gross, "Connecting Networks of Toys and Smartphones with Visible Light Communication," IEEE Communication magazine (2014).

BIBLIOGRAPHY

[1] I. Microvision, "Microvision: A World of Display and Imaging Oppurtunities," `http://www.microvision.com` (2012).

[2] C. Wheatstone, "The Bakerian Lecture–Contributions to the Physiology of Vision.–Part the Second. On Some Remarkable, and Hitherto Unobserved, Phenomena of Binocular Vision (Continued)," Philosophical Transactions of the Royal Society of London **142**, 1–17 (1852).

[3] Y. Takaki, H. Takenaka, Y. Morimoto, O. Konuma, and K. Hirabayashi, "Multi-view display module employing mems projector array," Optics express **20**, 28257–28266 (2012).

[4] M. Kawakita, S. Iwasawa, M. Sakai, Y. Haino, M. Sato, and N. Inoue, "3d image quality of 200-inch glasses-free 3d display system," in "IS&T/SPIE Electronic Imaging," (International Society for Optics and Photonics, 2012), pp. 82880B–82880B.

[5] S. Yoshida, M. Kawakita, and H. Ando, "Light-field generation by several screen types for glasses-free tabletop 3d display," in "3DTV Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2011," (IEEE, 2011), pp. 1–4.

[6] W. Matusik and H. Pfister, "3d tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes," in "ACM Transactions on Graphics (TOG)," , vol. 23 (ACM, 2004), vol. 23, pp. 814–824.

[7] K. Hirabayashi, H. Takenaka, O. Konuma, Y. Morimoto, and Y. Takaki, "Multi-view display module using mems projectors for an ultra-large screen autostereo-

scopic display," in "IS&T/SPIE Electronic Imaging," (International Society for Optics and Photonics, 2013), pp. 86480H–86480H.

[8] A. Jones, K. Nagano, J. Liu, J. Busch, X. Yu, M. Bolas, and P. Debevec, "Interpolating vertical parallax for an autostereoscopic three-dimensional projector array," Journal of Electronic Imaging **23**, 011005–011005 (2014).

[9] J. W. Goodman and S. C. Gustafson, "Introduction to fourier optics," Optical Engineering **35**, 1513–1513 (1996).

[10] M. C. Teich and B. Saleh, "Fundamentals of photonics," Canada, Wiley Interscience p. 3 (1991).

[11] H. B. Pryor, "Objective measurement of interpupillary distance," Pediatrics **44**, 973–977 (1969).

[12] A. Zajaczkowska, "Experimental test of luneburgs theory. horopter and alley experiments," JOSA **46**, 514–527 (1956).

[13] S. De Groot and J. Gebhard, "Pupil size as determined by adapting luminance," JOSA **42**, 492–495 (1952).

[14] J. Liang and D. R. Williams, "Aberrations and retinal image quality of the normal human eye," JOSA A **14**, 2873–2883 (1997).

[15] C. Pulfrich, "Die stereoskopie im dienste der isochromen und heterochromen photometrie," Naturwissenschaften **10**, 751–761 (1922).

[16] H. Urey, K. V. Chellappan, E. Erden, and P. Surman, "State of the art in stereoscopic and autostereoscopic displays," Proceedings of the IEEE **99**, 540–555 (2011).

[17] B. A. Parviz, "Augmented reality in a contact lens," IEEE spectrum **9**, 1–4 (2009).

[18] R. Sprague, A. Zhang, L. Hendricks, T. O'Brien, J. Ford, E. Tremblay, and T. Rutherford, "Novel hmd concepts from the darpa scenicc program," in "SPIE Defense, Security, and Sensing," (International Society for Optics and Photonics, 2012), pp. 838302–838302.

[19] K. Akşit, O. Eldes, S. Viswanathan, M. O. Freeman, and H. Urey, "Portable 3d laser projector using mixed polarization technique," Journal of Display Technology **8**, 582–589 (2012).

[20] C. Ellinger and P. Kane, "2D/3D SWITCHABLE COLOR DISPLAY APPARATUS WITH NARROW BAND EMITTERS," (2011). US Patent 20,110,285,705.

[21] H. Jorke and M. Fritz, "Infitec-a new stereoscopic visualisation tool by wavelength multiplex imaging," Proceedings of Electronic Displays, September (2003).

[22] C. Lanfranchi and C. Brossier, "METHOD AND EQUIPMENT FOR PRODUCING AND DISPLAYING STEREOSCOPIC IMAGES WITH COLOURED FILTERS," (2010). EP Patent 2,162,794.

[23] I. Howard and B. Rogers, *Perceiving in Depth, Volume 2: Stereoscopic Vision*, vol. 29 (Oxford Univ Pr, 2012).

[24] A. Woods and C. Harris, "Comparing levels of crosstalk with red/cyan, blue/yellow, and green/magenta anaglyph 3D glasses," in "Proceedings of SPIE Stereoscopic Displays and Applications," , vol. 7524 (2010), vol. 7524, p. 75240Q.

[25] M. Cowan, J. Greer, L. Lipton, and J. Chiu, "Enhanced ZScreen modulator techniques," (2007). WO Patent WO/2007/067,493.

[26] I. RealD, "RealD - The new 3D," `http://reald.com/` (2012).

[27] L. Bogaert, Y. Meuret, B. Van Giel, H. De Smet, and H. Thienpont, "Design of a compact projection display for the visualization of 3-D images using polarization sensitive eyeglasses," J. Soc. Inf. Display **17**, 603–609 (2009).

[28] S. Faris, "Novel 3D stereoscopic imaging technology," in "Proceedings of SPIE," , vol. 2177 (1994), vol. 2177, p. 180.

[29] Y. Wu, Y. Jeng, P. Yeh, C. Hu, and W. Huang, "20.2: Stereoscopic 3D Display Using Patterned Retarder," (SID, 2008).

[30] G. Saitoh and M. Imai, "LIQUID CRYSTAL SHUTTER GLASSES," (2009). WO Patent WO/2009/037,940.

[31] K. Chellappan, E. Erden, and H. Urey, "Laser-based displays: a review," Applied optics **49**, F79–F98 (2010).

[32] M. Freeman, M. Champion, and S. Madhavan, "Scanned laser pico-projectors: Seeing the big picture (with a small device)," Optics and Photonics News **20**, 28–34 (2009).

[33] J. Goodman, *Introduction to Fourier optics* (Roberts & Company Publishers, 2005).

[34] I. Micron Technology, "Micron Technology, Inc. - DRAM, NAND Flash, NOR Flash, MCP, SSD ,FLCOS," `http://www.micron.com` (2011).

[35] S.-U. GmbH, "SilverFabric - Optical Elements for Professional 3-D Projection," `http://www.silverfabric3d.de/` (2011).

[36] H. Urey and K. Powell, "Optical element that includes a microlens array and related method," (2005). US Patent App. 20,050/248,849.

[37] A. Woods, "Understanding crosstalk in stereoscopic displays," in "Keynote Presentation) at Three-Dimensional Systems and Applications (3DSA) conference, Tokyo, Japan," (Citeseer, 2010), pp. 19–21.

[38] P. Seuntiëns, L. Meesters, and W. IJsselsteijn, "Perceptual attributes of crosstalk in 3d images," Displays **26**, 177–183 (2005).

[39] A. Al-Qasimi, O. Korotkova, D. James, and E. Wolf, "Definitions of the degree of polarization of a light beam," Optics letters **32**, 1015–1016 (2007).

[40] H. Urey and K. Powell, "Microlens-array-based exit-pupil expander for full-color displays," Applied optics **44**, 4930–4936 (2005).

[41] E. Lueder, *3D Displays*, Wiley Series in Display Technology (Wiley, 2011).

[42] Y. Kajiki, H. Yoshikawa, and T. Honda, "Hologram-like video images by 45-view stereoscopic display," in "Proc. SPIE," , vol. 3012 (1997), vol. 3012, pp. 154–166.

[43] Y. Takaki, K. Tanaka, and J. Nakamura, "Super multi-view display with a lower resolution flat-panel display," Optics Express **19**, 4129–4139 (2011).

[44] K. Choi, H. Lee, Y. Choi, J. Bae, and H. Song, "3d display apparatus and method of displaying 3d images," (2012). US Patent App. 13/298,142.

[45] Y. Takaki and N. Nago, "Multi-projection of lenticular displays to construct a 256-view super multi-view display," Optics Express **18**, 8824–8835 (2010).

[46] J. Nakamura, K. Tanaka, and Y. Takaki, "Accommodation responses to horizontal-parallax-only super multi-view display," in "IS&T/SPIE Electronic Imaging," (International Society for Optics and Photonics, 2013), pp. 864818–864818.

[47] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," Computer vision and image understanding **104**, 90–126 (2006).

[48] T. Pascu, M. White, and Z. Patoli, "Motion capture and activity tracking using smartphone-driven body sensor networks," in "Innovative Computing Technology (INTECH), 2013 Third International Conference on," (IEEE, 2013), pp. 456–462.

[49] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović, "Practical motion capture in everyday surroundings," ACM Trans. Graph. **26** (2007).

[50] D. Kade, O. Özcan, and R. Lindell, in "Proceedings of the ICCGMAT 2013 International Conference on Computer Games, Multimedia and Allied Technology," (World Academy of Science, Engineering and Technology, 2013), pp. 500–506.

[51] D. Kade, O. Özcan, and R. Lindell, "Towards stanislavski-based principles for motion capture acting in animation and computer games," (2013).

[52] O. Cakmakci and J. Rolland, "Head-worn displays: a review," Journal of Display Technology **2**, 199–216 (2006).

[53] J. P. Rolland, K. P. Thompson, H. Urey, and M. Thomas, "See-through head worn display (hwd) architectures," in "Handbook of Visual Display Technology," (Springer, 2012), pp. 2145–2170.

[54] J. Rolland and K. Thompson, "See-through head worn displays for mobile augmented reality," in "Proceedings of the China National Computer Conference," (2011).

[55] M. I. Olsson, M. J. Heinrich, D. Kelly, and J. Lapetina, "Wearable device with input and output structures," (2013). US Patent 20,130,044,042.

[56] M. Guillaumée, P. Vahdati, E. Tremblay, A. Mader, G. Bernasconi, V. Cadarso, J. Grossenbacher, J. Brugger, R. Sprague, and C. Moser, "Curved holographic combiner for color head worn display," (2013).

[57] N. Firth, "First wave of virtual reality games will let you live the dream," New Scientist **218**, 19–20 (2013).

[58] R. Martins, V. Shaoulov, Y. Ha, and J. Rolland, "A mobile head-worn projection display," Opt. Express **15**, 14530–14538 (2007).

[59] H. Hua, C. Gao, and J. P. Rolland, "Imaging properties of retro-reflective materials used in head-mounted projective displays (hmpds)," in "AeroSense 2002," (International Society for Optics and Photonics, 2002), pp. 194–201.

[60] M. Bolas and D. M. Krum, "Augmented reality applications and user interfaces using head-coupled near-axis personal projectors with novel retroreflective props and surfaces," in "Pervasive 2010 Ubiprojection Workshop," (2010).

[61] P. Mistry and P. Maes, "Sixthsense: a wearable gestural interface," in "ACM SIGGRAPH ASIA 2009 Sketches," (ACM, 2009), p. 11.

[62] P. Mistry, P. Maes, and L. Chang, "Wuw-wear ur world: a wearable gestural interface," in "CHI'09 extended abstracts on Human factors in computing systems," (ACM, 2009), pp. 4111–4116.

[63] C. Harrison, H. Benko, and A. D. Wilson, "Omnitouch: wearable multitouch interaction everywhere," in "Proceedings of the 24th annual ACM symposium on User interface software and technology," (ACM, 2011), pp. 441–450.

[64] T. Illusions, "Castar," http://technicalillusions.com/castar/ (2014).

[65] T. Sonoda, T. Endo, N. Kawakami, and S. Tachi, "X'talvisor: full open type head-mounted projector," in "ACM SIGGRAPH 2005 Emerging technologies," (ACM, 2005), p. 32.

[66] R. T. M. KANBARA and N. YOKOYA, "A wearable augmented reality system using positioning infrastructures and a pedometer," in "Proceedings of the Seventh

IEEE International Symposium on Wearable Computers (ISWC03)," , vol. 1530 (2003), vol. 1530, pp. 17–00.

[67] M. Inami, N. Kawakami, D. Sekiguchi, Y. Yanagida, T. Maeda, and S. Tachi, "Visuo-haptic display using head-mounted projector," in "Virtual Reality, 2000. Proceedings. IEEE," (2000), pp. 233–240.

[68] G. Smits and D. Kikinis, "System and method for 3-d projection and enhancements for interactivity," (2013). US Patent App. 13/877,652.

[69] M. O. Freeman, S. P. Viswanathan, and D. Lashmet, "Mixed polarization imaging system for three-dimensional projection and corresponding methods," (2013). US Patent 20,130,038,837.

[70] R. D. DeMaster, "Low-profile raised retroreflective pavement marker," (1977). US Patent 4,035,059.

[71] M. S. Scholl, "Ray trace through a corner-cube retroreflector with complex reflection coefficients," JOSA A **12**, 1589–1592 (1995).

[72] J. Snyder, "Paraxial ray analysis of a cats-eye retroreflector," Applied Optics **14**, 1825–1828 (1975).

[73] A. F. Watch, "The anaglyph: a new method of producing the stereoscopic effect," Journal of the Franklin Institute **140**, 401–419 (1895).

[74] L. PICTET, "Device for projecting and viewing stereoscopic pictures," (1924). US Patent 1,503,766.

[75] L. Hammond, "Stereoscopic motion-eectuke device," (1924). US Patent 1,506,524.

[76] J.-S. Kim, D. Gracanin, and F. Quek, "Sensor-fusion walking-in-place interaction technique using mobile devices," in "Virtual Reality Workshops (VR), 2012 IEEE," (IEEE, 2012), pp. 39–42.

[77] Q. Wang, Y. Tao, W. Zhao, and D. Li, "A full resolution autostereoscopic 3d display based on polarizer parallax barrier," Chinese Optics Letters **8**, 22–23 (2010).

[78] H. Baker and Z. Li, "Camera and projector arrays for immersive 3d video," in "Proceedings of the 2nd International Conference on Immersive Telecommunications," (ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009), p. 23.

[79] W. Matusik and H. Pfister, "3d tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes," in "ACM Transactions on Graphics (TOG)," , vol. 23 (ACM, 2004), vol. 23, pp. 814–824.

[80] P. V. Harman, "Autostereoscopic teleconferencing system," in "Electronic Imaging," (International Society for Optics and Photonics, 2000), pp. 293–302.

[81] R. Borner, B. Duckstein, O. Machui, H. Roder, T. Sinnig, and T. Sikora, "A family of single-user autostereoscopic displays with head-tracking capabilities," Circuits and Systems for Video Technology, IEEE Transactions on **10**, 234–243 (2000).

[82] S. S. Kim, S. A. Shestak, K. H. Cha, and J. H. Sung, "Multiview 3d projection system," in "Electronic Imaging 2004," (International Society for Optics and Photonics, 2004), pp. 222–226.

[83] C. Gao and J. Xiao, "Retro-reflective light diffusing autostereoscopic 3d display systems," (2012). US Patent 8,328,360.

[84] K. Akşit, "kunguz/osman," `https://github.com/kunguz/osman` (2012).

[85] O. E. GmbH, "Reflective products - lighting optics, optical engineers, polymer processing - Reflexite," `http://www.reflexite.com` (2012).

[86] L. Co., "Luminit, The Light Shaping Diffuser (LSD) Company Luminit Shaping Light as Needed," `http://www.luminitco.com` (2012).

[87] A. J. Woods, "How are crosstalk and ghosting defined in the stereoscopic literature?" in "IS&T/SPIE Electronic Imaging," (International Society for Optics and Photonics, 2011), pp. 78630Z–78630Z.

[88] F. L. Kooi and A. Toet, "Visual comfort of binocular and 3d displays," Displays **25**, 99–108 (2004).

[89] P. V. Harman, "Retroreflective screens and their application to autostereoscopic displays," in "Electronic Imaging'97," (International Society for Optics and Photonics, 1997), pp. 145–153.

[90] T. Balogh, "The holovizio system," in "Electronic Imaging 2006," (International Society for Optics and Photonics, 2006), pp. 60550U–60550U.

[91] J. Jurik, A. Jones, M. Bolas, and P. Debevec, "Prototyping a light field display involving direct observation of a video projector array," in "Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on," (IEEE, 2011), pp. 15–20.