CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering
Department of Radioelectronics

Software tools for autostereoscopic display

Programové vybavení pro autostereoskopický displej

Master Thesis
Diplomová práce

Study programme: Communications, Multimedia and Electronics
Branch of study: Multimedia technology

Project advisor: Ing. Karel Fliegel, Ph.D.

Bc. David Inneman

Prague 2016

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra radioelektroniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **David Inneman**

Studijní program: Komunikace, multimédia a elektronika
Obor: Multimediální technika

Název tématu: **Programové vybavení pro autostereoskopický displej**

Pokyny pro vypracování:

Podejte přehled metod zobrazování na autostereoskopickém displeji včetně metod generování obrazového signálu pro tento displej. Ve vhodném prostředí realizujte programové vybavení, které umožní přímo budit autostereoskopický displej různými formáty vstupního obrazu. Funkčnost navržených algoritmů otestujte.

Seznam odborné literatury:

[1] Ozaktas, H. M., Onural, L.: Three-Dimensional Television: Capture, Transmission, Display, Springer, 2008.
[2] Javidi, B., Okano, F.: Three-Dimensional Television, Video, and Display Technologies, Springer, 2002.
[3] Technická dokumentace k autostereoskopickému displeji Philips/Dimenco BDL4251VS.

Vedoucí: Ing. Karel Fliegel, Ph.D.

Platnost zadání: do konce letního semestru 2015/2016

L.S.

doc. Mgr. Petr Páta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 2. 2015

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60(1) of the Act.

In Prague on 11th January 2016 .....

Abstract

The goal of this work is to create an overview of principles of human depth perception and their exploitation by various “3D” display technologies and to develop a computer software or compile a set of applications that would allow to correctly display images on a multi-view autostereoescopic display without the need for any third party hardware solution.

In order to be able to display multiview images properly, images of individual views have to be interleaved using an interlacing pattern, which depends on parameters of the lenticular lens sheet covering the display surface. As these parameters are not disclosed by the manufacturer of the display, an empirical approach is used in order to obtain the needed interlacing pattern.

An application is then created using the information obtained. The software allows the use of some of the most common stereoscopic image formats and allows the user to set basic parameters of the output image, which is then generated and fed to the display directly, instead of to a hardware rendering box supplied with the autostereoscopic screen. Image processing done by the application is thoroughly described and a simple user manual is provided.

Output images generated by the application are then compared to images processed by the hardware rendering core. Visible differences between the two displaying methods and their possible causes are then discussed.

Keywords

autostereoscopy, multiscopy, stereoscopy, display, screen, lenticular, slanted, multiview, 3D, image, interlacing, stereo pair, 2D plus depth, depth map, DIBR, software, subpixel, matlab

Abstrakt

Cílem této práce je vytvořit přehled principů lidského vnímání hloubky prostoru a jejich využití v technologiích pro stereoskopické "3D" zobrazování. Dalším cílem je vyvinutí softwaru, nebo vytvoření sady aplikací, které umožní korektní zobrazení obrazu na autostereoskopické obrazovce podporující větší množství pozorovacích úhlů bez dodatečného zpracování obrazu za pomoci specializovaného hardwaru.

Aby bylo možné zobrazit multiview obrazy správným způsobem, je zapotřebí použít prokládací vzor ke zkombinování vstupních obrazů obsahujících jednotlivé pohledy na reprodukovanou scénu. Tento prokládací vzor je závislý na parametrech vrstvy lentikulárních čoček umístěné před samotnou obrazovkou. Protože výrobce obrazovky tyto parametry v dostupných materiálech neuvádí, je potřeba získat prokládací vzor empirickou metodou. Postup získání prokládacího vzoru je v práci zevrubně popsán.

Za použití získaných dat je poté vytvořena počítačová aplikace, která umožňuje použití běžně používaných formátů pro uložení stereoskopického obrazu. Aplikace dovoluje uživateli nastavit základní parametry pro generování výstupního obrazu, který je zaveden do autostereoskopické obrazovky přímo, namísto hardwarového renderovacího jádra dodávaného spolu s obrazovkou. Zpracování obrazu, které aplikace provádí, je detailně popsáno a k práci je přiložen jednoduchý návod k použití aplikace.

Příklady obrazu vytvořeného za pomoci aplikace jsou porovnány s výstupem z původního hardwarového renderovacího jádra. Pozorované rozdíly a jejich možné příčiny jsou popsány v závěru práce.

Klíčová slova

autostereoskopie, multiskopie, stereoskopie, displej, obrazovka, lentikulární, šikmý, multiview, 3D, obraz, prokládání, stereo pár, hloubková mapa, DIBR, software, subpixel, matlab

Contents

List of used abbreviations

AC (mismatch)	(accommodation-convergence (mismatch))
HMD	(head-mounted display)
LCD	(liquid crystal display)
OLED	(organic light-emitting diode)
PDP	(plasma display panel)
LC	(liquid crystal)
DLP	(digital light processing)
FPD	(flat panel display)
RGB	(red-green-blue)
SBS	(side-by-side stereoscopic image arrangement)
OU	(over-under stereoscopic image arrangement)
H-SBS	(half side-by-side)
H-OU	(half over-under)
DIBR	(depth image based rendering)
YUV	(color space - Y=luminance, U and V=chrominance coordinates)
VOD	(video on demand)
SAM	(same-angle matrix)
SPM	(same-position matrix)
GUI	(graphical user interface)
OS	(operating system)
HDMI	(high definition media interface)
EDID	(extended display identification data)

List of figures

Introduction

Stereoscopic displays have seen a rise in popularity in recent years. While standard stereoscopic screens requiring the viewers to wear specialized glasses are commercially available, autosteroscopic screens that do not require any headgear are rare to come by, as their parameters are not yet fit for consumer use.

This thesis describes working principles of autosteroscopic display technologies, along with brief overview of common techniques used by commercial stereoscopic 3DTVs and principles of human depth perception.

The following chapters describe the process of determination of parameters of a Phillips autosteroscopic display with lenticular lens sheet, that are required in order to pre-process images for correct reproduction on the specialized screen. These parameters are not disclosed by the manufacturer of the display, so an empirical approach is employed. The used experimental methods are focused on finding usable image interlacing pattern by closely observing the displayed image and its behavior under various viewing conditions.

The subsequent chapters are focused on detailing the creation of a computer application that uses the obtained display parameters in order to generate multiscopic images viewable on the autosteroscopic screen. Inner workings of the application are described as well as the thinking behind the used workflows.

An overview of the applications user interface is then provided with description of individual elements and their effect on the results of the image generation. Some unexpected problems and their solutions are also described.

Advantages and downsides of the developed application in comparison with a commercial hardware solution are listed and their possible causes are discused.

Finally, a list of possible enhancements that would further expand the capabilities and reliability of the developed application is provided.

1. 3D perception and display technologies

This chapter serves as a theoretical introduction to some of the main topics of this thesis – depth perception, stereoscopic displays and stereoscopic image formats.

Information the human brain uses in order to create the impression of depth are described.

Later in the chapter, an overview of technologies commonly used to display stereoscopic 3D images on commercially available displays is presented, with emphasis on technologies that do not require the viewer to wear any additional headgear.

Several common stereoscopic 3D image formats are also shortly described and their advantages and disadvantages are listed.

1.1. Human 3D perception

The human visual system uses many depth cues to determine relative position of an object in 3D space. These depth cues can be divided into two categories: physiological, based on our visual system anatomy, and psychological, which is based on our understanding of the world [1]. Both are discussed in following sections.

1.2. Physiological depth cues

Usually the distance between eyes for an adult is between 60 and 70 mm [2]. This is called interocular or interpupillary distance. As one eye is only capable of perceiving a planar image, in order to perceive a 3D scene, cooperation of both eyes is required.

This binocular viewing provides the perception of depth and is used as basis of function of most stereoscopic displays.

1.2.1. Binocular Disparity

Binocular disparity is the difference in the images projected on the left and right eye retinas in the viewing of a 3D scene. The images that the eyes receive from the same object are different according to the different locations of the eyes [1].

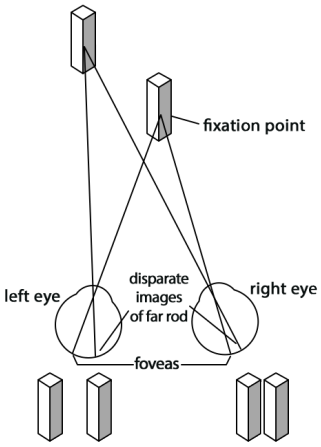


Fig. 1 Binocular disparity. [3]

The two images, along with other information discussed later, are then fused in the brain into a three-dimensional image. A process which is not yet fully understood [1].

Binocular disparity is the most important depth cue used by the visual system to produce the sensation of depth, or stereopsis. [4]

1.2.2. Vergence

Vergence is a type of simultaneous eye movement in opposite directions done by extrinsic muscles, which helps to obtain or maintain single binocular vision.

Convergence is the simultaneous inward rotation of the eyes towards each other, usually in an effort to maintain single binocular vision when viewing an object as it moves closer to the observer. Exaggerated convergence is called cross eyed viewing.

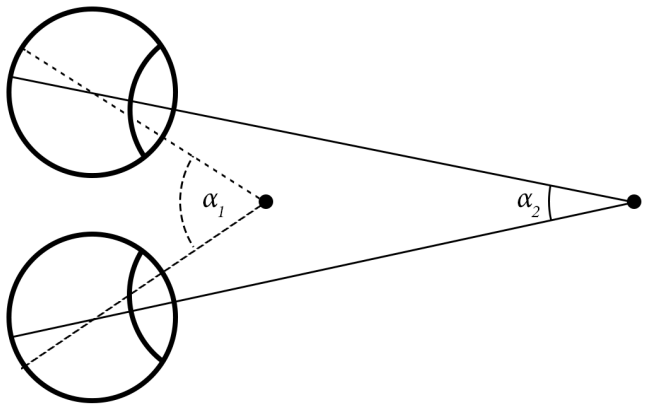


Fig. 2 Convergence on close and distant object.

Divergence is the exact opposite of convergence - outward rotation of the eyes away from each other. When an observed object is moving further away from the observer, the eyes diverge until they are parallel, basically fixating at the same point at infinity. [5]

Both convergence and divergence have certain limits which, when exceeded, can cause discomfort or even pain to the observer. Especially higher-than-parallel divergence is crucial to avoid in production of stereoscopic footage.

1.2.3. Accomodation

Accommodation is the change in optical power of the eye as it focuses on different distances in a 3D scene. The lens changes thickness (and effectively its focal length) due to a change in tension from the ciliary muscle.

This depth cue is normally used by the visual system in tandem with convergence. [4]

1.2.4. Accommodation and convergence mismatch

In stereoscopic and autostereoscopic displays the two different views of an object are presented next to each other on a planar surface of a display. That causes a problem, as the eyes accommodate on the plane of the display surface while the projected disparity stimulates a different depth perception.

This is called accommodation-convergence mismatch (or simply AC mismatch), since the eyes converge at the apparent point of fixation in the image but focus on the screen. [3]

As both disparity and accommodation convey depth information which might be contradictory, viewers may feel discomfort, manifested by eyestrain, blurred vision, or headache. [1] Small percentage of population is not even capable of fusing the images in one 3D scene.

1.2.5. Pseudoscopy

An undesirable effect called pseudoscopic viewing is another topic related to binocular viewing and stereoscopic displays. This effect occurs when images intended for individual eyes are swapped and perceived depth is thus inverted (far objects appear as close and vice versa). In combination with conflict with psychological depth cues mentioned in the following section, pseudoscopic viewing induces viewer confusion and discomfort.

1.3. Psychological depth cues

These monocular information about relative depth are based on our experience with the outside world gained as part of growing up. They usually work in tandem with physiological depth cues, but are also used to give the illusion of depth where none is present (printed pictures, 2D displays etc.).

Misalignment of physiological and psychological depth cues can induce confusion and discomfort of the viewer and is highly undesirable in stereoscopy.

1.3.1. Perceived size / Linear perspective

Linear perspective refers to the change in image size of an object on the retina in inverse proportion to the object's change in distance. As an object moves further away, its image becomes smaller. This effect is called perspective foreshortening. [4]

For example, we know from experience that an elephant is larger than human, so when we see a picture of an elephant smaller than a human figure, we assume that the animal is further in the distance.

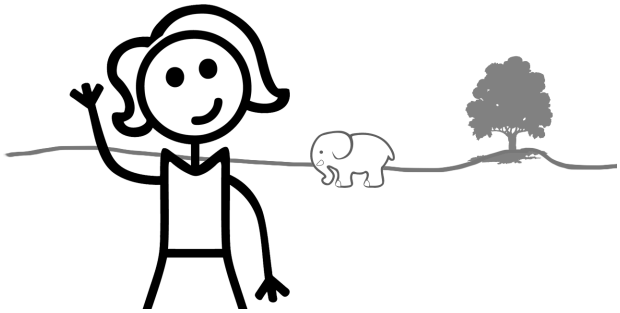


Fig. 3 Example of perceived size.

A 3D perspective can also be created by two parallel lines, intersecting at infinity (i.e. rails of a train track). [1]

1.3.2. Interposition

One of the strongest depth cues. One object overlapping, hiding or occluding another gives us information about their relative position (hidden object is further away). Can cause serious confusion when violated, especially when viewing stereoscopic footage, where depth is mainly conveyed by disparity.

In the previous example, the human would naturally occlude the small elephant in the distance. In case it is the other way around, we tend to assume there is a tiny elephant hovering in front of the human.

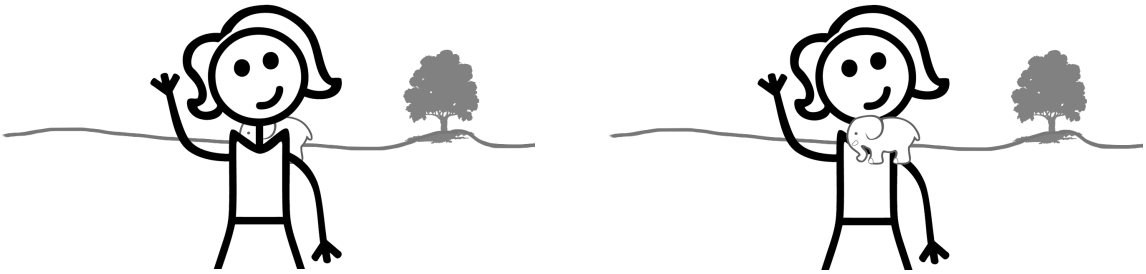


Fig. 4 Right and wrong occlusion of objects in an image.

1.3.3. Atmospheric occlusion

Distant objects tend to show less contrast and slight blue tint due to light traveling through thicker layer of atmosphere which is not 100% transparent. Blue, having a shorter wavelength, penetrates the atmosphere more easily than other colors. [4]

Some weather conditions also affect appearance of distant object e.g. rain or fog.

1.3.4. Motion parallax

Motion parallax is similar to binocular disparity in that it provides different views of a scene from multiple angles. The different views are achieved by side to side movement of either the viewer's head (or camera) or the scene itself. The relative position of object in a scene is then determined by their relative movement - the closer the object the more it appears to move when changing the viewer's position.

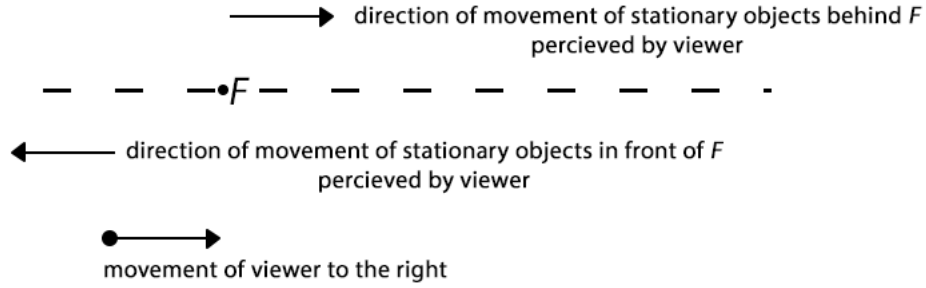


Fig. 5 Motion parallax. [1]

If a viewer is moving to the right with eyes fixed on the stationary point F, then the stationary objects behind F are perceived as moving in the same direction to the right as the viewer, while those in front of F are perceived as moving to the left. This motion parallax is part of the intuitive experience of the viewer and provides the locomotive viewer with depth information relative to F [1].

1.4. Stereoscopic displays

Any kind of display technology capable of delivering different images to each eye of the viewer can be considered stereoscopic display. Most commercially available direct-view stereoscopic devices (3DTV) make use of special glasses worn by the viewer to allow pass-through of only the correct image to each eye and thus creating a 3D sensation. Wavelength-multiplexed, polarization-multiplexed and time-multiplexed methods are the most used methods in the entertainment industry.

Other devices, such as head mounted displays (HMDs) are capable of delivering stereoscopic images with greater immersion due to various head movement tracking techniques used to control the projected image.

Autostereoscopic displays can be considered a subset of stereoscopic devices, but since the main topic of this work is autostereoscopy, these displays will be introduced in more detail in section 1.5.

1.4.1. Wavelength multiplexed methods

The anaglyph method can be considered the most popular low-cost solution of displaying 3D stereoscopic images. It uses cheap (often paper) glasses with complimentary color filters (usually red for the left eye and cyan for the right eye [6]) and does not require any special hardware, as any common color video equipment is sufficient. The main disadvantages of this method are the loss of color information and the high level of crosstalk (bleeding of left image into right eye and vice versa).

A wavelength multiplex solution with high color accuracy is called Infitec, which stands for *In*terference *fi*lter *tec*hnology and is owned by German company Infitec GmbH. Similarly to the anaglyph method, it uses color filters to direct left and right images to left and right eye accordingly. However where anaglyph filters can be seen as high-pass and low-pass filters, in the case of Infitec, selective narrowband filters at wavelengths representing R,G and B primary colors are used with slight difference in transmitted wavelengths for each eye as shown in Figure 6b .

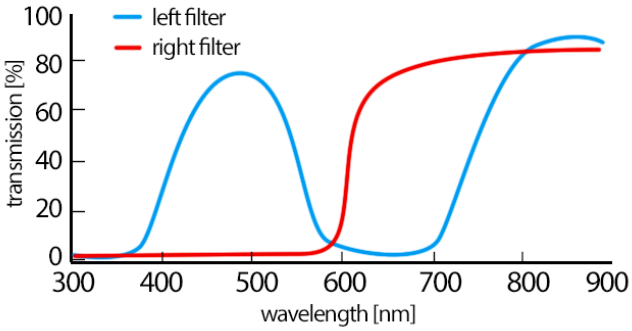


Fig. 6a) Transmission spectra of anaglyph glasses.

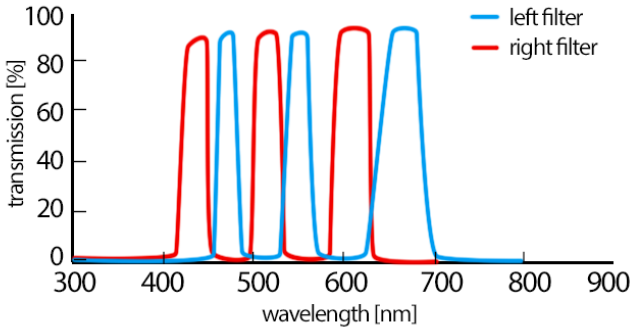


Fig. 6b) Transmission spectra of Infitec filters.

Infitec technology is currently widely used in projection displays, such as cinemas and projectors for personal use as it requires two light sources with differently filtered spectrum. Direct-view LCD display using Infitec filter glasses and time-multiplexed, filtered LCD backlight was demonstrated in [7]. The downside of this solution is low brightness of the image due to inefficient utilization of source light spectrum by the interference filters.

1.4.2. Polarization-multiplexed displays

Polarization-multiplexing has its roots in cinema projection where companies such as IMAX employed two aligned film projectors (or, in recent years, digital cinema projectors) each with linear polarization filter perpendicular to the other. The viewer would use eyewear with appropriate polarizers to block the image not intended for that eye.

Another method is using only one projector with electrically controllable polarization rotator synchronized to displayed frames [6] similar to color wheel in DLP projectors. This technology does not require any alignment of images and is used in cinema projectors from MasterImage or RealD, both using circular polarization instead of linear, which allows the viewer to tilt head without any negative effect to the image (crosstalk).

Common to both of these methods is the requirement of special non-depolarizing screen (with silver coating), which can prove to be too expensive for smaller cinemas (Infitec technology is then preferred).

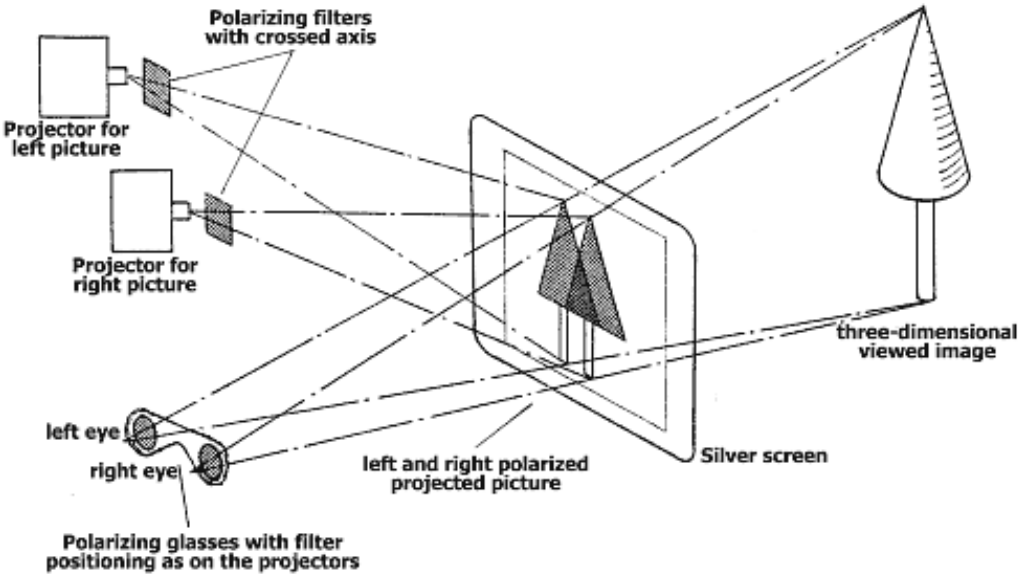


Fig. 7 Two projector setup for linearly polarized stereoscopic projection. [8]

For the direct-view flat panel displays, available technologies are micropolarizer based, patterned retarder based, and dual-panel type. In the case of micropolarizer and patterned retarders alternate horizontal pixel rows are orthogonally (circularly) polarized by the line-interleaved micropolarizers or patterned retarders attached to the display [6]. Odd horizontal lines are reserved for one eye, while even lines are used by the other eye, effectively sacrificing half of the vertical resolution in order to stimulate depth perception. Stereo pair images are then displayed in a horizontally interleaved format and low-cost passive polarizer glasses (same as RealD and MasterImage ones) worn by the viewer separate the two images. Because of the two images are interleaved instead of overlapped, this method is also called spatial or area-multiplexing.

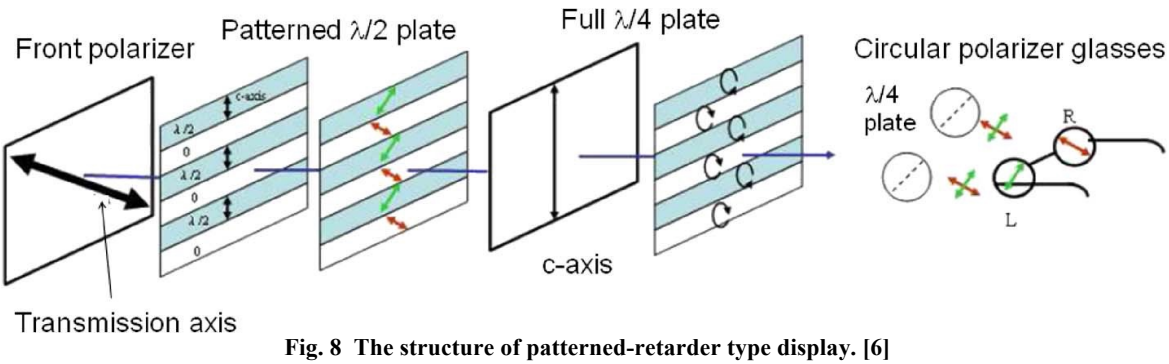


Fig. 8 The structure of patterned-retarder type display. [6]

Dual-panel based displays (or dual layer 3D LCD) use each pixel twofold by simultaneously transmitting the luminance and color for the right eye and the left eye view while preserving full display resolution for both eyes. This is achieved by placing two LCDs on top of each other, where the rear one, backlighted by unpolarized light, displays both stereo pair images combined as an ordinary non-stereoscopic LCD would. The second LCD, placed directly on top of the first one then changes polarization of the underlying polarized image on a per pixel (sub-pixel) basis. The change of polarization is dependent on the brightness ratio in left and right image and is controlled by precise pixel voltages. Polarization glasses, same as in the methods above with orthogonal polarization (both linearly and circularly polarized can be used), then transmit the right amount of incoming light based on relation of the polarization filter to polarization of the light. [1]

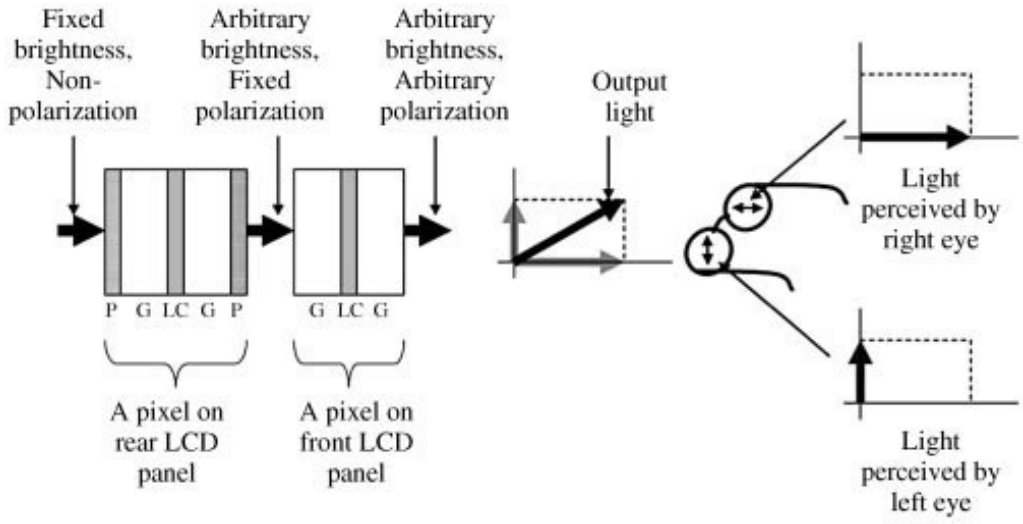


Fig. 9 Dual layer stereoscopic LCD principle. [1]

1.4.3. Time-Mutliplexed (active) displays

As opposed to time-parallel stereoscopic displays (i.e. polarization multiplexed), screens using time-multiplexed method rely on the persistence of vision of the human visual system. Left and right-eye-images in their full resolution are flashed on the screen in an alternating fashion at high frame rates, while viewer-worn shutter glasses synchronized with the screen refresh rate block light for the eye the current frame is not intended for.

A value of 120 Hz is often cited as the lowest frequency required to successfully display time-multiplexed stereoscopic images with no flickering visible to the user. According to Ferry-Porter law, the critical flicker frequency (c_{ff}) for a typical 200 cd/m^2 screen is around 60 Hz [9] which is just below the recommended frequency of 60 Hz (per one eye). However, as human peripheral vision has higher c_{ff}, some visible flicker of ambient light might be observed in a brightly lit room. Hence, higher frequencies are recommended.

Any high-frequency screen (or projector) is capable of delivering time-multiplexed stereoscopic image when equipped with a unit providing synchronization signal (in most cases infrared) for the shutter glasses. This synchronization unit can be either built right into the screen, or exist as a separate device that is connected to the display. Viewing angles are limited by reception of the synchronization signal.

The active glasses use fast liquid-crystal based shutters that block the left-eye images from reaching the right eye of the viewer and vice versa. One obvious disadvantage compared to passive polarization glasses is the need for a battery providing power for electronics that take care of receiving the signal and controlling the shutters. Active glasses are therefore bulkier, heavier and more expensive than their passive counterpart.

Cross-talk might occur with imperfect synchronization or slower pixel response. Older LCD screens do not change the brightness of a pixel fast enough to show the correct color/brightness in the subsequent frame, retaining some portion of the current. Other technologies are better in that respect. Black frame insertions in higher refresh rate screens are also used to reduce crosstalk by allowing the shutters to switch states without any image bleeding.

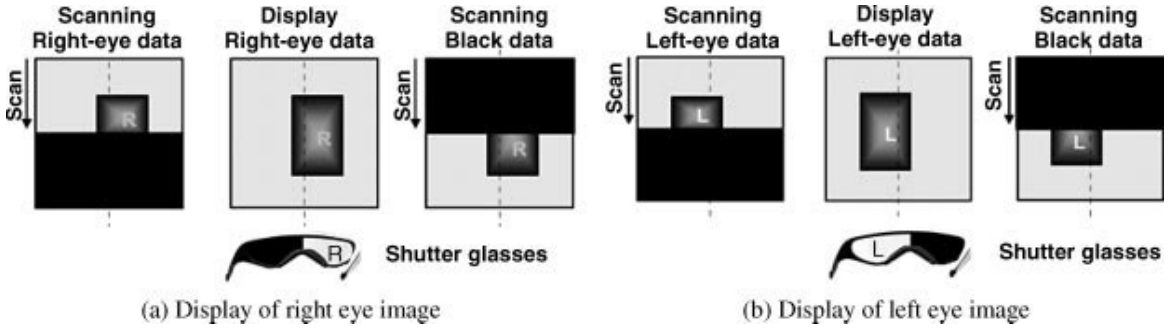


Fig. 10 Time sequential viewing of the two images with shutter glasses. [1]

1.5. Autostereoscopic displays

Autostereoscopic displays are special subset of stereoscopic devices which don't require any glasses or other user-mounted-devices in order to produce the required disparity and stereoscopic 3D sensation.

Direction of light from the screen is controlled by special optical element, usually placed in front the screen surface, establishing multiple viewing zones with different images in front of the display. Large Fresnel lenses, lenticular arrays, parallax barriers, or other components such as mirrors, micropolarizers, and prisms [6] can be used as such light directors, with parallax barriers and lenticular lenses being the most common.

Autostereoscopic displays can be further divided based on number of views of the displayed scene to either two-view or multiview display.

Two-view systems produce a single pair of parallax views formed at a single location in space or repeated in multiple zones in front of the display, allowing for simultaneous use by multiple viewers. The viewer has to be in the correct location (distance and angle) to perceive a stereoscopic image.

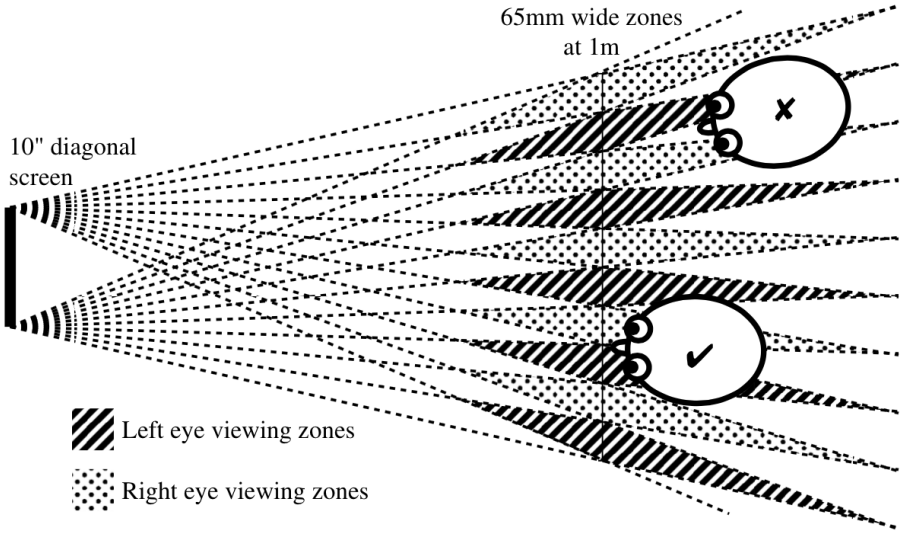


Fig. 11 Multiple viewing zones in front of two-view autostereoscopic screen, depicting correct and incorrect (pseudoscopic) positioning of the viewer. [10]

In multiview systems, multiple different stereo pairs are presented across the viewing field enabling the viewer to move his or her head and look at the scene from multiple angles. The number of views in multiview displays is normally insufficient for smooth motion parallax, and with increasing number of views, resolution of the image is usually decreased. Supermultiview systems with high amount of displayed images are researched, however current display technologies do not allow mass production of such device (low resolution) [6].

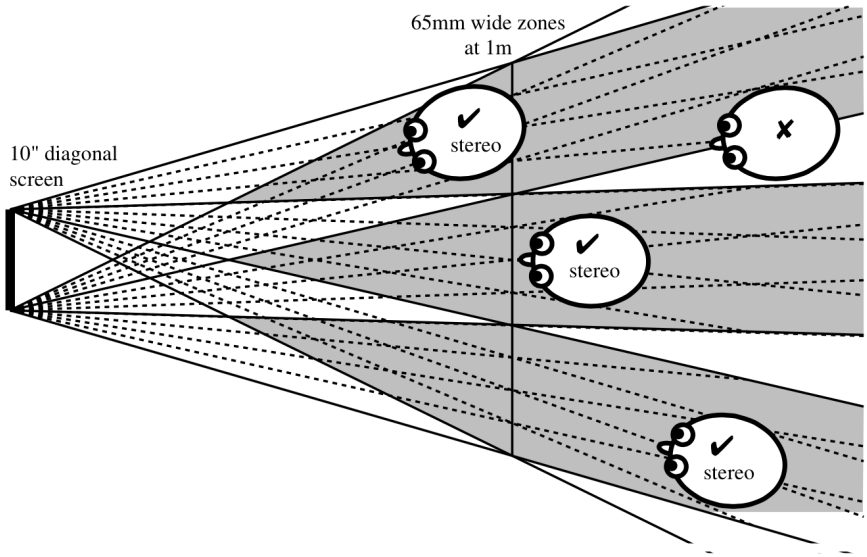


Fig. 12 Smaller number of wider viewing zones in front of multiview autostereoscopic screen. [10]

Head or eye tracking is another technique used to provide continuous motion parallax and suppress pseudoscopic viewing. A two-view system with adaptive optical element (physical movement of e.g. parallax barrier or lenticular lens array) and head-tracking system is capable of shifting the viewing zone according to viewer's movement in front of the screen. Such system requires special format of image data for generating smooth change of viewing angle e.g. high amount of discrete views or real-time rendered computer 3D models.

Head-tracking can also be used to eradicate pseudoscopic images in two-view autostereoscopic displays with multiple fixed viewing zones by switching left and right eye images according to position of viewers eyes in the viewing field.

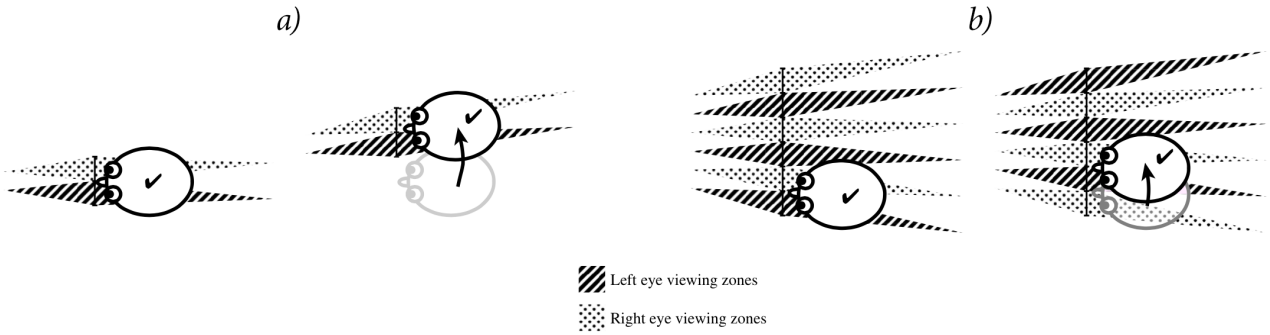


Fig. 13 Uses of head-tracking for pseudoscopic viewing suppression employing a) shifting of viewing zones b) image swapping. [10]

1.5.1. Parallax barrier displays

Autostereoscopic displays employing parallax barriers use spatially multiplexed left and right eye images in different columns of the flat panel display (FPD - an LCD, OLED or PDP), meaning the horizontal resolution is halved for each eye. A sequence of light blocking barriers and light transmitting slits sits in front of the FPD, guiding light from the left and right eye images to the corresponding eyes of the viewer. As depicted in Figure 14, light emitted by the left eye image is transmitted by the slits and is narrowed to a point at the position of the viewer's left eye, while light from the right eye image heading in that direction is blocked by the barrier. This leads to a loss of brightness, as half of the emitted light is blocked. Unlike for lenticular-based approach, the distribution of luminance is uniform [1].

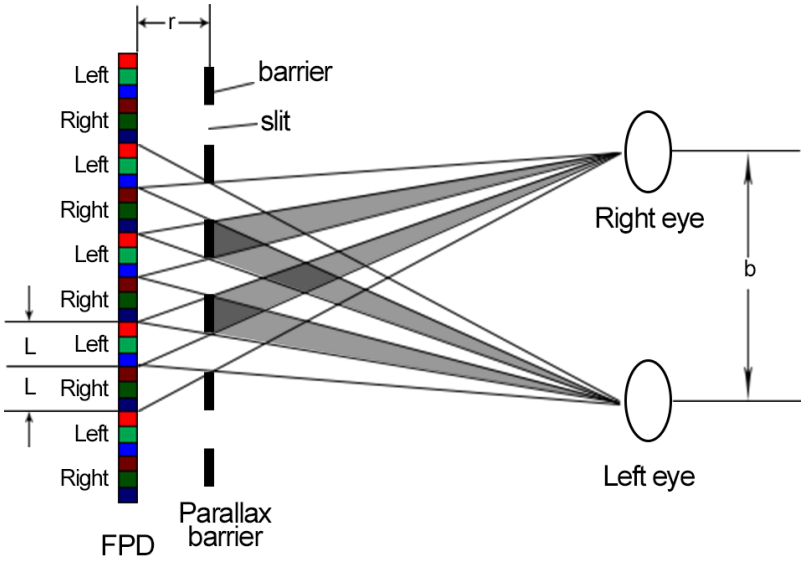


Fig. 14 The parallax barrier on an FPD with the right eye and left eye images. Based on [1].

Viewing distance z of the parallax barrier based autostereoscopic display depends on the width L of columns on the FPD, the distance r of the barrier from the FPD and on interocular distance b (distance from the center of one eye to the other, as a rule 65 mm), as described by equation (1).

$$z \sim \frac{r \cdot b}{L} \tag{1}$$

As L is given for any particular display, the viewing distance (image plane) can be controlled by the distance of the barrier from the FPD plane. Difference in interocular distance (i.e. small kids) also affects the distance of the image plane.

The barrier itself can be either physical (made of plastic or other material) or realized by liquid crystal technology, as is the case with most modern parallax barrier based screens. An LC based approach has the advantage that the barrier can be turned off and the screen can be used as regular 2D display. Voltage controlled LC shutter also allows to change the orientation of the screen, as the barrier can be oriented either vertically or horizontally. The only downside of liquid crystal approach is the addition of two glass layers which makes the display both thicker and heavier.

A time-multiplexed, LC based parallax barrier display is a potential solution to the loss of resolution common with autostereoscopic displays. In this solution, both the FPD images and the barrier are synchronously switched. In one timeframe, the left and right images and barrier are in the same state as in Figure 14. In the next time slot, columns with left eye images are replaced with columns with right eye images and vice versa, while the parallax barrier is inverted so that previously opaque barriers become transparent and transparent slits become opaque. This way, both eyes see the full resolution of the display, but as with the active-glass based time-multiplexed stereoscopic screen, a higher refresh rate display is required [1].

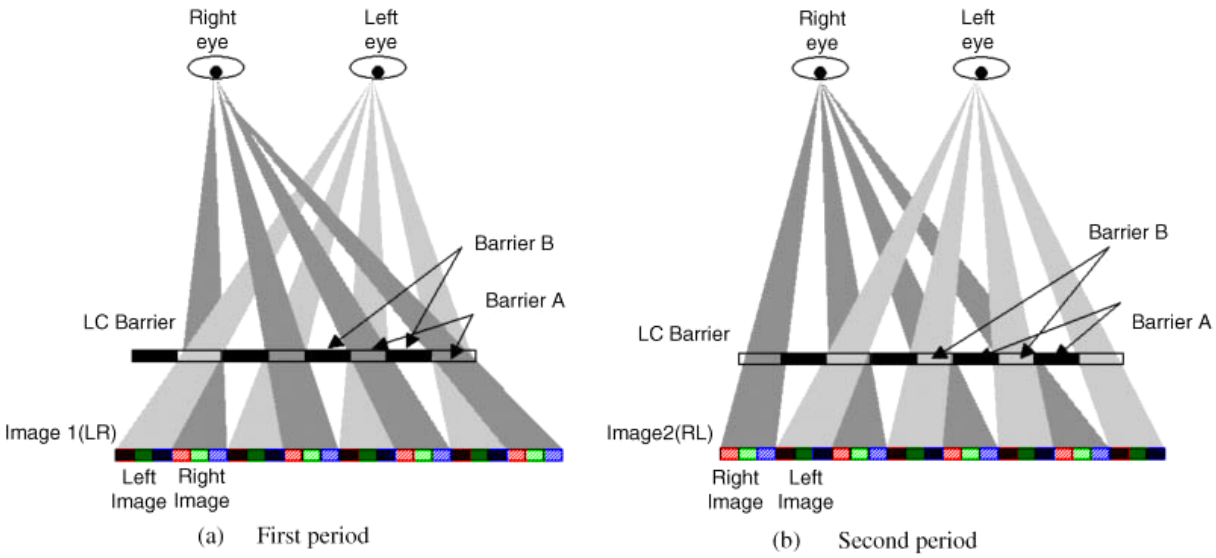


Fig. 15 Time-multiplexed switchable parallax barriers and slits (a) the transparent slits (barriers A) for the first half of the image for the left and right eye; (b) the transparent slits (barriers B) for the second half of the image for the left and right eye. [1]

1.5.2. Lenticular lens array displays

Instead of light-blocking barriers, displays with lenticular lenses (or simply lenticulars) make use of cylindrical lenses that project images from the FPD screen into repeating viewing zones on an image plane. A left and right eye image is located behind each lens, consisting of only one pixel or of a few pixels, depending on the size of the lens. In case of multiview display more than two images are located behind each lens. In that case, the views are divided into several viewing sectors as depicted in Figure 12.

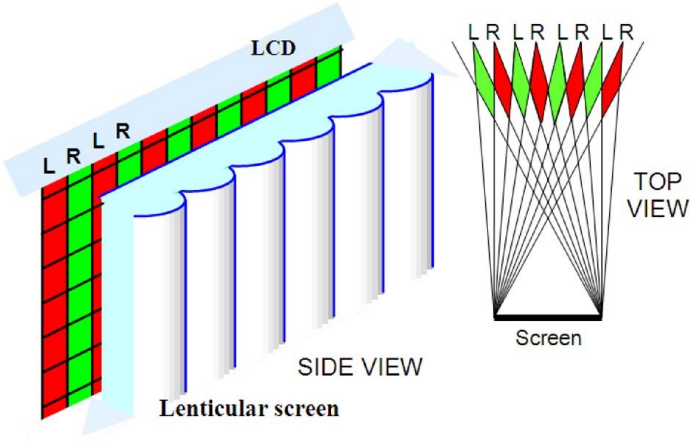


Fig. 16 Working principle of a lenticular screen. The left and right eye images are repeated along the image plane. [6]

In the previous image, the L and R image columns are in an order that would give a pseudoscopic image. Based on geometric optics and depicted by rays in Figure 17, the left and right eye images on the FPD have to be swapped to obtain the desired stereoscopic image in the image plane.

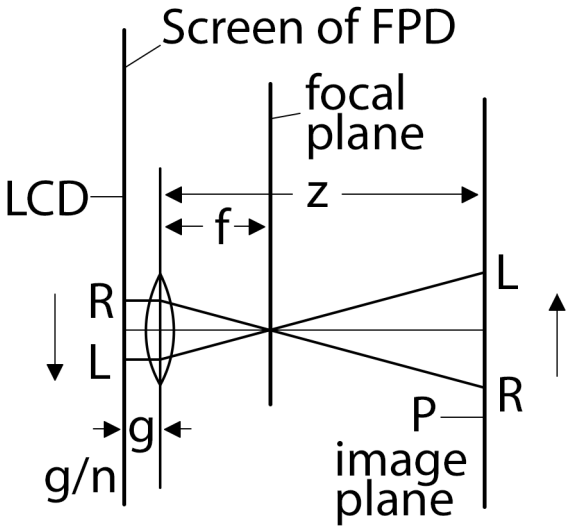


Fig. 17 Projection of a single image pair by a lenticular lens onto an image plane. [1]

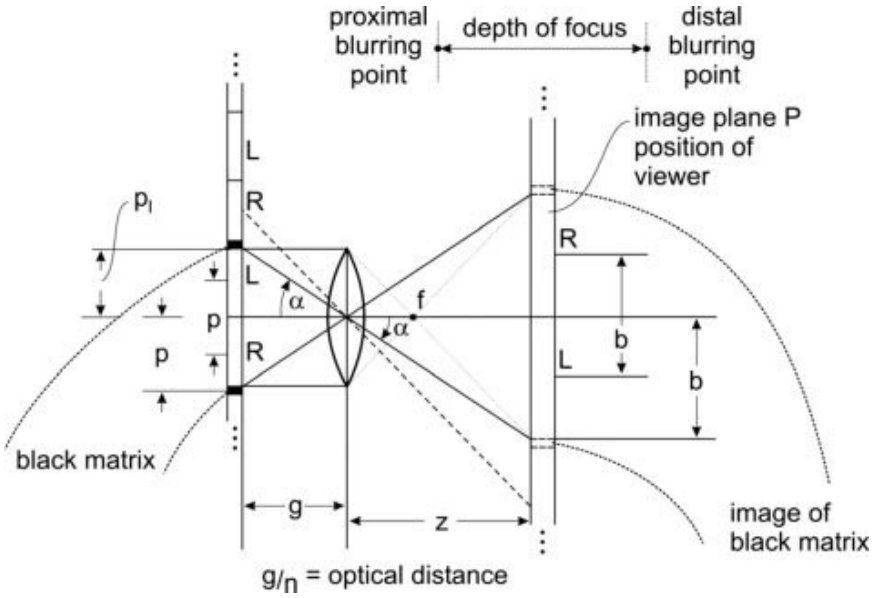


Fig. 18 Projection and magnification of the pixel pitch p into the image pitch b, the interocular distance in the image plane P. [1]

With help of Figures 17 and 18, design rules for lenticulars can be derived. Focal length *f* of a lens in the lens array, the distance *g* of the lens from the screen, the refractive index *n* and the image plane distance *z* are shown. From these parameters, the lens equation can be assembled

$$\frac{1}{g/_n} + \frac{1}{z} = \frac{1}{f} \tag{2}$$

and equations for focal length and viewing distance can be derived

$$f = \frac{z \cdot g/n}{z + g/n} \tag{3}$$

$$z = \frac{f \cdot g/n}{g/n - f} \tag{4}$$

Figure 18 also clearly depicts magnification of the pixel pitch *p* into a larger image pitch *b* on the image plane, corresponding with the interocular distance, which, on average, is 65 mm as mentioned earlier. This magnification is given by equation (5).

$$m = \frac{b}{p} = \frac{z}{g/n} \tag{5}$$

Based on these equations, the given pixel pitch of each of the two images and given interocular distance (65 mm), the magnification *m* is found. g/n is given by the FDP design, so that leaves focal length *f* and viewing distance *z* to be determined. As both parameters depend on each other, viewing distance *z* is defined and according *f* is found [1].

Black matrix depicted in Figure 18 presents a problem, as it is also magnified by *m* at the viewing distance and is very disturbing for the viewer. This effect can be suppressed by the use of smaller pixel pitch p1 as depicted in Fig. 18. However, sideward movement of the viewer in the image plane causes the diminished black matrix to become noticeable. A possible solution is the use of slanted pixels in combination with vertically arranged lenticulars, which ensures there are no black lines parallel with the lenticulars. More frequently used is vertical pixels with slanted lenticulars, which has the same effect and is shown in Figure 19.

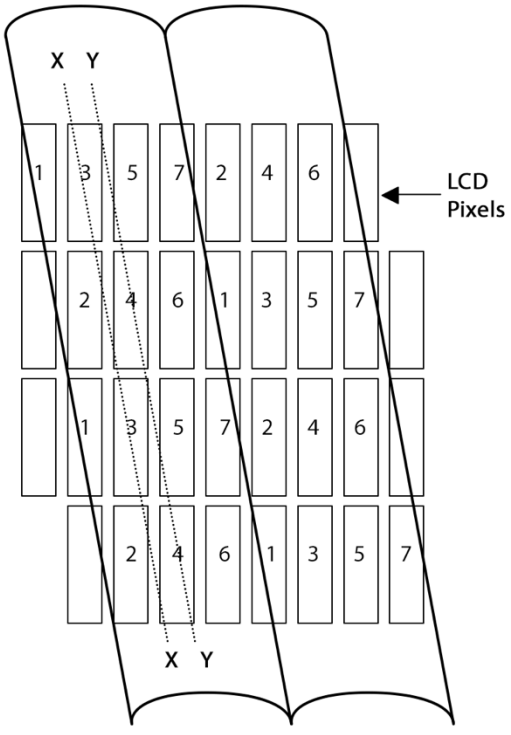


Fig. 19 Vertically aligned pixels covered by a slanted lenticular array in a seven-view display. [6]

An arrangement such as shown in Figure 19 has several other advantages over simple pixel/lens parallel arrangement. The loss of resolution is distributed to both horizontal and vertical direction as opposed to full resolution in vertical direction and more severe loss in horizontal direction when traditional arrangement is used. This leads to the ability to project more views of the scene with roughly the same loss of resolution - Figure 19 depicts a 7 view arrangement with lens width of roughly 3.5 pixels.

Transition between two neighboring views is made smoother with slanted pixels or slanted lenticulars. Instead of flipping into a new image, the „old“ pixel(s) fade away while the „new“ one simultaneously fades in, resulting in perception of an increased resolution and more pleasing viewing experience overall [1]. Slanting also helps to eliminate moiré pattern.

The main disadvantage of slanted design is more complicated arrangement of the image matrix displayed on the FPD, evident from Figures 19 and 20.

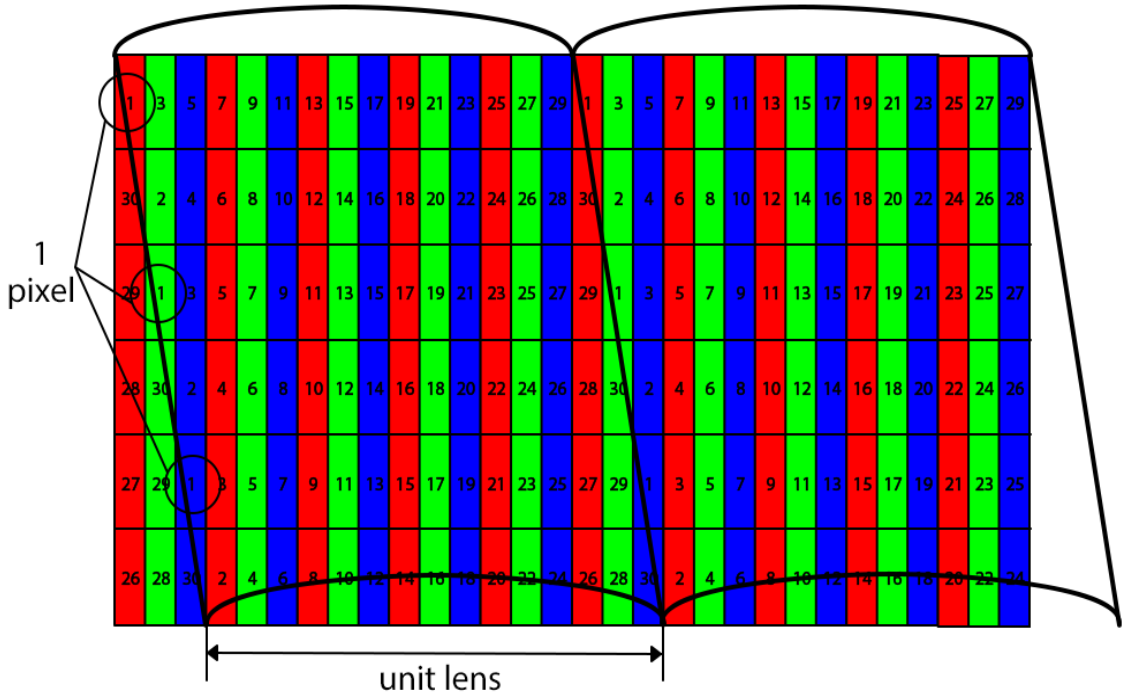


Fig. 20 Arrangement of the RGB pixel triplets for a 30-view display with a slanted lenticular lens array. [1]

In 1999, multi view pixel mapping algorithm was put forward for different LCD panels with respective slanted angle, lenticular pitch and offset [11].

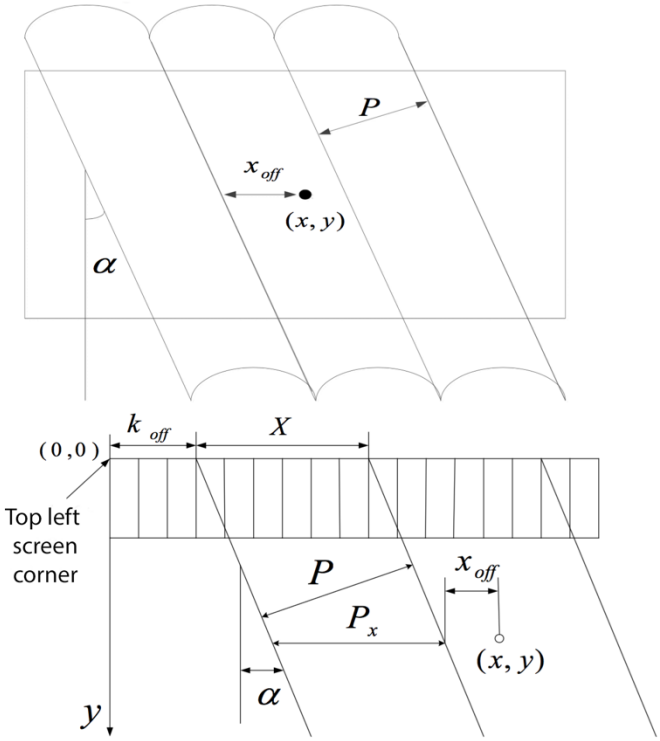


Fig. 21 Multiview pixel mapping and its parameters [11].

The refraction angle is related to x_{off} , if x_{off} is the same, the light will be refracted to the same direction. So the aim of sub-pixel mapping is to ensure that the light refracted to a certain angle comes from sub-pixels which have the same view number. We can easily acquire the relations between the parameters of autostereoscopic 3D system [11].

$$\frac{N}{N_{tot}} = \frac{x_{off}}{P}$$

$$x_{off} = \left(x - k_{off} - y\tan\alpha\right) mod P_x$$

$$P_x = \frac{P}{\cos\alpha} \tag{6}$$

Where N denotes the view number of a certain viewpoint, N_{tot} denotes the number of viewpoints, x denotes the distance from the point to the left edge of the grating line above, P denotes the width of a single line of grating, P_x denotes the width of a single grating in the horizontal direction [11].

With (6), each sub-pixel on the LCD can be mapped to a certain viewpoint. If x and y denote the horizontal and vertical coordinates for each sub-pixel, then we get:

$$N = \frac{\left(x - k_{off} - y\tan\alpha\right) mod P_x}{P_x} N_{tot} \tag{7}$$

Without loss of generality, assume the grating slants to the right. Set the top left corner of the TV screen as the origin, horizontal right as the positive direction of the x-axis, straight down as the positive direction of the y-axis to establish a coordinate. Each sub-pixel is represented by its view number in the coordinate.

$$N = \frac{(x_s - k_s - 3y_s \tan\alpha)\ mod X}{X} N_{tot} \tag{8}$$

Where x_s , k_s and y_s respectively denotes sub-pixel's horizontal, vertical and offset length, X denotes the length of sub-pixels covered by a single line of grating [11].

A slanting angle is commonly defined by a straight line from the upper left corner of a pixel to the lower left corner of the pixel just bellow and to the right as illustrated by dotted line X in Figure 19. This leads to a slant angle

$$\alpha = \arctan\frac{1}{6} \tag{9}$$

with α = 9.46°, when assuming the length of the pixels is three times their width [1].

Since the lenticulars do not block light as parallax barrier does, but focus it to the viewing zones, all the light emitted by the FPD is transmitted by the lens array to the viewer. The overall brightness of the screen is therefore twice as high as with the use of parallax barrier.

As with LC-based parallax barrier, use of special lenticular lens array allows to switch the screen between 2D and 3D modes. Switchable lenses are, again, realized by LCs where the molecules, when rotated, change their optical properties. The lenticulars are filled with liquid crystal molecules which change their refractive index when voltage is applied. The new refractive index then matches the one of the replica (shown in Figure 22) which effectively makes the two layers appear as a single, flat, non-refractive layer, allowing light to pass without any change in direction. This is the common optical setup for 2D presentation.

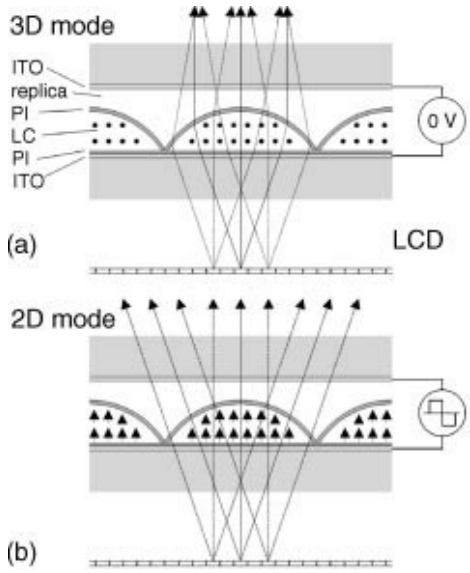


Fig. 22 Switchable lenticular lens (a) working in the 3D mode (b) working in the 2D mode. (ITO- Indium-Tin-Oxide) [1]

1.6. Stereoscopic 3D image formats

Stereoscopic "3D" images can be represented in a few different formats, each with its advantages and flaws. This chapter will briefly describe some of the main formats used by modern stereoscopic displays (excluding color multiplexing etc.).

1.6.1 Stereo pair images

As the name suggests, the most basic stereoscopic image format consists of two images – one destined for the left eye and one for the right eye. It is the most widely used stereoscopic format used, as all "3D" movies use it, be it in cinemas, on Blu-Rays and in other distributions (VOD).

The two images are either saved as separate 2D files (using twice the data) or can be multiplexed into one file with some compression added, getting as low as 1.5x the size of 2D image/video.

For motion pictures a time-sequential multiplexing is often used, alternating L and R images with each frame (same principle as described in chapter 1.4.3.).

For still images a spatial multiplexing approach is the only possibility, although it can be used in video as well. The most used arrangements are side-by-side (SBS) and over-under (OU), where the two images are placed next to each other in one file in a horizontal or vertical manner, respectively. In cases where bandwidth is limited (television broadcast, internet streaming), these two arrangements are resized so they occupy the same image area as a 2D image – these arrangements are sometimes referred to as half-side-by-side (H-SBS) and half-over-under (H-OU) as their horizontal/vertical resolution is effectively halved. Other, less used, spatial multiplexing method is interleaving the two images.

Stereo image pair is the easiest to create, as it only requires two cameras (or one camera used in 2 positions sequentially) for image acquisition. The same applies for CG production where a virtual "stereo rig" can be created with minimum effort.

The stereo image pair format has good image reproduction quality as it only consists of real images captured by the camera. It does not have problems with occlusions, light refraction etc. The main disadvantage of stereo pair is its inability to provide motion parallax.

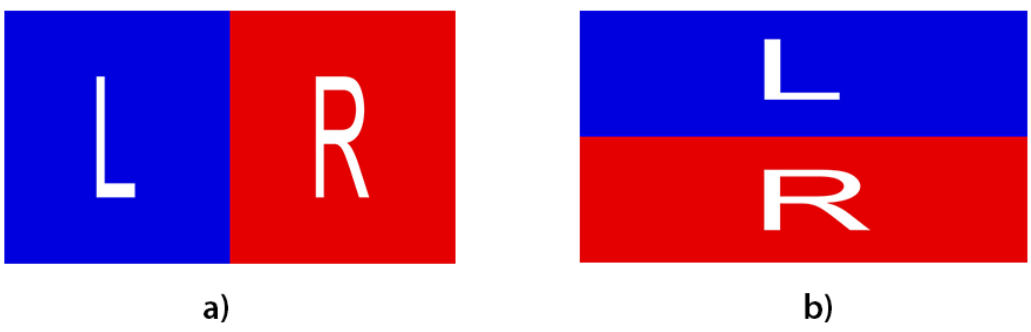


Fig. 23 a) H-SBS stereo image multiplexing, b) H-OU stereo image multiplexing.

1.6.2. Multi-view images

An extension of stereo pair with more images from different angles. It is the ideal format for mutliview displays as it has most of the positive properties of stereo pair but allows for motion parallax by switching between multiple pairs.

The image set can either allow for horizontal motion parallax, created a camera array with cameras offset in one axis, or both horizontal and vertical motion parallax in case of image acquisition by a two dimensional matrix of cameras. For the second option an even more specialized lens array is needed so in this work, only the horizontally offset images will be used.

The main downside of sets of multiview images is that they are quite demanding on storage space in order to be saved in full resolution. Spatial multiplexing is possible where severe resolution loss is acceptable – generally, the more views in a multiplexed image, the lower the resolution of each of the individual images.

Fig. 24 A spatially multiplexed 8-view image in a 3x3 matrix. Image downloaded as part of [12]

1.6.3. 2D-plus-depth format

2D-plus-depth or 2D+Z as the format is also called is a stereoscopic video coding format that uses a grayscale depth map that supplements each 2D image. Each pixel of the 2D image has corresponding pixel on the depth map that carries information about its position on the Z axis. Lighter shades of gray usually represent objects closer to the viewer and darker parts of image represent objects more in the distance, however in some cases the grayscale map is inverted. 256 depth planes can be encoded in 8-bit grayscale image, which is usually sufficient to create smooth gradient of depth within the image.

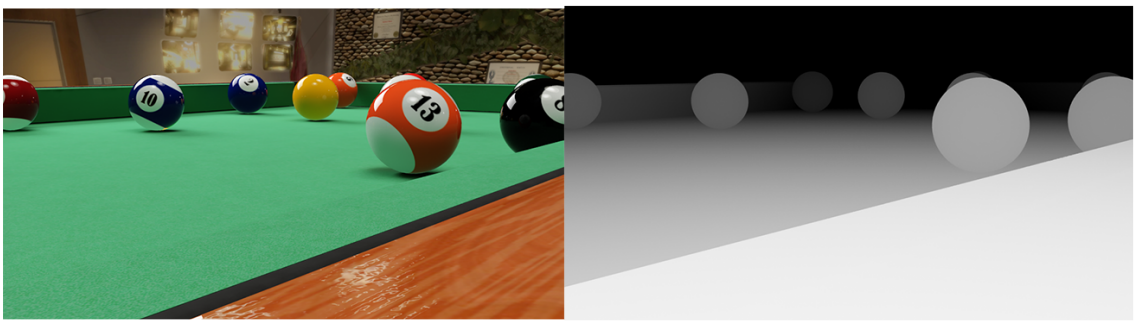


Fig. 25 An example of 2D-plus-depth image with the actual image on the left and its grayscale depthmap on the right.

This format is often used in postproduction of digitally created images – depth of field, atmospheric occlusion and other effects which would take long time to render directly are easily achievable with depth map.

2D to 3D conversion is another possible use case for the format as it is possible to render a stereo pair or multiview image set with arbitrary number of views using a depth image based rendering (DIBR) algorithm [13].

Most 3D graphics software suites (3DS Max, Maya, Cinema 4D etc.) are capable of saving the depth map along with the base RGB image. Creation of depth map for non-computer generated images is fairly complicated and sometimes does not give good enough results. Algorithms for automatic and semi-automatic computation the depth map from two or more images are heavily researched [14]. For cases where only one image is available, manual creation of depth map is a viable option.

The main disadvantage of the 2D-plus-depth format is its inability to contain information about objects occluded by foreground elements. Texture for occluded objects is usually computed from neighboring pixels by the DIBR algorithm, resulting in smeared edges around foreground objects. Another disadvantage of the format is with its handling of transparent refractive materials and reflections, which do not work properly and are only approximated by a static texture on the supposedly refractive/reflective object.

Advantages of the format include its ability to supply virtually infinite number of views and small file size of the depth map as only 8-bit image is required. 2D+Z is also compatible with 2D screens as they can easily ignore the depth map and display only the 2D image.

2. Determination of correct interlacing pattern

In order to successfully display multiview images on the available autostereoscopic screen, it is necessary to find out how to interlace the images so that each of them is refracted by the lens array to a corresponding viewing zone. The interlacing pattern (or sub-pixel mapping scheme) can be obtained using several different methods, both analytical and empirical, some of which are discussed and used in this chapter.

2.1. Used screen parameters

The subject of study of this work is an autostereoscopic 3D screen with slanted lenticular lens array. The screen is made by Dutch technology company, Philips, and the model is BDL4251VS. Some of the most important parameters of the display given by the manufacturer are listed in Table 1 [15].

The screen is equipped with a rendering core manufactured by Dimenco (a company founded by former Philips engineers). It takes care of real-time image processing and rendering of the required multiview image matrix. The native image format for the rendering core is 2D + depth. The supplied software for MS Windows computers is able to convert other stereo formats into 2D+depth for the core to work with.

Table 1. Parameters of the used screen

Diagonal screen size	42“ (107 cm)
Panel resolution	1920x1080
Pixel pitch	0.485 x 0.485
LCD panel type	TFT-LCD
Brightness	700 cd/m2
Viewing angle (V / H)	150° / 150°
Lenticulars	fixed, slanted
Number of views	28

One parameter missing from Table 1 is the slanting angle of the lens array. As the angle affects many of the parameters in the image creation pipeline, it will be determined in the process of finding the proper image matrix for the display.

According to [11], the non-disclosed parameters of the display are as follows. The slanted angle of lenticular sheet is $\alpha = \arctan\frac{1}{6}$, the pitch value is $P_x = \frac{14}{3}$, and $k_{off} = 0$. As the origin of these numbers is unknown, they will have to be verified later.

2.2. Originally proposed solution

The very first thing that will have to be solved is determining the assignment of sub-pixels on the FPD to the individual viewing angles of the screen. Besides the slanting angle, this distribution of (sub)pixels depends on the pitch of the lenticulars, so it could be theoretically possible to determine it analytically. But since neither of these parameters is given by the manufacturer and might prove to be difficult to measure, either optically or mechanically, another method might be preferred.

This method would take advantage of the preexisting rendering core, by feeding it with set of artificial stereoscopic side-by-side images with varying color and disparity. By capturing the output of the rendering core by an image capture device such as a DVR or a device specialized for that purpose, it should then be possible to analyze these images (that otherwise directly drive the FPD behind the lens array) and determine the correct relationship between sub-pixel position and its viewing angle.

2.3. Actual solution used

Capturing the images from the output of the rendering core proved to be difficult. The signal is not encoded in any way and can be displayed on a regular screen, allowing the observation of the interleaved image. However, the output format is 1920x1080 pixels at 60 frames per second which is not supported by capture cards available at the time of this work as they only accept frame rates up to 30 fps or higher frame rates on lower resolutions.

Another approach had to be chosen in order to get the results needed. An experimental approach of observing the behavior of the screen by an operator or a camera was decided upon. The analysis of the screen itself instead of the rendering core does not require any specialized equipment and provides deeper understanding of the forming of the final image observed by the user. The only downside of this approach is that everything has to be made from scratch. Even the formation of a single “white” pixel.

2.3.1 Finding single pixel layout

Regular television and computer monitor screens use a fairly simple way of creating a white pixel by placing three colored rectangular sub-pixels (red, green and blue) next to each other horizontally, together forming a square pixel capable of producing millions of colors. That is done by controlling the intensity of light emitted or passed through the sub-pixels while exploiting the limited spatial resolution of human vision (the sub-pixels close in proximity appear as a single light source). Some mobile OLED screens use a more exotic arrangement, but the basic principle stays the same.

In case of autostereoscopic screen with lenticular lens array the formation of a “white” pixel is not as easy, since the main purpose of the lenses is refracting light from different horizontal locations on the FPD screen to different locations on the image (viewing) plane. That applies for the sub-pixels as well so each of the sub-pixels appears at a different position and is not visible from other angles. Figure 26 gives a clearer idea of that.

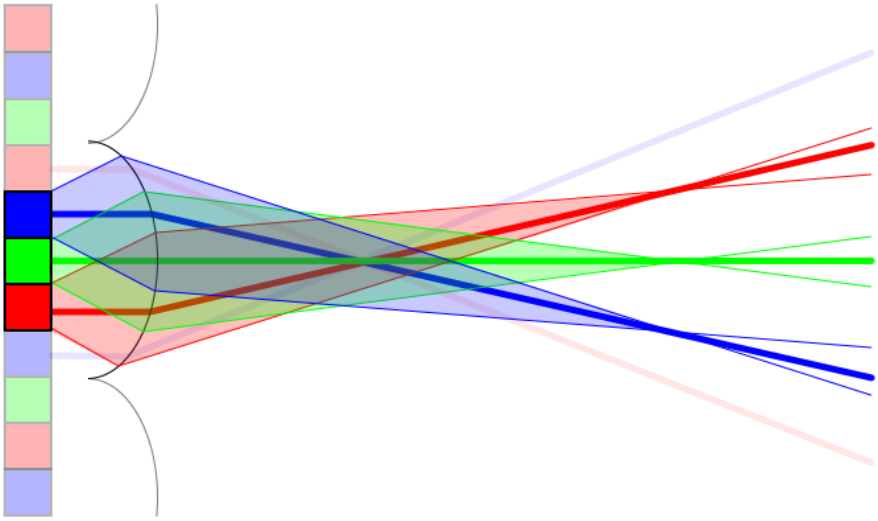


Fig. 26 Refraction of sub-pixels of a single pixel to different angles in front of the screen

It is obvious from the figure above that it is not possible to use vertical lenticulars on such screen in order to properly display full color images. Either other (vertical) sub-pixel arrangement in a pixel is required or it is necessary to use slanted lenticulars as is the case with the screen used in this work.

With slanted lenticular array the effect above persists as evidenced by Figure 27. One slight difference is that the R, G and B sub-pixels do not appear to have the same position vertically as shown in Figure 28. The main difference however, is in the possibility to assemble a RGB pixel on the lens from several pixels on the FPD that are located above each other. For example, the red sub-pixel can be used from one FPD pixel, the green one from the pixel directly below it and blue from one below that. Effectively creating a vertical pixel that is refracted as a whole in a given direction as the sub-pixels have the same x_{off}.

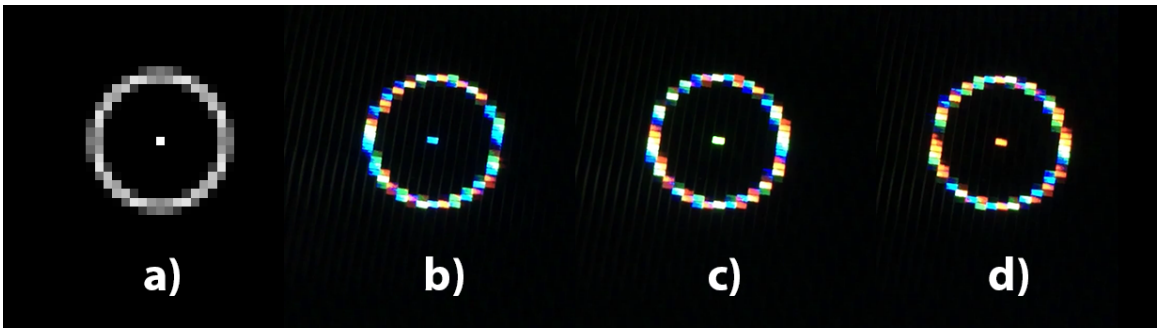


**Fig. 27 Single lit FPD pixel under the lens array photographed from three different angles.
Circle added for better orientation.
a) actual raster image used (scaled to size), b) image from the leftmost position,
c) image from middle position, d) image from the rightmost position.
Images captured at ~10 cm with several mm shifts sideways.**

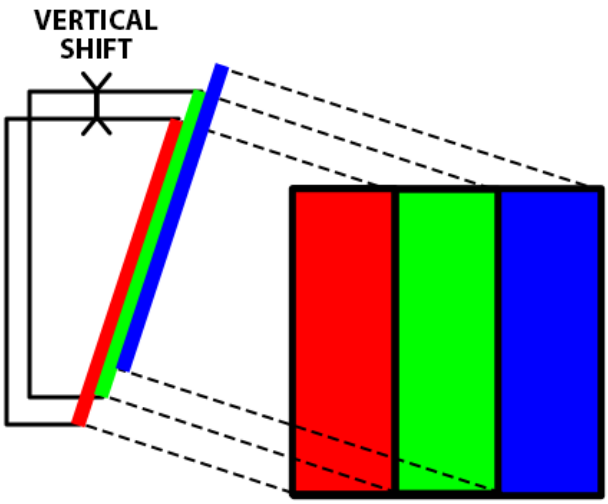


Fig. 28 Vertical shift of sub-pixels belonging to a single FPD pixel caused by slanted lens.
(Note: each sub-pixel is visible from different angle, not all three together as depicted)

The screen in this work has lenticulars that rise from left to right as seen in Figure 27 or 29 and is indicated in Fig. 28. That means the green sub-pixel will be located in one of the rows above the red-sub pixel and the blue sub-pixel above the green one, depending on the angle of the lens array. The layout could look as suggested by Fig. 29. However, while observing the behavior of such arrangement (prototyped as a raster image in Adobe Photoshop), the color components were not fading in and out simultaneously, but were still offset by some degree – better than in Figure 27 but clearly each sub-pixel still belonged to different viewing angle.

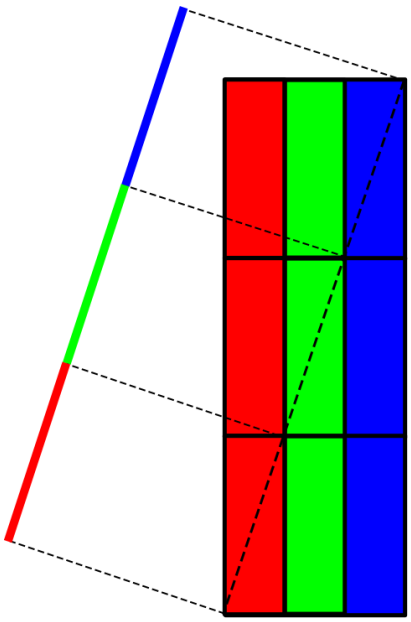


Fig. 29 First proposed sub-pixel arrangement

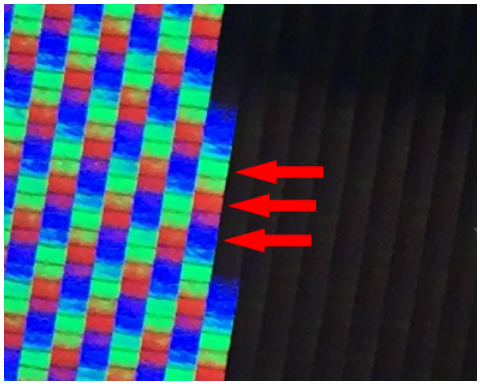


Fig. 30 Detail of the screen with visible sub-pixels.

When a macrophotography of an edge of a white rectangle on black background was closely studied, it became clear that skipping one row would be needed. Fig. 30 shows that in some instances (depending on the angle of view) the sub-pixels are doubled and in other cases every other sub-pixel is fully lit while the ones in-between are halved. The red arrows mark the latter case with evident line-skipping. The angle of the columns is also much lower (less inclined) than in Figure 29.

This observation led to proposal of a new layout. One that would not use color components of a directly neighboring pixels, but would skip one row between each sub-pixel and thus compensate for the lower slanting angle. This arrangement, shown in more detail in Figure 31, proved to be right as all three color components of the output pixel fade in and out simultaneously when continuously changing the viewing angle.

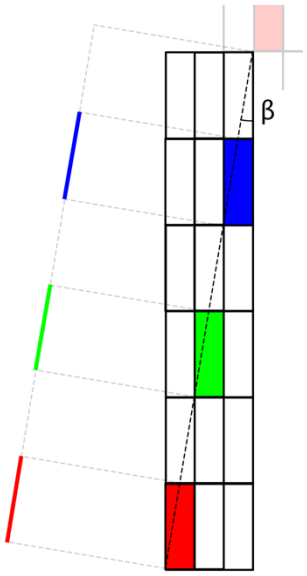


Fig. 31 Final, true layout of sub-pixels in a single output pixel for the autostereoscopic screen. Notice the dashed line is also crossing other sub-pixels belonging to output pixels of different views.

2.3.2. Computing the lens array slanting angle

The inclination of the dashed line in Figure 31 is the same as the slanting of the lenticular lenses and so the slant angle can be calculated from knowledge of the arrangement of sub-pixels (the sub-pixels have all the same x_{off} so the dashed line is parallel with the lens).

We see in Fig. 31 that the elementary output picture cell is 1 FPD pixel wide and 6 pixels high, forming a triangle with the dashed line as hypotenuse. With the knowledge of trigonometric functions, the slant angle α can be calculated. Since the dimensions are identical to the example in chapter 1.5.2, formula (9) can be used:

$$\alpha = \arctan\frac{1}{6} \tag{9}$$

The result of (9) gives us approximately 0.165 or preferably (and more precisely) 9.462°, same as in the example mentioned above. However, in this case the lens array is angled in an opposite direction.

2.3.3. Finding a same-angle pattern

As it was necessary to find location of sub-pixels that are refracted by the lens at the same angle in order to display a single pixel for certain viewing angles, it is necessary to determine the pattern of these pixels so that they all are refracted in the same direction.

Part of the solution lies in the arrangement of the sub-pixels themselves. As hinted on in the top right corner of Figure 31, logically, the whole 1x6 cell repeats itself along the dashed line so the next sub-pixel in line after the top right blue one is red again. That means the next "white" output pixel is located one (display) pixel to the right and six pixels up from the previous one. This way a whole line of pixels can be created and it will be refracted by one single lens column. Figure 32 gives a good idea of appearance of such line.

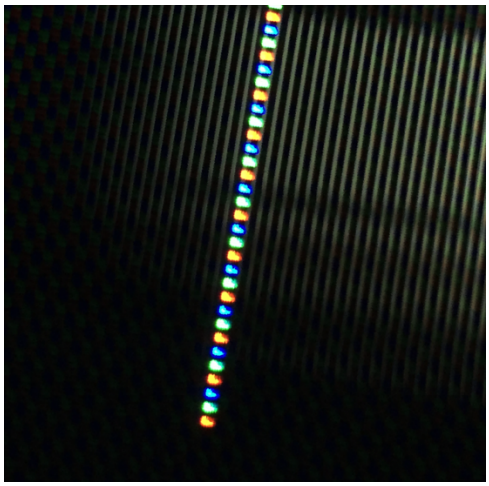


Fig. 32 Line of pixels parallel to the overlaying lens structure (visible due to reflection). All sub-pixels are refracted at the same angle.

For easy duplication of this effect on neighboring lenses, knowledge of their width would be necessary but the datasheet does not provide this information. Moreover, a non-integer width of the lens (in number of underlying pixels as a measurement unit) would further complicate things so a different approach might be preferred.

A similar trial-and-error approach as with the single pixel arrangement could be employed. Shifting a copy of the whole line until both lines fade simultaneously is one possible approach, but better yet, the complete pattern can be determined analytically.

The manufacturer claims the display is capable of displaying 28 different views (although [16] came to conclusion the real number is half of that), so in theory, the pattern should repeat every 28 pixels if not more often. A square matrix of 28 by 28 pixels was therefore created with the assumption that it would be repeated along both x a y axis of the screen.

Only one color channel is now considered (red, for example), as the other two would make things seem too cluttered and are unnecessary as they can be achieved by simply shifting the whole matrix by 2 or 4 pixels up, depending on the color channel needed.

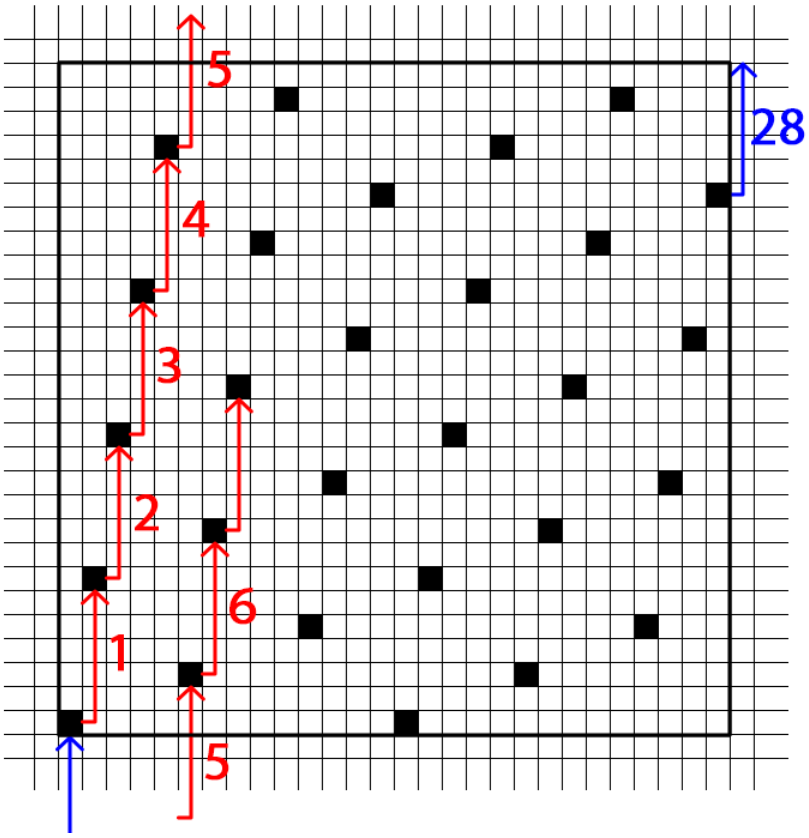


Fig. 33 Illustration of the shifting of pixels in the base matrix with wraparound.

Same pixel shifting as mentioned previously is employed (6 up, 1 to the right) starting for example in the lower left corner in the matrix (this doesn't really matter). Since 28 is indivisible by 6, the fifth shift would end up out of the matrix as illustrated by Figure 33. But because it was stated the whole matrix is repeated, the shift will end up in the matrix directly above. This can be interpreted as a simple wraparound and the shifting will continue from the bottom of the matrix. Same rule applies when horizontal edge of the matrix is reached. When the shifting is repeated enough times, no new pixels are marked as the shifting ends up on coordinates already filled (note the blue arrow in Fig. 33). In total, there is 28 pixels in the 28x28 matrix that are all refracted identically. That proves the conclusion in [16] was incorrect and there are in fact 28 discrete views.

Figure 33 also shows only a half of the matrix horizontally would suffice as pixel arrangement is repeated every 14 pixels on the x axis. In this 14-pixel interval, there are 3 "lines" of pixels, meaning there will be 3 lenticular lenses covering this area on the screen surface. The width (or pitch) of a single lens is then $P_x = \frac{14}{3}\ [px]$, same as was stated in [11]. According to Table 1, the pixel pitch is 0.485 mm so that leaves us with:

$$P_x = \frac{14}{3} \cdot 0.485 = 2{,}26\overline{3}\ [mm] \tag{10}$$

With knowledge of P_x and α, P can be calculated using (6):

$$P_x = \frac{P}{\cos \alpha} \tag{6}$$

$$P = P_x \cdot \cos \alpha \doteq 2{,}233\ [mm]$$

The matrix is now complete and can be duplicated along the whole surface of the screen. For simplification, let's assume the square matrix as a base. With its dimensions being 28 by 28 pixels and resolution of the screen being 1920 by 1080, it is necessary to duplicate the base matrix so that the resulting large matrix is slightly larger than the screen resolution and then crop it accordingly. This step is required because neither vertical nor horizontal resolution of the screen is divisible by 28.

2.3.4. Testing the same-angle matrix

Having the full screen matrix of points that are refracted at the same angle, it is now time to test its functionality. Initial theory was that the whole lit matrix should be visible from one position in front of the screen while invisible from other viewing angles. But since the pixels are refracted at the SAME ANGLE and not to the SAME POSITION, the light rays are effectively parallel to each other and the required viewing distance would be infinite, which happens to be highly impractical.

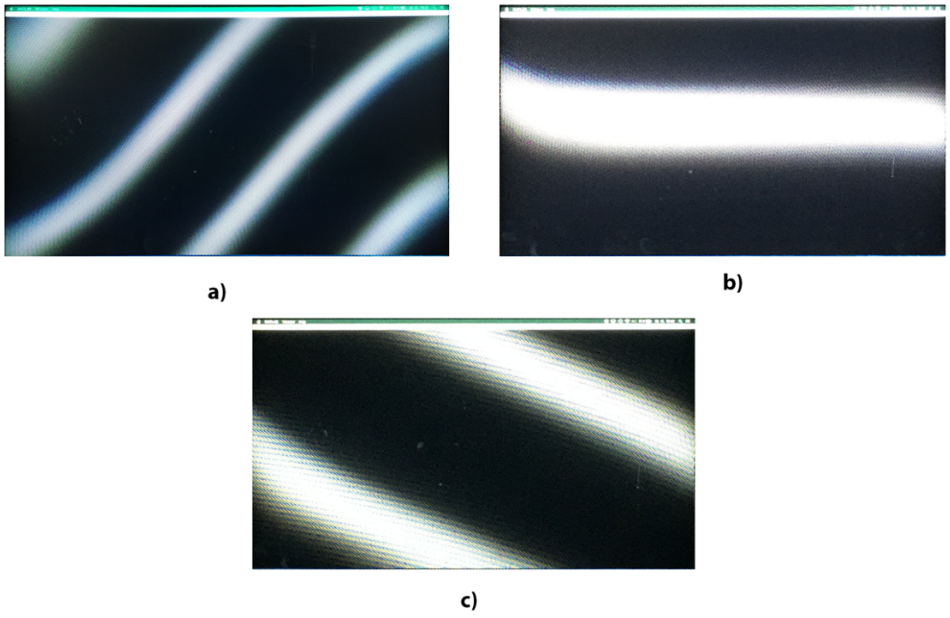


Fig. 34 Stripes produced by the refraction of the same-angle matrix as observed at a) 1 meter, b) 2 meters and c) 3 meters distance from the screen surface. Images scaled to appear the same size.

Viewing the displayed pattern from more realistic distances results in a stripe(s) of lit pixels, ones that happen to be refracted to the general direction of the viewing position. The number and orientation of the stripes changes with distance of the viewer to the screen. Different stripe patterns at 1,2 and 3 meters viewing distance are shown in Figure 34.

This phenomenon can be explained rather simply. At such relatively close viewing distances, the angle from the left edge of the screen to the viewer is vastly different from the angle at the center or the right edge. Visual explanation can be found in Figure 35.

In order to provide light from all parts of the screen to the viewer, additional refraction angles have to be used and therefore according pixel positions have to be found.

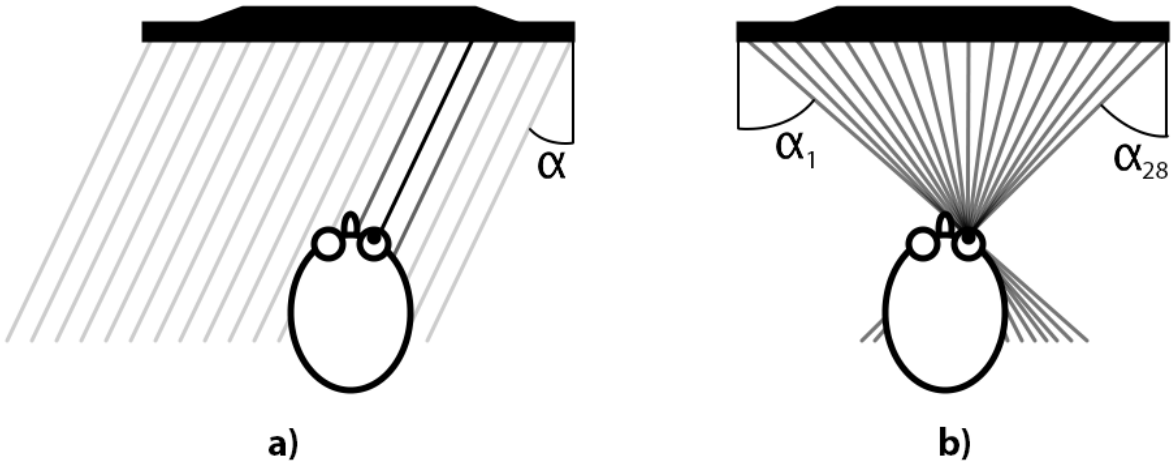


Fig. 35a) Same-angle matrix refracted by the lens array at the same angle α. Only some of the rays are visible from a given position.
Fig. 35b) In order to properly display images at a given position in space, rays coming from different angles (and thus originating from different matrices) are required.

2.3.5. Determining relative position of pixels from neighboring views

The only image matrix known at this time was shown in Figure 33. For simplicity this matrix will be referred to as matrix belonging to Angle 1. In order to find the matrices for other angles, the knowledge of arrangement of this matrix and its "stripe" appearance on the screen will be used.

Note in Figure 33 that the pixels refracted at the same angle are repeated every 14 pix horizontally and every 28 pix vertically. Every other row does not contain any pixels belonging to Angle 1. With this in mind, it can be assumed every column will contain pixels belonging to all the angles while neighboring rows will consist of totally different sets of angles. For this reason, the main operation used to find the relation between neighboring angles will be vertical shifting. Horizontal shifting alone would lead to unchecked positions in the odd rows.

As stated in 2.3.4. the output image of such matrix appears to the viewer as a lit stripe on the screen that continuously changes its position with the change of viewing angle. The same position-changing also happens the other way around, when the viewer is stationary and the underlying image matrix is changed.

When observed from approximately 2 meters, only 1 stripe is visible with 28 possible vertical positions on the screen – same as the number of possible image matrices.

The position of the stripe is sequential, approximating the continuous change of its position with the change of viewing angle – the stripe displayed by matrix of Angle 1 is directly next to (above or below) stripe from Angle 2, which is next to Angle number 3 and so on. By observing the effect of the shift of the base image matrix on the stripes, it is possible to determine the angle number of each of the base matrices.

A Matlab script was made that shifted the Angle 1 matrix in vertical direction in increased amounts with each iteration starting with shift by 1 pixel and ending with 28 pixels. As expected the maximum shift of 28 px resulted in no change of position of the stripe whatsoever. The smallest change of position manifested when the matrix was shifted by 9 or 19 pixels in either direction (up or down) - with wraparound 9 pixels down is equal to 19 pixels up and vice versa. Depending on the orientation of numbering the angles, 9 pixels shift down of Angle 1 matrix can be either Angle 2 or Angle 28 matrix (28 neighbors with 1 as the angles were numbered artificially).

The following Figure shows the complete base image matrix with location of pixels belonging to all the possible angles (for a single color channel) when Angle N+1 is shifted by 9 pixels down from the position of Angle N.

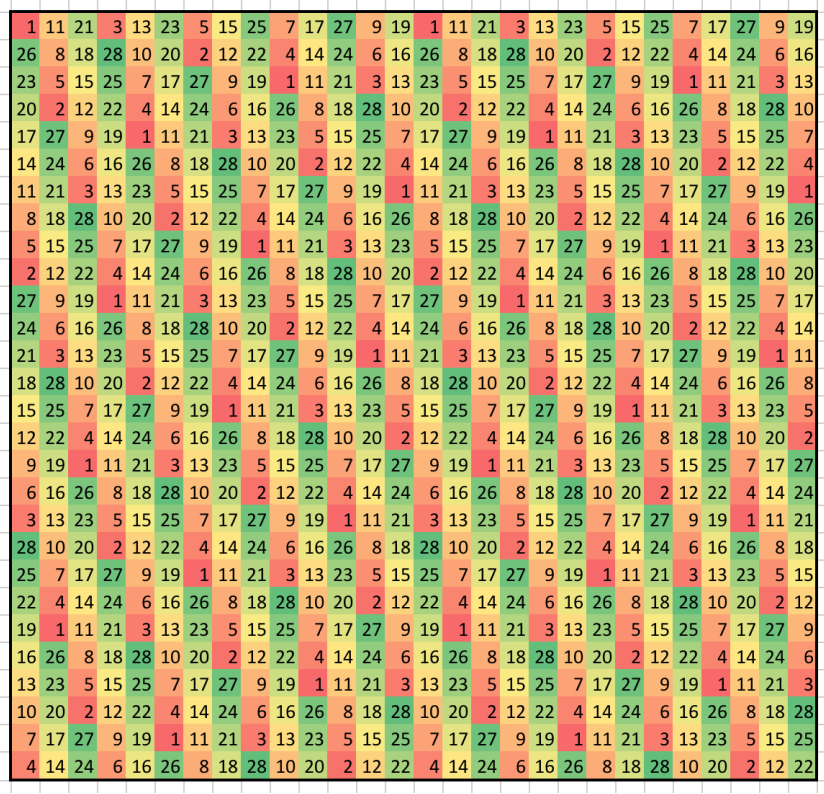


Fig. 36 Complete 28x28 image matrix with pixels bellonging to different viewing angles (one color channel only). Color coded for easier location of pixels from neigboring angles.

2.3.6. Creating same-position matrix

With the knowledge of the matrix above, it is now possible to display pixels – and therefore images - from any part of the screen to any viewing angle. In accordance with the results from chapter 2.3.4. the pixels with the same Angle number still produce a stripe pattern. When attempted to replace the pixel values with values from actual multiview source images (with each pixel being replaced only by values from image with the same number), the screen displays a skewed image as each view is displayed in its own stripe and all of them are visible from the same position.

It is clear that the matrix (or the source images) has to be modified so that one image is displayed by pixels belonging to multiple different Angles depending on the position on screen. The required change is illustrated by Figure 37.

Let's only use pixels belonging to one angle (Angle 1 for example) for the next experiment – all the other pixels remain black. We know that when all pixels numbered 1 are white (meaning there are R, G and B FPD pixels forming the 1x6 "white" output pixel) we can only see the one stripe changing its position across the screen (a horizontal stripe visible from 2 meter distance is assumed). A white stripe similar to the one displayed by the screen, only thinner, can be used as an input image, essentially masking the image matrix so that it only appears lit on certain part of the screen.

This stripe is only visible from a certain angle (from certain position) at which the stripe produced by the screen is positioned the same as the stripe in the source image. The screen remains completely black from other angles (some minimal crosstalk is present). Visual approximation of this experiment is depicted in Figure 38.

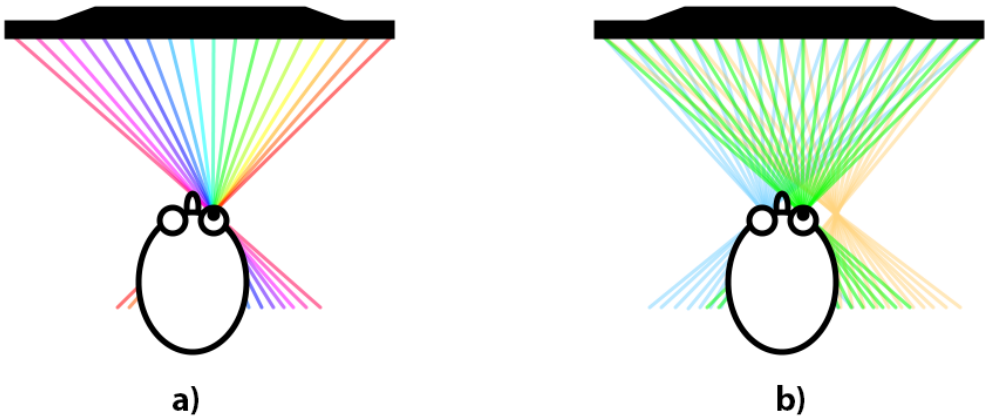


Fig. 37 a) Result of direct mapping of images to numbered pixels in the matrix - all the images (or parts of them – stripes) are visible from any position in front of the screen (each color in the image represents one source image displayed), b) the desired result – only one image is visible from each position.

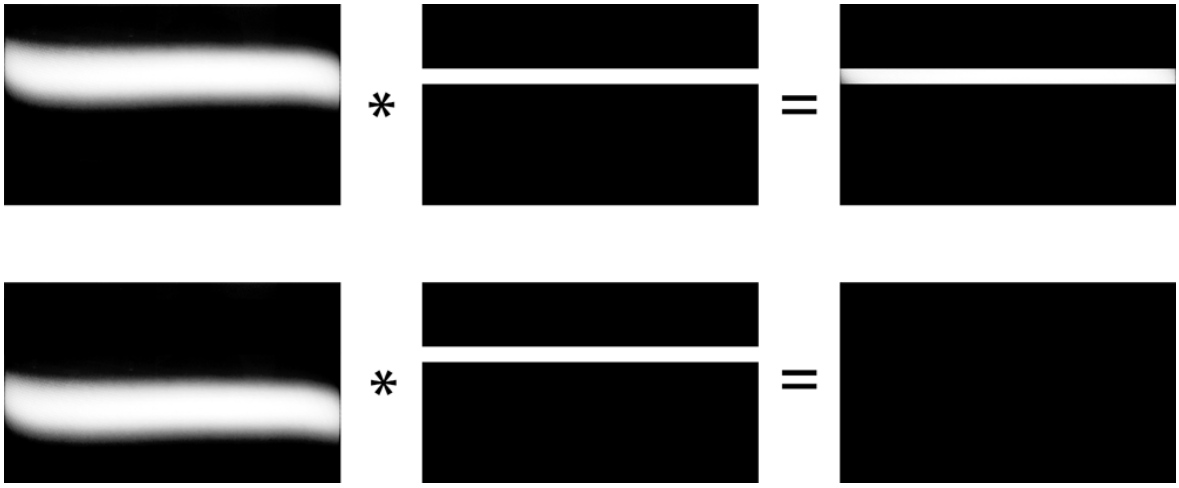


Fig. 38 Appearance of the stripe image at desired position (top) and at other positions (bottom).

As the stripe created by the lenticulars has different position for each pixel number it is possible to create similar "masks" for each of them, resulting in fully lit screen visible from one viewing position and black screen from other positions.

In order to find the ideal height of the white stripe "mask" used as an input, some simple calculation is needed. Since there are 28 possible vertical positions of the stripe, the difference in their position can be calculated by as $\frac{1}{28}$ of the screen height:

$$\delta = \frac{1}{28} \cdot 1080 = 38{,}57\ [px] \tag{11}$$

When rounded to the nearest ten for simplification (the 1.5-pixel difference should not make any visible difference), it can be assumed each stripe is vertically offset by 40 pixels from the next one, meaning the height of each of them should be 40 pixels (except for the one at the bottom of the screen which is cropped due to rounding).

All 28 same-angle matrices (SAMs) masked by 40 pixels high horizontal stripes can be summed into single matrix that is displayed only in one position, essentially creating a same-position matrix for one position.

In order to assemble same-position matrices for other positions in front of the screen, shifting the masks by 40 pixels and then multiplying them with the same-angle matrices is required. Since shifting the mask by 40 pixels is equal to using the mask intended for the neighboring SAM, it is sufficient to shift the order in which the masks and matrices are multiplied. The complete overview of the same-position matrices creation shown in Table 2.

Table 2. Creation of same-position matrices.

Angle number	Same-angle matrices																														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28		Position number	
mask number	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*			
∑	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	=	1	Same-position matrices
∑	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	=	2	
∑	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	1	=	3	
∑	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	1	2	=	4	
∑	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	2	3	=	5	
∑	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	3	4	=	6	
∑	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	4	5	=	7	
∑	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	5	6	=	8	
∑	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	6	7	=	9	
∑	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	7	8	=	10	
∑	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	8	9	=	11	
∑	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	9	10	=	12	
∑	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	10	11	=	13	
∑	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	11	12	=	14	
∑	15	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	12	13	=	15	
∑	16	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	13	14	=	16	
∑	17	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	14	15	=	17	
∑	18	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	15	16	=	18	
∑	19	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	16	17	=	19	
∑	20	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	17	18	=	20	
∑	21	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	18	19	=	21	
∑	22	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	19	20	=	22	
∑	23	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	20	21	=	23	
∑	24	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	21	22	=	24	
∑	25	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	22	23	=	25	
∑	26	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	23	24	=	26	
∑	27	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	24	25	=	27	
∑	28	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	25	26	=	28	

The same-position matrices obtained can be used to display multi-view images on screen properly (only one image is displayed at given position) but create some visible artifacts caused by hard edges of the masks. As illustrated by Figure 39, pixels located near the edges of the masks might be missing or doubled as a result of differences in position of pixels in each of the stripes.

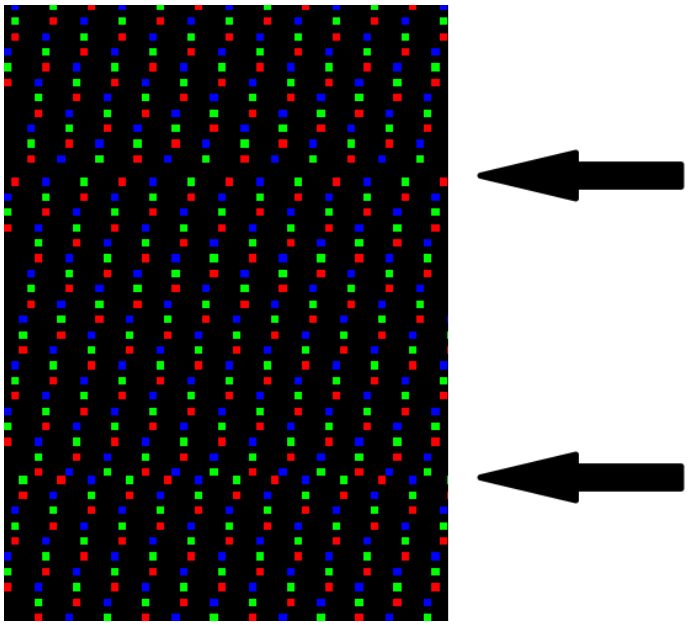


Fig. 39 The cause of artefacts in displayed images.

In order to eliminate this unwanted effect, the used masks need to allow for smooth transition from one stripe to another. A set of slightly wider stripe masks with gradient falloff at the edges was created. The artifact was successfully suppressed but a tradeoff in form of a small, almost imperceptible artificial crosstalk had to be introduced into the image (the pixels in the gradient sections of the masks are computed as a weighted average from neighboring views – can also be visually explained with Fig. 40b).

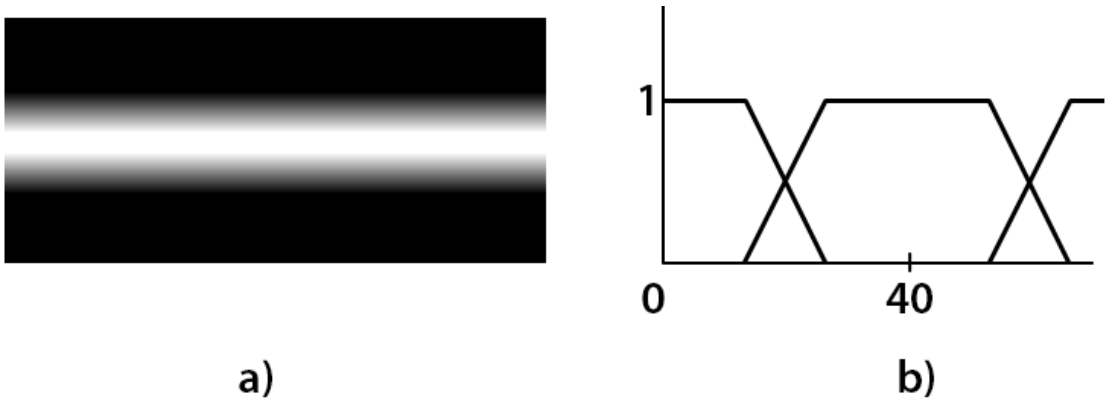


Fig. 40 a) Example of the smoothed mask, b) principle of its summing with neighboring masks.

Part of one of the 28 final same-position matrices is depicted in Figure 41. Note there are no visible artifacts as in Figure 39. The apparent difference in brightness is later compensated by summing with other matrices.

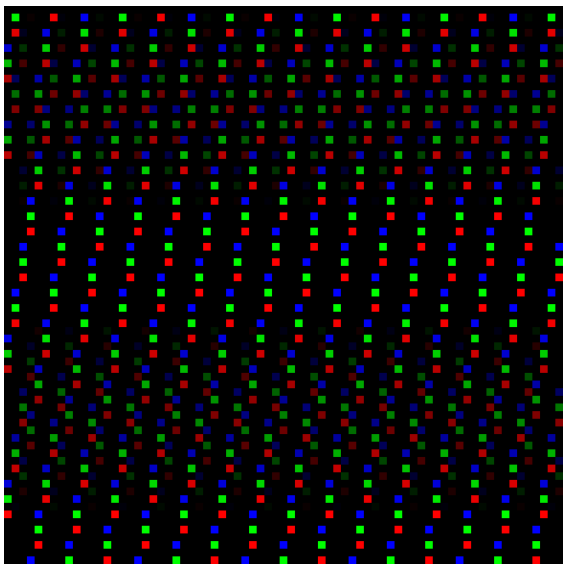


Fig. 41 A close-up of part of the final same-position matrix.

The result of multiplying each of the SPMs with an image taken from slightly different angle is a set of 28 partial images that is each refracted to its according viewing zone (basically it is a set of SPMs with mapped pixel brightness values). By summing these images into one, the final interlaced image that can be displayed on the screen is created.

A cropped part of such image is pictured in Figure 42. Notice there is no visible trace of the use of the stripe masks and only a little, near imperceptible trace of the difference in pixel brightness depicted in Fig. 41.

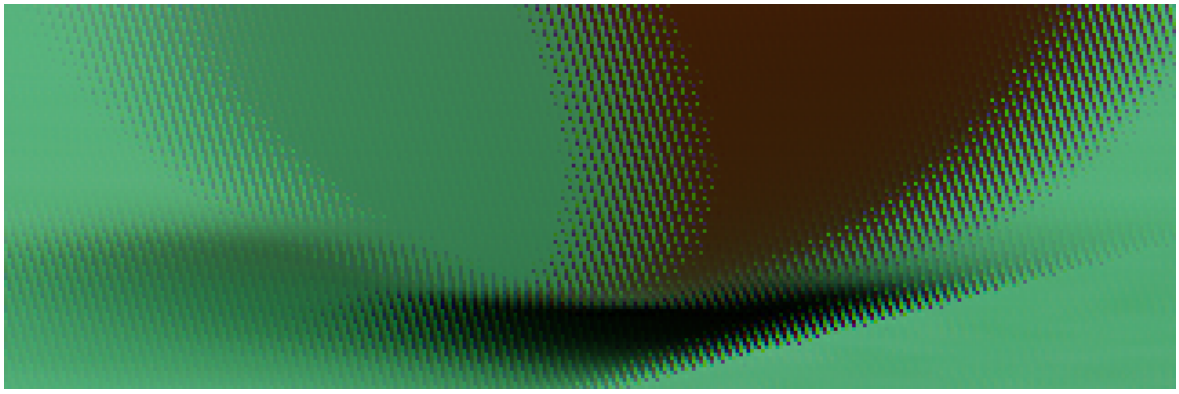


Fig. 42 Detail of the interlaced output image. Source image pictured in Figure 25 has been used.

2.4. View mapping

With the ability to display different images to actual positions (or zones) in front of the screen, the next step is to decide which source multi-view image is displayed in which viewing zone. This process is called view mapping (or sub-view mapping).

2.4.1. Direct mapping

The simplest and most logical view mapping scheme is directly displaying image number 1 to viewing zone number 1 (trough same-position matrix number 1), image number 2 to zone number 2 and so on. Figure 42 shows an example of such mapping scheme.

This view mapping scheme provides the smoothest possible motion parallax as the full range of source images is used. On the other hand, the downside of directly mapping images to views with according number is the largest possible cross-talk (although that can be taken advantage of in simulating sort of a depth-of-field effect where foreground elements remain relatively sharp as their position does not change too much from view to view whereas background and foreground objects with large parallax get blurred by the cross-talk).

The other problem with direct mapping scheme is hard transition from view number 28 back to view number 1 that lies right next to it (all 28 viewing zones are repeated several times in front of the screen as shown back in Chapter 1.5., Figure 12) – the well-known "flip" that appears with lenticular prints. The hard jump from the rightmost image back to the leftmost image causes the viewer to observe pseudoscopic image with large parallax (made even worse due to cross-talk).

This mapping scheme, called "Smooth" and represented by blue line in Figure 43, is best used with multiview images with relatively small parallax (to reduce cross-talk) and in cases where the viewer(s) is not expected to move too much (in order to avoid the uncomfortable pseudoscopic image in certain viewing zones).

2.4.2. Direct view mapping with mirroring

In order to suppress at least one of the flaws of the direct mapping scheme – the "flip" from image 28 back to image 1, it is possible to smooth the transition by reusing some of the images in between. The easiest way to do this, a way that is also implemented in the resulting software of this work, is to only use half of the source images available and "mirror" them in order to get all 28 images required. The allocation of the images is then like: 1, 2, 3…, 13, 14, 15, 14, 13, …, 3, 2.

Such mapping scheme results in smooth transition between viewing zone 1 and 28 as they display images 1 and 2 respectively. As a tradeoff, the second half of viewing zones (15-28) display pseudoscopic images (as the source images are in incorrect order) - that is still uncomfortable for the viewer but much better than the "flip".

In order to observe the correct stereoscopic image, the viewer is limited to only first half of the viewing zones which might be quite limiting for static viewing (e.g. presentation to a group of people that need to position themselves correctly). For moving audience (e.g. customers in a shop where the screen is used as an ad kiosk), the motion parallax with no "flipping", only some "squashing and stretching" provides much better experience than the simple direct mapping scheme – the pseudoscopic image is subjectively not that evident when the viewer is moving.

Other properties of this mapping scheme – both cross-talk and smooth motion parallax - are identical with the simple direct mapping.

In Figure 43, this mapping scheme is represented by red line and is called Smooth + Cyclic ("Cyclic View" is term used by Dimenco in their mobile application [17] for controlling some newer versions of their rendering core).

2.4.3. Original Philips mapping scheme

According to [11], Philips (or more likely Dimenco) uses a different view mapping scheme that partially suppresses the high amount of cross-talk and offers similar "flip" reduction as "Smooth + Cyclic" does.

The allocation of the 28 views can be described by the following number sequence:

1, 1, 1, 4, 4, 4, 7, 7, 7, 10, 10, 10, 13, 13, 13, 16, 16, 16, 19, 19, 17, 15, 13, 11, 9, 7, 5, 3.

The ascending part of the Philips scheme (1, 1, 1, … , 19, 19) indicates that the audience can get correct 3D perception. The usage of the same view for three viewing zones can help reduce most of the cross-talk.

In the descending part (17, 15, 13, 11, 9, 7, 5, 3), audience receives pseudoscopic 3D perception. As the difference between neighboring viewpoints is gradually reduced rather than skipping to 1 directly from 28, as with the conventional scheme, the transition is smooth and almost imperceptible when compared to direct mapping [11].

By using each of the 15 views three times (in the ascending part), this mapping scheme not only reduces cross-talk (no cross-talk is created using identical images) and thus make images appear sharper, but unfortunately limits the smoothness of motion parallax. By skipping two views with each change of the (now three times wider) viewing zone, the switch between the images becomes more apparent. Moreover, some amount of cross-talk is still present and no longer produces the relatively smooth blur as with direct view mapping scheme, but appears with more sharp, "jagged" edges.

The difference between all described mapping schemes can be quite nicely visualized in a line graph such as the one in Figure 43. With its help, some general rules for creating view-mapping schemes can be defined:

- straight diagonal line means smoother motion parallax but more cross-talk
- "stairs" with multiple points on the same horizontal line mean less cross-talk but worse motion parallax
- ascending parts produce correct stereoscopic images
- descending parts produce pseudoscopic images
- leveled parts means repetition of the same image in multiple viewing zones
- large difference between viewing zone 1 and 28 means "flipping" effect
- small difference means smooth transition from one instance of the viewing zones to another
- the number of levels used is equal to source images required

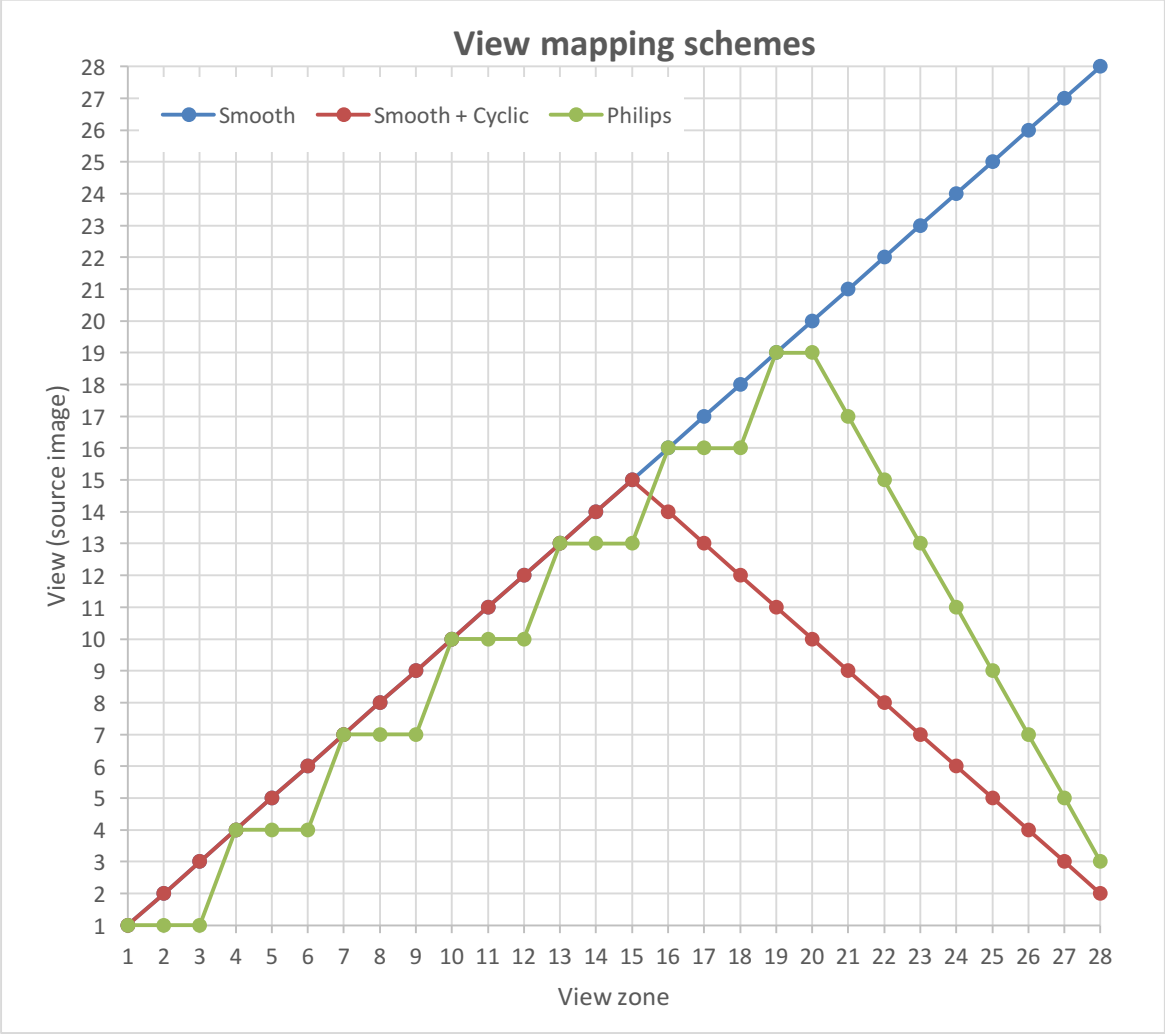


Fig. 43 Different view-mapping schemes.

2.4.4. Other mapping schemes

Many more view-mapping schemes can be designed (as evidenced by the blank space in the graph above) with different properties and for different use cases.

For example, [11] proposes an 8-view mapping scheme:

The proposed scheme makes the allocation like

1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,8,8,6,5,4,3,2,

which uses only 8 different views. If the original disparity is small,

1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,6,6,7,7,7,8,8,8,6,5,4,3

is an even better scheme. If the original disparity is quite big,

1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5,4,4,4,3,3,3,2,2

is used instead and get almost the same effect [11].

3. Matlab implementation

All operations described in previous chapters were realized as a set of scripts in Matlab version R2014a with use of Image Processing and Computer Vision System toolboxes.

3.1. Same-angle matrix creation and display

First, a set of scripts enabling creation and display of same-angle matrices was written along with additional functions that enable user-friendly switching between different matrices via a graphic interface. With help of these scripts, effects of different same-angle matrices displayed on the screen (in form of stripes) can be studied.

Hierarchy of the used scripts is laid out in the following figure.

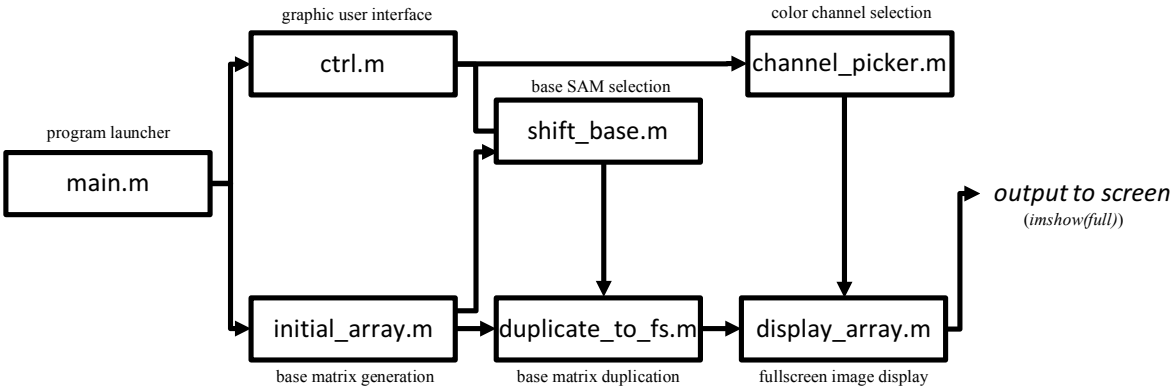


Fig. 44 Generation, duplication and displaying of same angle matrices using custom Matlab scripts and GUI.

main.m - Serves as a launcher for the set of scripts. It contains declaration of some initial variables and calls other functions (**initial_array.m**, **duplicate_to_fs.m** and **ctrl.m**).

initial_array.m – Declares global variables and creates a 28 by 28 base matrix in accordance with steps described in chapter 2.3.3, mainly using *circshift()* function. The output is a 28x28x3 matrix, where the third dimension represents color channels.

duplicate_to_fs.m – Uses *repmat()* function to copy the base matrix enough times that it covers the area of the whole screen, then crops the image (as explained at the end of 2.3.3.). Inputs for the function are the base matrix and information if any shifting was triggered by the user. There is no direct output from the function, it only changes values of global variables. This function directly calls **display_array.m**.

display_array.m – Creates a new figure window with all menu bars hidden and positions it so that its contents appear on the second screen (autosterescopic display) as a full screen image. Input for the function is information about shifting – if shifting has been done, the window is already created and some lines of the script are not needed (resulting in faster response). Otherwise the code works with global variables.

ctrl.m – A control GUI enabling the user to select which same-angle matrix and what color channel (R G, B or all) to display. The look of the user interface is pictured in Figure 45. A change in the selected "View number" calls **shift_base.m** with the selected number passed on as an input value. View selection can be made either by mouse cursor or by using the arrow keys. Pressing the "Cycle Channels" pushbutton calls the **channel_picker.m** function.

Fig. 45 User interface for experimenting with same-angle patterns.

shift_base.m – Shifts the base matrix (passed on as a global variable) in order to move active pixels to positions belonging to an angle number selected in **ctrl.m**. Then calls **duplicate_to_fs.m**, passing the shifted matrix as its input.

channel_picker.m – Changes value of global variable "color" used to determine which color channel (R, G, B or all of them) to display. Calls **display_to_fs.m** in order to make the change visible on the screen.

Note: the currently displayed same-angle matrix can be saved to a file using a *imwrite(full,'filename.png')* command.

3.2. Creation of same-position matrices

With same-angle matrices created by the previous scripts it is possible to convert them into same-position matrices as described in chapter 2.3.6. The following scripts create the necessary masks, multiply them with same-angle matrices and save the results as PNG files for further use.

Relations between each of the scripts used in the process are described by Figure 46.

mask_success.m – A main part of the script that takes care of all the computation and calls all other functions. It creates a 28-element cell object with all the same-angle matrices, multiplies each cell with the according mask and sums the results into one same-position

matrix. This process is then repeated for each of the remaining 27 positions and the resulting matrices are saved as PNG image files.

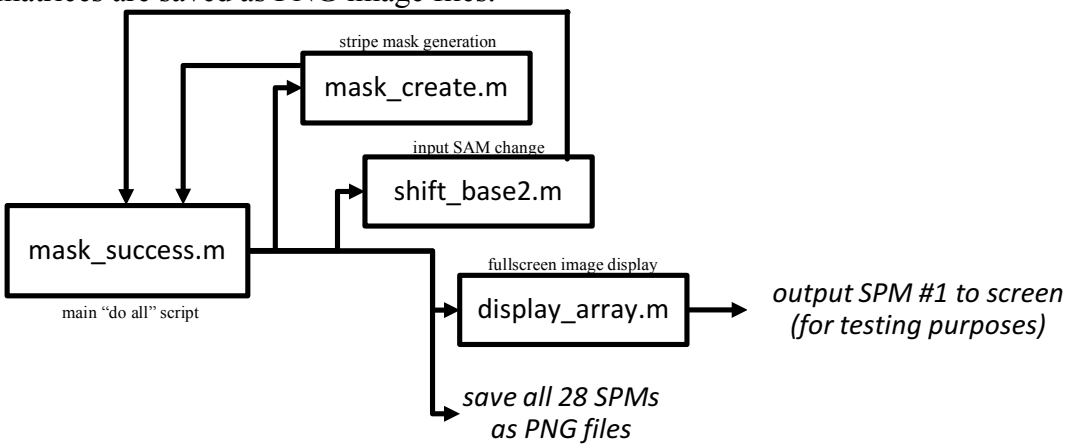


Fig. 46 Same-angle to same-position matrix conversion.

mask_create.m – A function that creates the gradient mask as a single pixel wide element that is then stretched to the width of the screen and added to 1920x1080 black background. Using a *circshift()* function in a cycle, all other masks are created by shifting the mask by 40 pixels with each iteration of the cycle. The output of the function is a 4-D (1920x1080x3x28) matrix containing all 28 masks.

shift_base2.m – An almost identical function to **shift_base.m** described on the previous page with lines from **duplicate_to_fs.m** added so that the output is already duplicated along the screen area – a 1920x1080 same-angle matrix.

Saving the computed same-position matrices as image files is used to save time when displaying actual multiview images on the screen. As computation of the SPMs requires considerable amount of processing (which may be caused by imperfect implementation), it is more time-efficient to have them saved with the image displaying scripts as a set of files. Loading of the SPMs from files is significantly faster than their generation which results in better user experience. By having the matrices saved, there is no need for some of the previously described scripts when only on screen image displaying is required.

3.3. Generating new views from 2D+depth source

Having the SPMs available, it is possible to display a set of individual images on the screen, forming an interlaced multiview image. In order to be able to display images in 2D+Z format, 28 individual source images have to be generated from the 2D "middle" view and the depth map.

The method for this new view generation is called Depth Image Based Rendering (DIBR). The basic idea behind the method is that pixels of the middle view (Fig. 47a) are shifted left or right by an amount derived from the depth map (Fig. 47b) - same shade of gray means same depth plane - shift by the same amount). Blank, previously occluded parts of the new

image (holes, Fig. 47c) are then filled with textures synthetized from neighboring areas (Fig. 47d).

A Matlab implementation of this method by Mr. Varuna De Silva was found on Mathworks File Exchange website [18]. His implementation, however, was designed only to generate a stereo pair (left and right image) from 2D+depth video in YUV color space. To suit the needs of this work, some heavy modifications of the code were necessary.

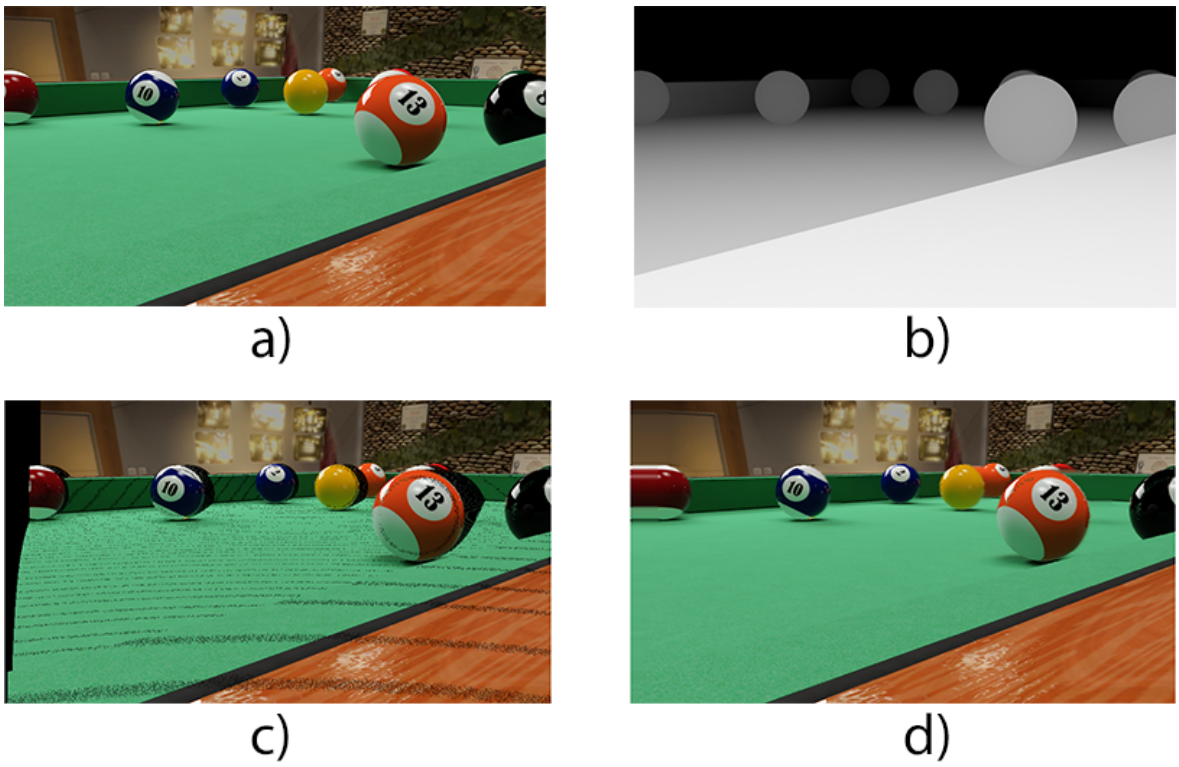


Fig. 47 DIBR generation of new view of the scene. a) Original 2D "middle view" image, b) its depth map, c) newly generated image with black holes, d) with hole-filling.

First up, change of the input from YUV video to RGB image files was made. A *for* cycle switching the video frames was removed as well as any color space conversion functions. The script solved shifting of luminance and chrominance components separately, so the chrominance part of the code was removed and the luminance part was modified so it works with all three color components of the RGB image simultaneously. A few lines of code enabling the use of both 8-bit grayscale and 24-bit RGB depth map files have been added.

The next modification made to the code was to add the ability to render more than two new images. Each of the left and right images has its own code computing the shift to the left and right respectively. In order to create 28 images total, each part of the code have to provide 14 images (14 to the left of the middle view and 14 to the right) so a *for* cycle with 14 iterations was introduced to both parts of the code. In each iteration of the cycle, the maximum amount of shift is divided by 14 and immediately multiplied by the number of currently running cycle, which results in different (gradually increasing) amount of shift in each iteration of the cycle.

Each newly rendered view is then added to a cell array (separately for left and right halves of the views). The last step is to unify the two arrays containing the 14 different "left" and 14 different "right" images which results in a single cell array with all 28 rendered views sorted from left to right.

As some of the shifted pixels happen to end up outside of the actual image boundaries, an intermediate step has been used in form of adding "safe zones" to the sides of the source image. After the new image is rendered, these safe zones are cropped from the image. Only the cropped result is saved in the cell array.

The modified algorithm is saved as **dibr.m** function file. Its inputs are the 2D "middle view image, the depth map, vertical and horizontal resolution of the input images and the number of required new views (usually 28). The output of the function is the cell array containing the newly rendered images.

Note: The algorithm is fairly time consuming. It is recommended to use lower resolution images for testing purposes as the render time is directly related to input image resolution (it is also possible to set lower resolution as the input variable of the function – the images will be resized accordingly).

Tip: In order to view the rendered images, it is possible to create a simple "look around" animation (sometimes referred to as wigglegram) by displaying the individual cells of the array in fast succession. Use the following code to do so:

```
for n=1:28
        imshow(uint8(array_name{n}))
        pause(.1)
end
```

3.4. Deriving depth map from stereo pair

Stereo pair source images are not capable of providing full motion parallax by themselves. In order to enable the creation of other views of the same scene, the stereo pair has to be converted to 2D+Z format and generate new images as described above.

The basic idea behind the conversion is that the two images are compared on a pixel (or block of pixels) level and horizontal distance (disparity) of the same (ideally, or more likely similar) pixels or blocks is measured and saved in a new image matrix – the disparity map. As described in chapter 1.2.1, binocular disparity is one of the strongest depth cues, so the disparity map can be considered as a depth map.

In practice, many different algorithms can be used to calculate the disparity map with varying complexity and quality of results. Bachelor thesis by Martin Krupička [19] compared some of the possible algorithms and showed that the results can differ quite significantly. Some of

the scripts used or mentioned in [19] were tested together with the DIBR algorithm in order to find the best solution for the resulting application created as a part of this thesis.

Several stereo pair images were used for testing the algorithms, with each of the scripts giving varying results – generating good depth map estimations in some cases and failing in others. None of the tested scripts stood out above the others so significantly that it could be definitively chosen as the best one to use. The implementations also varied in their computation speed, some taking as long as a minute to generate the disparity map, others only taking several seconds. Also, quality of the generated maps usually was not proportional to the time taken.

As the tested algorithms were not satisfying enough, a new script has been created. A *disparity()* function found in the Matlab Computer Vision System Toolbox has been used as a basis for the new script, with image filtering applied to smooth out the generated rough disparity map. This new script generates the disparity map with roughly the same results as the other tested scripts, providing usable map for some stereo image sources and failing to do so with others. The main difference is with its speed which is much better than the other scripts, taking only fraction of the time.

The newly created code is implemented in the final application but as it does not generate usable depth maps in all cases (hence the "Experimental" label in the displaying application), other way of creating the depth map might be preferred.

Both freeware and commercial application specialized in depth map generation are available. These applications allow for greater control over the results by giving the user the ability to set various parameters used in the computation process.

Commercial applications are most notably represented by software and plugins by YUVSoft Corp. [20], whose products were used in stereo 3D conversion of both theatrical and TV movies.

Several freely available applications using different algorithms, also tested by [19] are available from [21]. DMAG5 application has been tested with quite good results, making it probably the preferred way to pre-process a stereo pair image before using the created application for displaying images on the autostereoscopic screen.

3.5. Problem with no support for fullscreen figures in Matlab

By default, Matlab is not able do display figures in true fullscreen, without window borders, header, menubar etc. so a workaround has to be used.

Because figure properties are well adjustable, it is possible to turn off any unnecessary window elements such as toolbar and menubar, leaving only a bare window. When maximized, this window still behaves a regular window in the OS and its header and borders are visible. In cases where Windows taskbar or OS X Dock and/or menubar are set visible on the connected screen, these are visible as well so it is necessary to set them to only display on the main monitor.

In order to mimic real fullscreen look of the figure, the window has to be left as floating (i.e. not maximized). By resizing the window an positioning it in a way that the borders are off-screen, only the actual image area is visible (at least on some Windows systems that allow resizing the window beyond the screen resolution, OS X does not allow moving the window header above the screen area).

To set the window this way, it is necessary to know the resolution of both the main display and the autosteroscopic screen, which is obtainable by *get(0,'ScreenSize')* function. The width of the main screen can then be used to offset the window to appear at the left side of the autostereoscopic screen. Resolution of the autostereoscopic screen then can be used to set the dimensions of the window (as this resolution is always the same, it can be set to 1920x1080 directly). In order for these settings to work it is necessary to use the autostereoscopic screen as secondary monitor and have it positioned (virtually) to the right of the primary monitor, as shown in Figure 52.

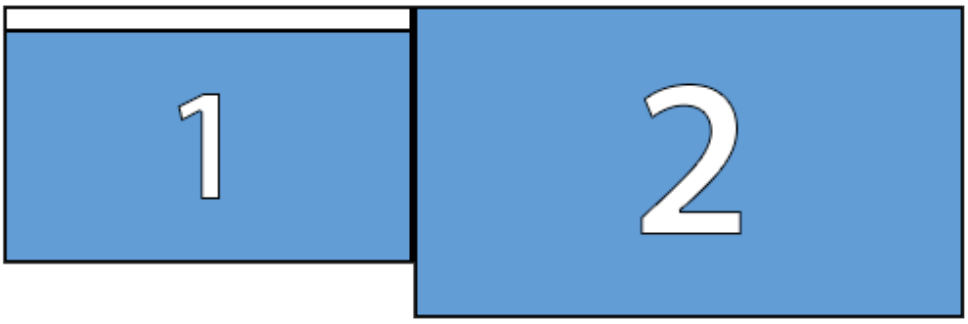


Fig. 48 Arrangement of monitors to be used with the software. Monitor 2 represents the autostereoscopic screen.

These figure settings are used in **display_array.m** function, along with some additional settings like black figure background (instead of default gray). When the figure is created with the function a simple *imshow()* is sufficient for proper fullscreen image display as it uses the last figure created.

4. Application overview

With the use of previously described scripts and generated same-position matrices, an application allowing the display of correctly interlaced output images has been created. This application uses a simple GUI (created using Matlabs GUIDE tool) allowing the user to select preferred source image format and view-mapping scheme, lets him choose the source images from files saved on the computer and makes all the necessary calculations required to generate the correctly interlaced multiview image which is then displayed on the autostereoscopic screen.

A screenshot of the main program window in its default state is shown in Figure 49.

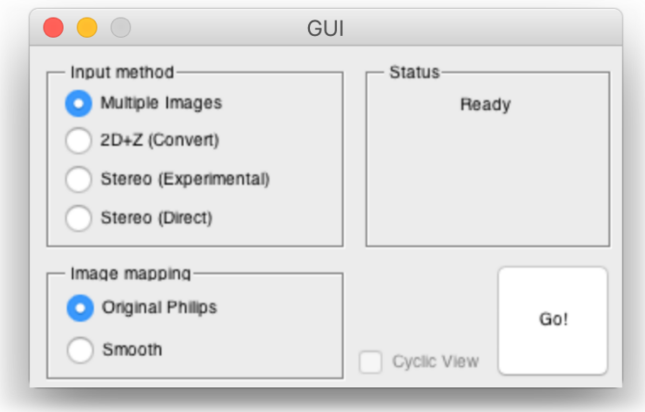


Fig. 49 User interface of the created application.

The main window of the application is divided into four sections. The "Input method" section, "Image mapping" section, a status information section and a program-launching button.

The input method section contains four radio buttons that let the user switch between four image format inputs – multiveiw image files dubbed "Multiple images", a 2D image and its depth map, dubbed "2D+Z (Convert)" (informing 2D+Z to 28 images conversion using DIBR will be applied), Left and Right image files converted to multiview, dubbed "Stereo (Experimental)" (informing the user that the algorithm is not 100% reliable and might produce bad results) and direct use of stereo pair images.

The image mapping section allows the user to switch between all three view mapping schemes described in chapter 2.4. – the original scheme used by the render core (Original Philips) and the direct "Smooth" mapping scheme. The latter can be further modified with the "Cyclic view" checkbox enabling the suppression of the image "flipping".

The status part of the window informs the user about the currently selected combination of input method and image mapping (when changed from default settings) and also displays progress information about currently running task such as the percentage of DIBR calculations already done.

The “Go!” button launches the actual image processing, beginning with the opening of dialog windows allowing the user to select source files. The subsequent processing of these files differs according to the selected “Input method”. These differences are described by Figures 50, 51 and 52.

4.1. Processing multiple image files

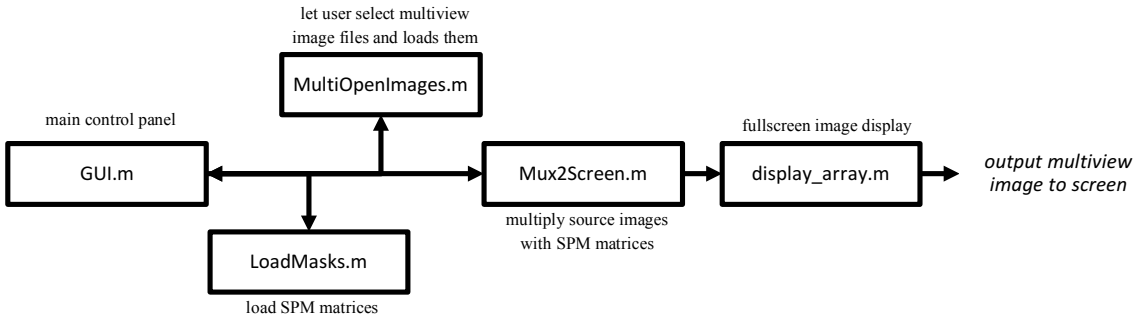


Fig. 50 Scripts used when displaying multiview image files.

GUI.m – The main script of the application containing the user interface described earlier. It calls other functions after the “Go!” button is pressed, passes on settings selected by the user and receives the status messages displayed in the “Status” section of the GUI.

LoadMasks.m – This script is called immediately after the application is launched and after every change of selected image mapping scheme. It loads the SPM image files saved by the SPMs-creating script described in chapter 3.2. and passes them to **GUI.m** as a 4D matrix.

MultiOpenImages.m – Creates a new “Open” window which allows the user to select multiple image files containing individual “views”, then loads the images into a cell array which is passed back to **GUI.m**.

Mux2Screen.m – This script does all the actual computation of the final interlaced image. It multiplies (per pixel and per channel) the SPMs with images selected by the user, resulting in 28 image matrices, each containing only pixels refracted to the same viewing zone. These matrices are then summed into one, final image matrix – the output picture.

The function also contains simple logic that decides which images will be used multiple times or which will be discarded, in case different amount than 28 images are selected by the user. (Example: If 17 images are selected, only first 14 images are used, each of them is used twice. If 12 images are selected, the 13th and 14th image is created by duplicating the 12th and 11th image respectively and each of the 14 images is again used twice.)

If the user selects less than 6 image files, the script returns an error, informing the user (via Status panel in the main program window) that at least 6 images are required for “Multiple Images” method.

As a last step, the function calls **display_array.m** script described in chapter 3.1. in order to display the interlaced image on the autostereoscopic screen.

4.2. Processing a 2D+Z source

When a 2D+Z is selected as an input method, the workflow is slightly modified, mainly to incorporate the script generating the actual views. However, most of the used scripts remain the same.

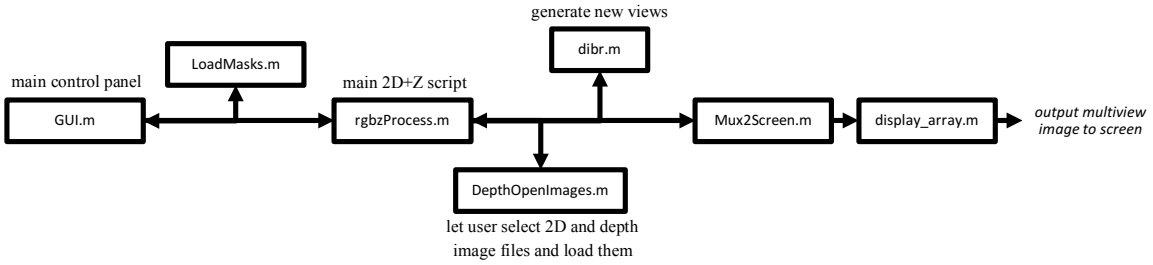


Fig. 51 Scripts used when displaying 2D+Depth image files.

rgbzProcess.m – An intermediate script used in the process of new-view generation. Loads the selected image files, calls both **DepthOpenImages.m** and **dibr.m** and returns the generated images to **GUI.m** in form of a cell array.

DepthOpenImages.m – similarly to **MultiOpenImages.m**, creates two "Open" dialog windows allowing the user to select a 2D image and depth map image files. The files are not loaded directly, only filename and path are passed to **rgbzProcess.m**.

dibr.m – Does the actual new image (new view) generation. Detailed description can be found in chapter 3.3.

4.3. Processing a stereo pair source

A "Stereo" input method uses near identical workflow as with 2D+Z.

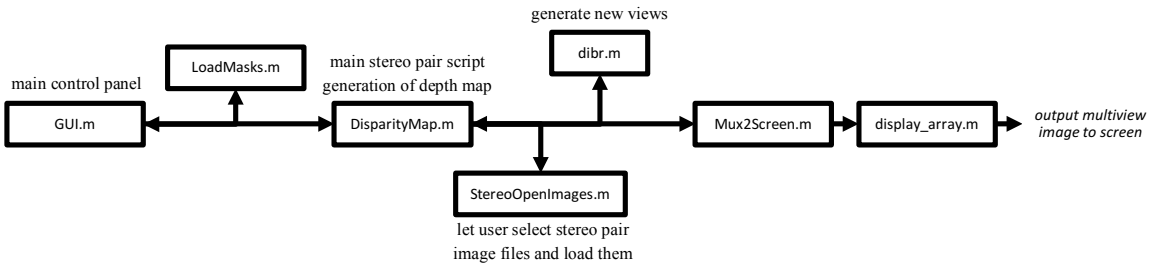


Fig. 52 Scripts used when displaying stereo pair image files.

DisparityMap.m – Similar to **rgbzProcess.m**, only adding depth map creation (see ch. 3.4.)

StereoOpenImages.m – Same as **DepthOpenImages.m** only with different text in dialog window headers.

4.4. Direct displaying of stereo pair source

A different option with stereo pair is to display it directly to the screen without any additional image generation, in much the same way multiple multiview images are displayed. The only difference is that each image (left and right) is copied several times in order to get the 28 required images, resulting in a cell array with the first 14 cells containing left image and cells 15-28 containing right image. With this displaying option, the screen essentially serves as a two-view autostereoscopic screen.

In theory, it should then be possible for the viewer to position himself in a way that he sees the stereoscopic image correctly. In other positions a single 2D view, or an incorrect pseudoscopic image would be visible. Similar viewing zone arrangement was illustrated in Figure 11, although in this case viewing zones are much wider.

Unfortunately, the practical use of this display method is complicated by the presence of considerable amount of cross-talk that blurs the theoretical sharp transition between the two views. The viewer, while positioned correctly, still sees some amount of the right image by his left eye and vice versa, causing some confusion and discomfort. As the viewing zones are wide enough (14 elementary viewing zones wide), using the screen as a 2D display is possible to some degree.

Two different images could be theoretically displayed in this way so that two viewers can see totally different images on the same screen from different viewing positions.

5.1. Problem with YCbCr output in OS X

When a switch was made from Windows to Mac as a lead development platform, suddenly the previously flawlessly working scripts started producing incorrectly looking images, as if there was something wrong with color displacement of each channel.

After double checking the results of Matlab scripts it became clear the results are in order, and the problem lies elsewhere. Comparison of screenshots of the same matlab figure displaying the base image matrix for one angle shown that on display of the Macbook, the image looked correctly but on the autostereoscopic screen, it produced blurred dots instead of crisp single-pixel points.

After some research, it became clear that OS X detects (trough HDMI) the autostereoscopic display as a TV set and sets output color space as YCbCr with some chroma subsampling applied. Since the autostereoscopic screen requires input image with sub-pixel level precision, only direct RGB signal is acceptable. OS X only outputs RGB when a computer monitor is detected. Tested Windows machines did not show this problem, but it is possible it might appear in the operating system as well.

In order to fix this problem, the operating system had to be forced to use RGB mode with the autostereoscopic screen using EDID (Extended Display Identification Data) settings override. The steps required to do that, along with all necessary scripts can be found at [22].

After the settings are successfully changed and the computer outputs RGB signal, the image looks correctly again.

5. Comparison with Dimenco render core

To assess the practical usability of the developed application, its output was compared with the output of the original Dimenco render core, which, being a commercial product, can be assumed as a reference of the highest achievable quality.

The developed application was set as closely as possible to the way the render core works by using 2D+Z source image format (as that is the native input format for the Dimenco solution) and using "Original Philips" view mapping method. In order to make the comparison as relevant as possible, identical image sources for both display tools were used.

Because of reasons detailed in chapter 2.2. of this thesis, it was not possible to directly capture the images supplied to the screen and make detailed analysis of the differences between the two solutions for example by using an image editing software. Instead, a subjective comparison of the produced images has been made. This subjective comparison was not conducted in accordance to any standard test methods, but only as an observation by the author.

5.1. Image quality

Even though the developed application produces usable images, the quality of the image is in some ways inferior to the output of the render core.

Probably the main difference is in the amount of cross-talk, which is handled quite well by the Dimenco render core but is very prominent in the images produced by the application. The other difference is the visibility of the interlacing structure of the image. Images rendered by the Dimenco solution have well defined, though slightly blurred edges resulting in a pleasing (smooth) look, whereas the output from the application is quite rough with sharp, blocky edges and visible interlacing structure, especially where objects in the image have large parallax. This effect is visible in Figure 53 on the objects right below the large central triangle (the enlarged area).

Both of these effects can be explained by the use of slightly different pixel mapping schemes. Dimenco probably uses more sophisticated interlacing patterns that are able to minimize cross-talk and optimize the overall appearance of the output image.

The other probable cause of the differences is the use of different DIBR algorithms. It seems that the Dimenco hardware produces images with less parallax, making the individual images more similar to each other and therefore leaving less room for cross-talk to occur. Unlike the created software solution, the render cores DIBR is also able to produce images with negative parallax, positioning some object seemingly in front of the screen surface. Otherwise the generated images have roughly the same quality in terms of hole filling precision and general impression of depth.

The only observed advantage of the new implementation is that it produces slightly sharper images than the render core. The main reason probably being the fact that the application is

able to process images with larger resolutions (up to 1920x1080, larger images are downscaled) whereas Dimenco uses 960x540 as the default source image resolution and applying upscaling in order to display the image as fullscreen [23].

The performance of both solutions is very similar when suppression of the transition from the rightmost to left leftmost image is compared with the Dimenco core having a slight advantage due to the aforementioned better overall image quality.

Artifacts caused by incorrect viewing distance are practically identical with both solutions, leading to conclusion that Dimenco might use similar stripe-based masking solution.

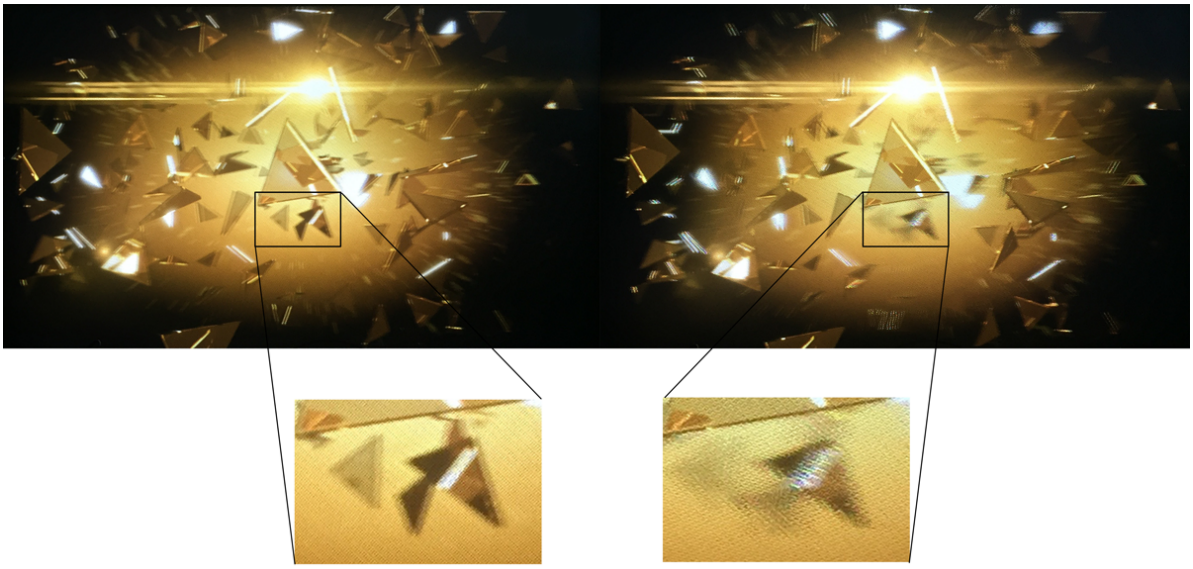


Fig. 53 Photographed screen with the same source image displayed by Dimenco render core (left) and the developed solution (right).

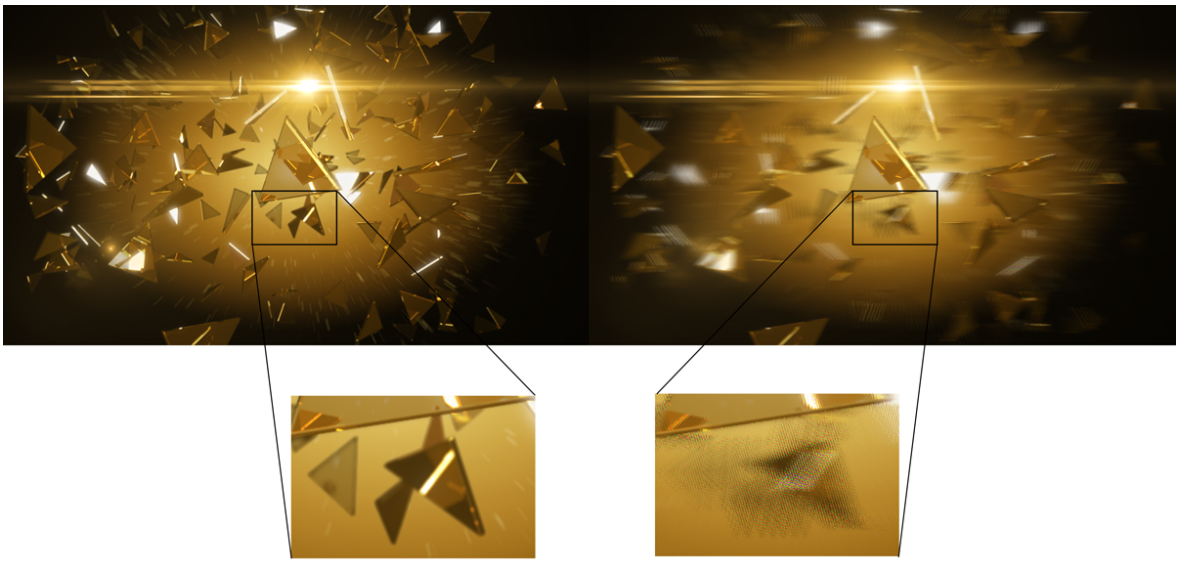


Fig. 54 Comparison of the source 2D image (left) and the interlaced image generated by the application (right).

5.2. Options available

Besides image quality, the developed application differs from the Dimenco solution in different parameters the user is able to control.

As stated earlier, the application was designed to allow the use of some source file formats that are not supported by the Dimenco hardware, namely the direct use of multiple images and of stereo pair. It also has the ability to use different view mapping schemes, while the render box has its mapping scheme hardwired.

While the render core lacks these options, the accompanying Dimenco software allows setting some parameters of the displayed image such as a viewing angle and the intended viewing distance. The former seems to be a simple shift of the numbering of the SPMs, that could be implemented in Matlab quite easily. The latter is discussed in the next chapter.

5.3. Other differences

While the developed Autostereoscopic Display Tool application is able to process images in several common image file formats, the Dimenco Player software used for communicating with the render core requires specialized file formats (*.b3d, *.sbs etc.). While these proprietary file formats can be quite easily created using any bitmap editor, it is an unnecessary step in the preparation of the image.

The second main difference that is not connected to the image quality or user-controllability is the speed of both solutions. While the Dimenco method, being a dedicated hardware solution whose only purpose is to generate the interlaced image, is capable of processing video files in real time and displaying images almost immediately, the developed software solution (depending only on general-purpose CPU with no parallelization) generates the output image for several seconds or even several minutes depending on the source.

6. Suggestions for possible improvements

Although the created application and all the accompanying scripts produce perfectly usable results, there is still a lot of space for improvement. In order to make the application better, improvements in the following areas would be recommended.

6.1. Speed

While the native hardware render core from Dimenco is capable of displaying 2D+Z video in real time, the current implementation of the application in Matlab requires several seconds in order to compute a single frame from multiview image source and up to several minutes to generate new images from 2D+depth source.

Code optimization could at least partially improve the speed calculations. Removal of some potentially redundant *for* cycles or streamlining some of the image manipulation operations could theoretically result in some speed improvements.

Parallel computing seems like an obvious choice for this sort of image manipulation. Unfortunately, Matlabs Parallel Computing Toolbox is only compatible with CUDA (Compute Unified Device Architecture) capable nVidia graphics cards, meaning only users with the supported hardware would benefit from the speed improvements. Such GPU was not available during the development of the application. In order to make the GPU computing available on more machines a switch from CUDA to OpenCL would have to be made by developers of Matlab.

6.2. Optimized memory usage

When tested on an older Windows laptop, the application failed to build the final output image, because it ran out of available system memory. The application generally uses a lot of memory as it stores several high resolution images concurrently. Some optimization in this area would perhaps speed up the computations as well.

Currently the system requirements of the application are at least 4 GB of RAM. A single core processor is sufficient and no graphics card is required.

6.3. Adjustable viewing distance

As described in chapter 2.3.6. the current implementation of the application is optimized for viewing distance of approximately 2 meters. In order to make the application more universally usable, it would be possible to allow the user to set the preferred viewing distance manually.

As the creation of masks used in SPM generation is derived from the stripe-like appearance of SAMs on the screen from a set viewing distance, closer analysis of the stripes would be needed. An equation describing both the angle and frequency of the stripes depending on the viewing distance would have to be created from more detailed measurements. With this

equation it would be possible to generate new masks containing accordingly angled stripes and thus create SPMs for use with arbitrary viewing distances.

Addition of this capability would require generating new SPMs every time a user sets new viewing distance, resulting in additional time required to compute the output image. In order to at least retain current processing time of each image some of the previously mentioned speed improvements would be needed.

6.4. Better algorithm for disparity map estimation

Currently the application uses a basic depth map estimation algorithm that produces results heavily dependent on the source image used. For this reason, the use of a third-party software for the creation of the depth maps is recommended in chapter 3.4.

A new, more robust algorithm for depth map estimation could be used but as stated in chapter 3.4., the time required would be probably significantly higher (again, calling for speed improvements elsewhere in the code).

6.5. More capable DIBR algorithm

The current algorithm used for generation of new views of the displayed scene from 2D+Z sources only creates stereoscopic images with fairly large positive parallax (images appear behind the screen). The code could be altered in away so that it would produce images with both positive and negative parallax and the observable amount of depth could be set by the user.

6.6. More input formats

Only sources saved as individual image files are currently supported. It would be possible to expand the capabilities of the application by adding the option to use side-by-side and over-under images as a source file. Modification of the user interface and dialog window-creating scripts would be required to allow the user to select the used multiplexing method and open only one file. Additional scripts splitting and resizing the multiplexed images would have to be created.

6.7. Standardized subjective assessment of image quality

While the images produced by the application have been compared to the output of Dimenco rendering core by the author, no image quality assessment has been conducted using a standardized method. In order to evaluate the overall usability of the developed application, a standardized assessment of image quality could be used similarly to the one in [19].

7. Conclusion

The aim of this thesis was to create a software solution that would allow the analysis of properties of autostereoscopic displays and of the images they require in order to properly display stereoscopic 3D content. The application would generate such interlaced images and supply them directly to the autostereoscopic screen.

In order to obtain the parameters necessary for correct interlacing, properties of the autosteroscopic display had to be found by experimental analysis of the way the lenticular lenses refract the displayed images. All empirical methods used in the analysis process that led to the discovery of the required interlacing pattern were thoroughly described in this text.

A computer application, named Autosteroscopic Display Tool by the author, has been developed with the use of the obtained interlacing pattern. The application, developed in Matlab, allows its users to process images stored in various formats and display them properly on the autosteroscopic screen.

When compared with the hardware box the application is supposed to replace, some image quality differences were observed. While the developed application produces inferior results than the original hardware-based solution, it is very well usable for casual image viewing as well as for demonstrating the displaying properties of the autostereoscopic screen for education purposes.

The application, being a software solution, also offers some options not possible with the hardware rendering box and with the source code and detailed description of the application provided as a part of this thesis, it can be further developed to offer more options and even produce results of better quality. Some ways in which the application can be enhanced are suggested as a part of this work.

8. References

[1] LUEDER, E., *3D Displays*, 2012, John Wiley & Sons, Ltd,
ISBN: 978-1-119-99151-9

[2] DODGSON, N. A., *Variation and extrema of human interpupillary distance*,
University of Cambridge Computer Laboratory, Cambridge, UK

[3] HEEGER, D., *Perception Lecture Notes: Depth, Size, and Shape* [online], 2006,
New York University, Department of Psychology, [cited on 17-01-2015]
http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/depth/depth-size.html

[4] McALLISTER, D. F., *Display Technology: Stereo & 3D Display Technologies*, 2000,
North Carolina State University

[5] CASSIN, B., *Dictionary of Eye Terminology*, 1990, ISBN 9780937404683

[6] UREY H., CHELLAPPAN K. V., ERDEN E., SURMAN P.,
State of the Art in Stereoscopic and Autostereoscopic Displays, 2011
Proceedings of the IEEE | Vol. 99, No. 4

[7] SIMON A., PRAGER M. G., SCHWARZ S., FRITZ M., JORKE H.,
Interference-filter-based stereoscopic 3D LCD, 2010,
Journal of Information Display, vol. 11, No. 1

[8] WAACK, F.G., Viewing requirements for stereo photographs, In: WAACK, F.G,
Stereo Photography [online], [cited on 19-01-2015]
http://www.stereoscopy.com/library/waack-ch-5.html

[9] INNEMAN, D., *Systém pro demonstraci barevných prostorů*,
Bakalářská práce, 2013, ČVUT Fakulta elektrotechnická

[10] DODGSON, N. A., *On the number of viewing zones required for head-tracked autostereoscopic display*, University of Cambridge Computer Laboratory

[11] Dongxiao LI, Dongning ZANG, Shaojun YAO, ChengLiang LIN, LiangHao WANG,
Ming ZHANG, *Realtime side-by-side to super multi view 3D Display*, 2014,
Zhejiang University, Hangzhou, China

[12] MODDYZ, *stereo-multiview-matlab* [software], 19 Jan 2015, [cited on 7-12-2015]
https://github.com/moddyz/stereo-multiview-matlab

[13] NDJIKI-NYA, P, KÖPPEL, M., DOSHKOV, D., LAKSHMAN, H., MERKLE, P.,
MÜLLER, K., WIEGAND,T., *Depth Image-Based Rendering With Advanced Texture Synthesis for 3-D Video*, 2011, IEEE Transactions On Multimedia, VOL. 13, NO. 3

[14] BROWN, M. Z., BURSCHKA, D., HAGER, G. D , *Advances in Computational Stereo*, 2003, IEEE Transactions On Pattern Analysis And Machine Intelligence, VOL. 25, NO. 8

[15] DIMENCO B.V., *3D Made Simple Flyer*, promotional material,

[16] MELECHOVSKÝ, T., *Měření parametrů autostereoskopického displeje*, Bakalářská práce, 2014, ČVUT Fakulta elektrotechnická

[17] DIMENCO Holding B.V., *Dimenco Display Control Tool*, [software], 30 Apr 2015, [cited on 15-12-2015] https://itunes.apple.com/app/dimenco-display-control-tool/id982491439

[18] DE SILVA, V., *Depth Image Based Rendering* [software], July 2010, [cited on 27-11-2015] http://www.mathworks.com/matlabcentral/fileexchange/28283

[19] KRUPIČKA, M., *Zpracování obrazu pro autostereoskopický displej*, Bakalářská práce, 2014, ČVUT Fakulta elektrotechnická

[20] *YUVSoft* [online], YUVsoft Corp., [cited on 27-12-2015], http://www.yuvsoft.com

[21] CAPETO, U., *3D Stereoscopic Photography* [online], [cited on 2-1-2016] http://3dstereophoto.blogspot.cz/p/software.html

[22] INKSON, M., *Force RGB mode in Mac OS X to fix the picture quality of an external monitor* [online], 2013, [cited on 20-10-2015] http://www.ireckon.net/2013/03/force-rgb-mode-in-mac-os-x-to-fix-the-picture-quality-of-an-external-monitor

[23] DIMENCO B.V., *3D Content Creation Guidelines*, 2011

List of Appendices

Appendix A

Contents of the enclosed CD

- **ADT_Mac** – A folder containing an installer of the compiled application for use with OS X systems.
- **ADT_Win** – Folder with an installer of the compiled application for use with Windows systems.
- **Autostereoscopy.docx** – Text file with this document in MS Word file format.
- **Autostereoscopy.pdf** – Text file with this document in PDF file format.
- **Converted_images** – A Folder containing examples of interlaced images saved from the application.
- **Lab.pdf -** Appendix C – Laboratory task in a separate document
- **Manual.pdf -** Appendix B – Autostereoscopic Display Tool – User Manual in a separate document
- **matlab** – Folder containing all Matlab scripts used in this thesis
    - **MASKS** – A subfolder with 28 basic SPMs saved as PNG files
    - **MASKS2** – A folder with SPMs used for Philips view mapping
- **Pictures** – Folder with all images created for use in this document
- **References** – Folder containing electronic documents used as a reference (see 8. References)
- **readme.txt** – A file describing the contents of the CD in more detail
- **Test_images** – Folder containing sets of multiview and 2D+Z images
    - **2D_Z_images** – A folder with four sets of 2D+depth images (4 2D + 4 depth)
    - **NUM_test** – a folder containing 28 artificialy created images of white nubers on dark gray background. These images are best suited for demonstrations of crosstalk presence and of effects of different image mapping schemes.
    - **photo_test** – A folder containing 28 multiview images photographed to test the used algorithms
    - **photo_test2** – A folder containing 15 photos for use as a multiview source, these photos are better aligned than the ones above
    - **photo_test3** – A folder containing 27 video frames saved as image files for use as a multiview source. Images from this folder are also the most suitable to use as a stereo input.
- **XLS** – Folder with all Excel tables used in this document

Appendix B

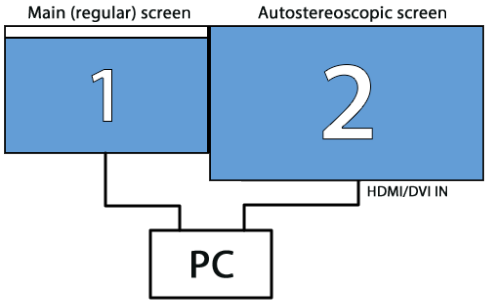
Autosteroscopic Display Tool - User Manual

The Autostereoscopic Display Tool (hereinafter application) is designed to create and display interlaced multiview images on Philips BDL4251VS autostereoscopic screen.

Display Setup

In order for the application to work properly, the autostereoscopic screen has to be connected directly to the computer (via HDMI) and set as a secondary monitor positioned to the left of the main one with top edges aligned. Check the picture below for reference.

The desktop has to be set to extended mode, not to screen mirroring or only to display on the second (autostereoscopic) display.



Also make sure the screen format (aspect ratio) is set to “full” instead of other “zoom” setting. In that way the screen displays true 1:1 pixel reproduction of the source signal with no overscan.

If the output images look incorrectly with too much crosstalk, check that the signal delivered to the screen is in YCbCr format and if that is the case, try to set the output format on the computer to RGB.

Instalation

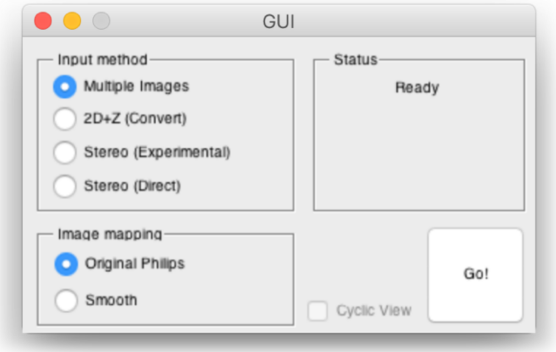
Both Windows and OS X versions of the software have been compiled into an executable applications independent on Matlab installation but as the application is written in Matlab language, it still requires a correct version of Matlab Runtime in order to work properly.

The runtime is installed with the application itself, downloaded from the internet during the installation process. The installation is pretty straightforward and does not require any unusual steps. It may, however, take some time, depending on your internet connection.

The default installation path on Windows is */Program Files/Autsteroscopic_Display_Tool/* and on OS X, the application is installed directly to */Applications/* .

Using the application

The main window of the application is divided into four sections. The “Input method” section, “Image mapping” section, status information area and a program-launching button.



Input method selection allows the user to set the format used as an image source:

Multiple images

Multiview images format used for 6-28 individual image files, each representing a different viewing angle of the displayed scene.

2D+Z (Convert)

2D plus depth image format consisting of two separate image files, one containing RGB “middle” view of the displayed scene and the other its grayscale depth map. This format is converted into a set of 28 artificial multiview images.

Stereo (Experimental)

A standard stereo pair format using two image files, “left” and “right”. This format is converted first to 2D+Z format and trough that into 28 multiview images, effectively creating 28 new views of the displayed scene.

Stereo (Direct)

Uses the same input format as the experimental method but displays it into two wide viewing zones, creating a two-view autosctereoscopic image. This displaying method is not suitable for regular image viewing as it produces several artifacts.

The Image mapping section selects the interlacing method used for the creation of the final output image:

Original Philips

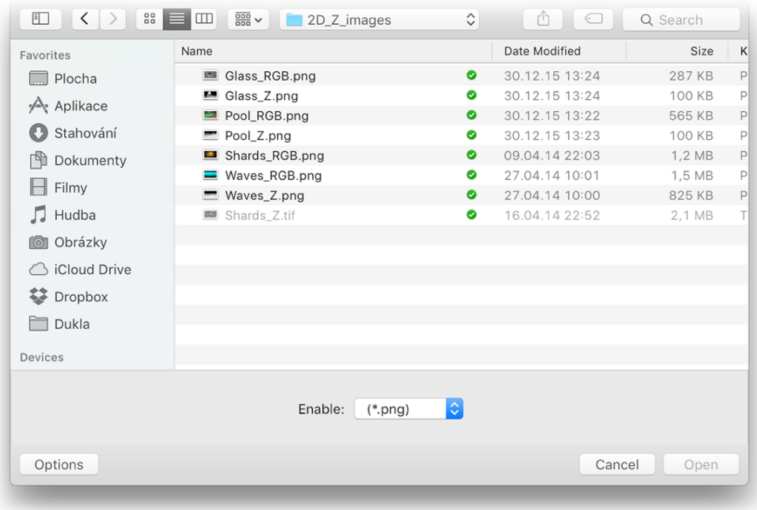
A view-mapping method using only 15 source images to reduce the cross-talk effect and other unpleasant artifacts.

Smooth

A direct view-mapping method using the full range of 28 source images resulting in smoother motion parallax at the cost of additional undesirable effects (grater cross-talk, right-to-left image transition …). The **"Cyclic View"** checkbox modifies the method, using only 14 images to suppress the image "flipping" effect.

With the parameters set, launch the process of creating the interlaced output image with the press of the "Go!" button. A new dialog window opens prompting the you to select the source image files (either all at once in case of "Multiple Images" format or one by one in the other two cases).

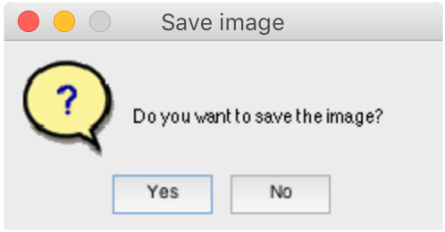
The application supports files in .jpg, .png, .tif and .bmp formats.



After the source files are selected, press the Open button and the creation of the output image starts. The current progress information is displayed in the main window "Status" area.

(Note: The computation may take some time, percentage information in the status area is only approximate.)

After the creation of the interlaced image is done, it is displayed in a fullscreen-sized window and a dialog box opens, giving the option to save the generated image into a .png file.



After the image is saved or the dialog box is closed either by selecting "No" or by closing its window, the application is ready to process new images.

Troubleshooting

Problem: The application cannot be launched, returning a runtime error.

Solution: Try reinstalling the runtime. If that does not solve the problem, try running the application straight from Matlab using the GUI command.

Problem: The application takes long time to launch and to deliver images.

Solution: Wait. For reasons most likely connected to the Matlab runtime, the application is quit quite slow (especially under Windows OS). Using the GUI command straight in the main Matlab software might result in slightly faster response.

Problem: The application reports "Finishing up" but nothing is happening.

Solution: The finishing part of computation takes a long time on Windows machines. Look at memory usage of the application and if it uses several thousands of MB of RAM, the computation is still running. If the memory usage is lower, and it is possible to change the settings in the main application window, an error occurred. If problems persist, launch the application from Matlab and check the command window if it returns the error report.

Problem: The image is generated but appears blocky.

Solution: The image is probably scaled to the size of its window, which is smaller than required. Check that all OS elements (taskbar, dock, menubar etc.) are hidden on the secondary display and the image window covers the full area of the screen. If not, try to maximize it.

Problem: The program returns error when generating depth map from stereo pair.

Solution: If you are running the application straight from Matlab environment and it reports unknown function, you might be missing some of the required toolboxes. Try running the standalone application as the necessary code should be available as part of the Matlab Runtime. Another workaround is to generate depth map using a third party software such as DMAG5[1] and then use it as 2D+Z input. This solution should also give better results as more advanced algorithm is used.

[1] http://3dstereophoto.blogspot.cz/p/software.html

Appendix C

Laboratory task

Introduction

Autosteroscopic screens are able to display stereoscopic "3D" images without the need for any specialized headgear and can even provide motion parallax (change of image with change of viewing position). These screens, however, suffer from higher amount of cross-talk and significant loss of perceived resolution.

Tasks

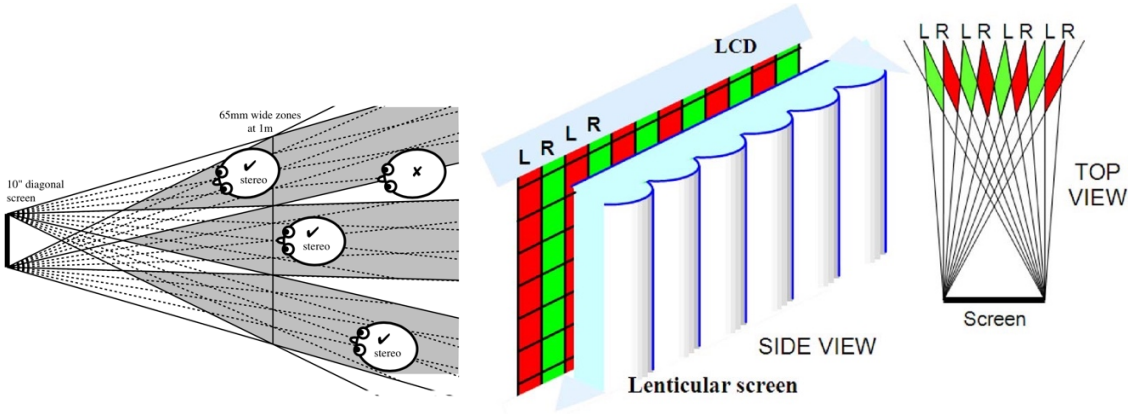
1. Measure visible crosstalk of different image mapping methods
2. Observe effects of different image mapping methods on the displayed image
3. Observe and describe artifacts caused by incorrect viewing distance

Tools used

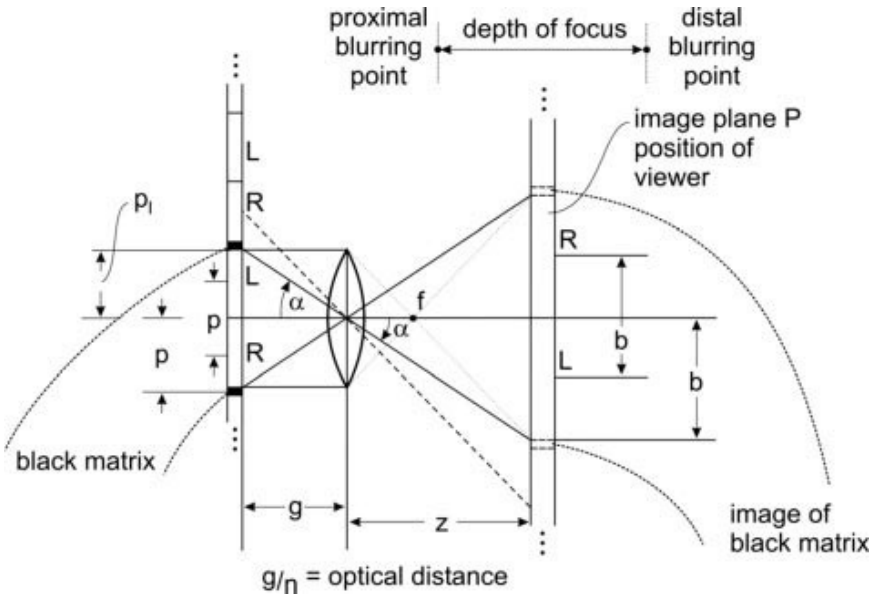
- Computer with installation of Autosteroscopic Display Tool
- Philips BDL4251VS lenticular based autosteroscopic display

Theoretical analysis

The used autosteroscopic multiview display is a type employing slanted lenticular lens technology that use refraction to focus light rays emitted by the screen itself. Displays with lenticular lenses (or simply lenticulars) make use of cylindrical lenses that project images from the FPD screen into repeating viewing zones on an image plane (at a viewing distance). In case of multiview display, a higher number of images (28 in this case) is located behind each lens and the visible views are divided into several repeated viewing zones.



Geometry of a screen with lenticular lenses is depicted in the following figure:

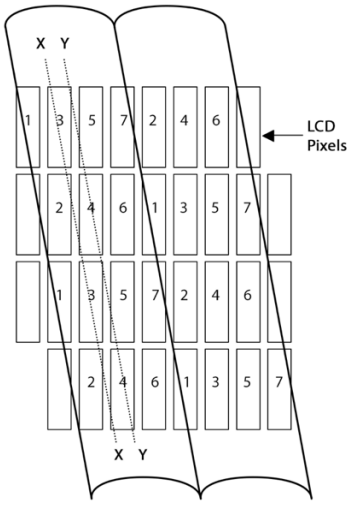


Notice there is some magnification of the pixel pitch *p* into a larger image pitch *b* on the image plane, that should be ideally corresponding with the interocular distance, which is 65 mm on average. This magnification is given by:

$$m = \frac{b}{p} = \frac{z}{g/_n},$$

where b is the image pitch (interocular distance), p is the pixel pitch, z is the viewing distance and g/n is the optical distance, given by the design of the lenticulars.

The black matrix depicted in the figure above presents a problem, as it is also magnified by *m* at the viewing distance and is very disturbing for the viewer. This effect can be suppressed by the use of smaller pixel pitch p1. However, sideward movement of the viewer in the image plane causes the diminished black matrix to become noticeable. A possible solution is the use of slanted pixels in combination with vertically arranged lenticulars, which ensures there are no black lines parallel with the lenticulars. More frequently used is vertical pixels with slanted lenticulars, which has the same effect.



An arrangement such as shown this has several other advantages over simple pixel/lens parallel arrangement. The loss of resolution is distributed to both horizontal and vertical direction as opposed to full resolution in vertical direction and more severe loss in horizontal direction when vertical arrangement is used. This leads to the ability to project more views of the scene with roughly the same loss of resolution – for example a 7 view arrangement with lens width of roughly 3.5 pixels.

Transition between two neighboring views is made smoother with slanted pixels or slanted lenticulars. Instead of flipping into a new image, the „old" pixels fade away while the „new" ones simultaneously fades in, resulting in perception of an increased resolution and more pleasing viewing experience overall. Slanting also helps to eliminate moiré pattern.

The main disadvantage of slanted design is more complicated arrangement of the displayed image, so a specialized hardware or software, such as the Autostereoscopic Display Tool has to be used to interlace the individual images.

Measurement

Task 1 – Cross-talk

1. Use the enclosed user manual to familiarize yourself with the used software
2. Launch the Autosteroscopic Display Tool on your computer
3. Set the input method to Multiple Images and image mapping to Smooth
4. Load all 28 test images found in the 'test_images/NUM_test' folder
5. Position yourself at roughly 2m distance from the screen and observe the image displayed from different angles.
6. Fill the number of concurrently visible values displayed on the screen in the table bellow. (Example: in the image on the right there are 9 values visible).
7. Repeat steps 3-5 with the other two image mapping options (with the Cyclic view option enabled and the Original Philips option).

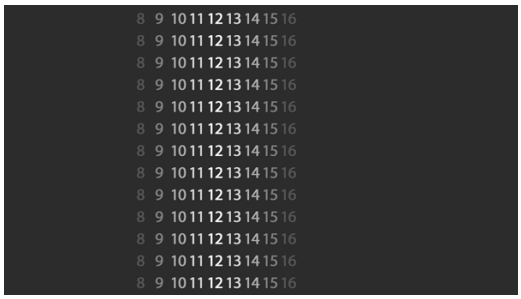


Image mapping	Original Philips	Smooth (Cyclic view off)	Smooth (Cyclic view on)
Visible values			

The filled values represent the number of views that are shown to a position in front of the screen. Ideally, only one value should be visible from a given position (always the one in the middle, the brightest), visibility of the others is caused by cross-talk.

Task 2 – Effects of different image mapping methods

1. Use the same setup as in Task 1
2. Gradually change your viewing position (move form left to right, right to left) and observe the behavior of the displayed image
3. Instead of concurrently visible values, focus on which values are visible and which do not show at all.
4. Use your own words to describe the difference between all three image mapping methods (number of values displayed, difference in brightness displayed values, effects of movement of viewing position etc.):

5. Use different set of images (either provided in 'test_images' folder or your own) to observe the effects of image mapping methods on real multiscopic images.
6. Which method gives the best results? What artifacts does each method suffer from?

Task 3 – Viewing distance

1. Load an image source of your choice. You can try other input methods as well.
2. When the image is displayed, position yourself at roughly 2 meters from the screen. You should see the desired image without any distortions.
3. Move closer to the screen and observe the effect of the change of distance on the visible images.
4. Move further from the screen than 2 meters and observe the effect on the displayed image.
5. Describe the effects of incorrect viewing distance. How is the image distorted? What are the differences between closer/further viewing positions?

References

- LUEDER, E., *3D Displays*, 2012, John Wiley & Sons, Ltd
- UREY H., CHELLAPPAN K. V., ERDEN E., SURMAN P., *State of the Art in Stereoscopic and Autostereoscopic Displays*, Proceedings of the IEEE | Vol. 99, No. 4, April 2011
- DODGSON, N. A., *On the number of viewing zones required for head-tracked autostereoscopic display*, University of Cambridge Computer Laboratory