論文 / 著書情報
Article / Book Information

題目(和文)	3D タッチとジェスチャー検出に基づき、ホログラフィックライトフィールドディスプレーとの直接なインタラクション
Title(English)	Direct interaction with holographic light field displays enabled by 3D touch and gesture detection
著者(和文)	SanchezAlexis
Author(English)	Alexis Sanchez
出典(和文)	学位:博士（工学）, 学位授与機関:東京工業大学, 報告番号:甲第12192号, 授与年月日:2022年9月22日, 学位の種別:課程博士, 審査員:山口 雅浩,渡辺 義浩,熊澤 逸夫,長谷川 晶一,中谷 桃子
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第12192号, Conferred date:2022/9/22, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Tokyo Institute of Technology

Direct interaction with holographic light field displays enabled by 3D touch and gesture detection

IVÁN ALEXIS
SÁNCHEZ SALAZAR CHAVARRÍA
(Student ID 19D18426)

SUPERVISOR

Prof. Masahiro Yamaguchi

CO-SUPERVISOR

Prof. Yoshihiro Watanabe

Contents

Abstract

To enhance the 3D Graphic User Interfaces (GUIs) recently developed, we propose to apply the detection of the light scattered by a user when interacting with the LF reproduced by a holographic LF display, to track its movements and position. Color detection is used to detect the user direction of movement, as well as its position in 3D space. This spatial measurement is used to register the content with the spatial position of the user. This technique is applied to the spatial registration between the displayed content and a commercial gesture-3D tracking sensor. The combination of a registered sensor with an implementation of GPU-based rendering that generates an integral image in real time permits us to create a system where the user can interact with the content using gestures and creating LF based on his movements in real time. The GUI with the registration achieved with the proposed method is evaluated by a usability experiment, finding the scenarios where this registration has a significant benefit in the task completion time, demonstrating the improvement in the GUI usability.

This study presents details about the place this 3D display technique occupies within the existing methods, the fabrication of the basic elements of the screen based in Holographic Optical Elements (HOEs), the setup of the system and the description of all the steps necessary to realize the evaluated 3D GUI.

Acknowledgement

まずは本研究に進めるにあたり、指導教員として 5 年間ご指導いただきました山口雅浩教授に深く感謝申し上げます。東京工業大学で研究を経験させていただき心の底から重ねてお礼申し上げます。また副指導教員として様々なアドバイスを頂きました渡辺義浩准教授にも深く感謝致します。また本研究の基礎知識について色々教えていただいた中村友哉先生にも厚くお礼申し上げます。また研究や博士課程の課題についてアドバイスをいただきました武山彩織先生にもこの場を借りてお礼申し上げます。そして本論文をまとめるにあたり多数のご助言をいただきました熊澤逸夫教授、中谷桃子准教授、長谷川晶一准教授にもこの場を借りてお礼を申し上げます。

加えて本研究について色々教えていただきました柿沼建太郎様、渡辺史顕様をはじめ、お世話になった本研究室の卒業生に感謝申し上げます。最近、コロナ禍で研究室の人間関係が薄くなったにもかかわらず、同テーマを共に研究し積極的に取り組んできた下村杏華さんの興味とやる気のおかげで自分のやる気を新たにさせていただき深くお礼申し上げます。または杜琳遥さんとテーマのディスカッションをさせていただいて感謝申し上げます。

この 5 年間の日本の研究室の風土を一緒に経験し、話し合って議論してお互いに色々学んできた卒業生または現在の学生の皆様にも深く感謝します。

I would like to thank to all the members of the Yamaguchi laboratory that have been with me during this long journey. Specially to Mr. Xiao Chen, Mr. Xiuxi Pan and Mr. Cunyuan Ji, with whom I had many interesting conversations about research and life in general, which made the working days less lonely and more satisfactory. I also would like to thank to Dr. Ilya Reshetouski and his family for being my family in this country and becoming a great friend while struggling both of us to thrive in Japan.

Thank you to all the incredible friends that I have made during these 5 years in Japan and 8 years outside my homeland. It has been a really life-changing period.

Agradezco primero que a nadie a mi madre Hilse Chavarría, por haberme apoyado siempre y jamás haberme negado nada. Este logro, y todos los demás que la vida me permita, son también tuyos. A mi padre José Luis, a mi hermano Irving y a toda mi familia y amigos que tengo en mi México, mi patria. A mi universidad, la UNAM, porque con lo que me enseñó he podido afrontar tanto. Gracias a todos mis amigos y mis seres queridos por su apoyo desde la distancia, en estos tiempos tan complicados de pandemia. Tengan por seguro que siempre están en mi corazón.

Yokohama, Kanagawa, julio de 2022

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.1.1 3D displaying techniques

The purpose of a display is to depict objects and scenes we humans see in our world with as much detail as possible. Two-dimensional (2D) displays are nowadays very developed. They have been integrated into our daily lives thanks to the adoption of smartphones with touchable screens that are capable to recognize several gestures and have other interactive functions. Aiming to represent the real world exactly the way we experiment it, technology to represent three-dimensional (3D) content has been under development since the 17th century, when in 1692 the French painter G.A. Bois-Clair proposed a method to divide a picture in *stripes* for showing a viewer different images as the viewer walked in front of a painting [37], becoming the first proposal of a *barrier method* (see Fig. 1.1). These methods later evolved into *parallax stereograms*, which is a technique that enables a user to see two different views of the same scene from different angles, alternated by the use of very thin stripes. All these methods are early demonstrations of *autostereoscopic* methods, displaying different views of a scene without the user wearing any special hardware. When the system requires that the user adapts his point of view by using a dedicated hardware, then the system is classified as *stereoscopic*.

In 1908, Gabriel M. Lippmann proposed [40] to use a series of lenses in front of a photographic film layer instead of the previously described parallax barrier. He proposed to capture a scene using a lens array (similar to a fly's eye lens, see Fig. 1.2), recording in the photographic film the light rays (or light field, abbreviated as LF in the following) from different viewpoints. The LF of the image would be recovered by impinging a backlight

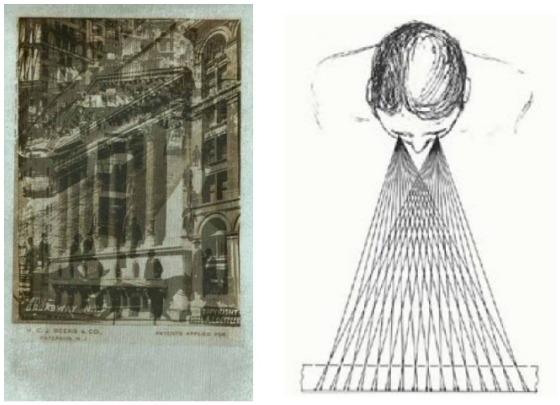


Figure 1.1: Left: Example of a parallax stereogram. This postcard from 1906 reveals two different images depending on the position of the user. Right: Scheme showing the principle of a parallax barrier screen. Source: [51]

on the recorded photograph and back propagating the rays through the different lenses of the array. The result would be an image with a more accurate parallax and with different views. Due to the practical difficulties of implementing Lippmann's theory, it was not tested until the 1950s by Roger de Montebello, who found severe limits on the image depth that could be provided without blurring [8].

Another important breakthrough in 3D image reproduction was the development of holography, which is a technique that is used to record in a physical medium both the phase and the amplitude of an impinging light wave. This technique was first demonstrated by Dennis Gabor in 1948 [19], but the requirement of a coherent light source delayed its widespread use until the invention of the laser and the development of the off-axis recording of holograms by Leith and Upatnieks, who polished the technique to record three-dimensional table-top scenes [8]. Images generated by holography preserve the depth cues thanks to the conservation of the phase information that would normally be lost in a conventional photography exposure.

The development of this technique soon brought the desire of not only reproducing stand-still 3D scenes, but of expanding this technique to reproduce scenes with motion. Since the reproduction of the LF of a scene by a hologram has its origin in the interference pattern created by the reference beam and the object beam, the most straightforward idea to create a hologram moving in real time is to control these interference patterns. This idea gave birth to the branch of holography known as Computed Generated Holography (CGH). Although research on CGH is very active, there are several hardware and software limitations to the creation of a 3D display of the adequate size and with an angle of view large enough for it to be practical. One of them is the large amount of

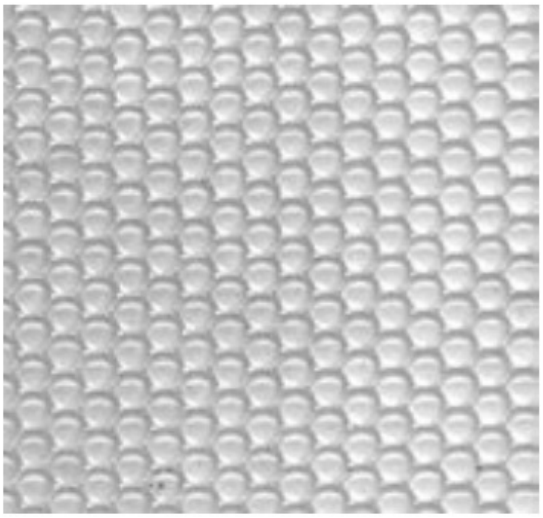


Figure 1.2: A lens array proposed by Lippmann to obtain an integral image. Source: [51]

pixels, and therefore computational resources, required to obtain even a display with a modest size. As an example, if we consider a pixel pitch of 1μm, and a size of 100 mm $\times$ 100mm, the required amount of pixels is 1×10^{10}. Compared to the 8K resolution, one of the newest display formats, which manages on the order of 3×10^{7} pixels, the computational power required for a holographic display of that size is around 330 times the one required for an 8K display.

An alternative to the generation of complicated diffraction patterns to realize an interactive holographic screen, is a technique called *holographic stereogram*. Holographic stereograms angularly multiplex 2D computer generated or optically captured parallax views of a scene [8]. These devices sample a scene more coarsely than interference-based techniques, but that brings the advantage of being able to use common rendering techniques and optically captured images. The coarse sampling helps to reduce the computation time, reducing the bandwidth requirements and enabling a more dynamic interface exploiting the capabilities of hardware available nowadays. In this research, we will employ a holographic stereogram composed by holographic optical elements (HOEs), which encode in a diffraction pattern the functionality of a convex mirror. This pattern is used as the *primitive* diffraction pattern that will be used to angularly multiplex the light and create a volume of light with 3D depth cues that the user can experience without any additional hardware.

As 2D displays developed during the 20th century, the incorporation of touchscreens and gesture recognition functions improved the Human-Computer Interaction (HCI) ap-

plications and turned the 2D screens and the now ubiquitous smartphones into daily life necessities. Now, the development of 3D displays demands that they count with user interfaces (UIs) that can mimic and even improve the interaction 2D displays offer currently.

One step into having 3D User Interfaces (UIs) has been the development of 2D aerial interfaces. These interfaces reconstruct a 2D image in midair using optical devices such as Fresnel lenses [47], Dihedral Corner Reflector Arrays (DCRA) [76] or Aerial Imaging by Retro Reflection (AIRR) [70]. Similar to the case of holographic stereograms, they have the merit of taking profit of the computational resources developed for 2D imaging even in a more straightforward way, since they reconstruct in midair the 2D image created by standard 2D displays using optical methods. They are little by little getting widespread due to its hygienic features and *impressive* look by reconstructing a floating plane in 3D space. However, they are still displaying 2D images and not reconstructing the different captured views.

3D UIs that reconstruct different views of an object have been previously proposed. They rely on the concept of ray-based Light Field (LF), which is inspired on the previously described idea by Lippmann. Care should be put on distinguishing *LF displays* from *holographic displays*, since both of them have different capabilities, different principles, and different advantages. LF displays, although they reconstruct a 3D image in midair, cannot reproduce the amount of depth holographic displays are capable of. On the other hand, they can be updated at a much faster rate than holographic displays.

Interactive LF displays have been proposed before, and there are several examples of them [79, 72, 54, 43]. However, there are only few examples (see [1]) that aim to register the content with the position of the user to improve the realism of the UI. The existing registration methods, although they are capable of matching the position of the user and with the display content, are not automatic methods that can be seamlessly integrated into a UI. Different from 2D interfaces, 3D interfaces have no physical medium for the user to touch or make contact. Therefore, registration of these elements is a challenging task that will be one of the main goals of the present research.

1.2 Motivation

The goal of this research is to give a contribution to the development of a more natural and intuitive way to interact with a machine and its displayed content.

Observing all the modern capabilities we enjoy today after perfecting the 2D touchable interfaces, it is natural to think that enriching the current interaction by adding the missing dimension will offer to the final user more freedom and capacity on whatever its endeavor might be. Techniques to artificially reproduce the visible world around us have existed for a long time, but the technology to update the 3D images and to capture the gestures of a user is more recent.

There exist several ways to input 3D information of a user to a machine. We describe several of them in chapter 2. We should not lose from sight that we are aiming at implementing an interface as user-friendly as possible, avoiding the use of external hardware. Another desirable feature is to have a correspondence between the displayed content and the interaction, therefore the use of the 3D reproduced image is desirable. Since 3D displays are still a research topic, their interfaces are also not standardized. This issue provokes that many 3D interfaces attached to 3D displays of different principles are heavily dependent on the nature of the 3D system used, requiring special sensors and cues that may be useful for the particular setup but make them difficult to use in more general circumstances. Therefore, we would like to propose a method that can be applied to many types of displays that reconstruct a LF as a real image.

In this work, we propose a method that is easy to implement and includes the correspondence between the content and the user interaction. Since the dependency with the display is only the creation of a real LF, it can be thought to be useful for other similar systems that reproduce a LF.

1.3 Thesis structure

The present thesis is divided as follows:

- Chapter 2 provides a review on the most important 3D display technologies and 3D measuring sensors
- Chapter 3 will be about the principle of operation of the display used for this research and how the scattered light detection is used as the principle to develop a touchable interface.
- Chapter 4 explains the color classification, enhancement, and integration into a LF display to measure the 3D position of the finger of a user using a single RGB camera.The content of this chapter is mainly based on a previously published article [53].

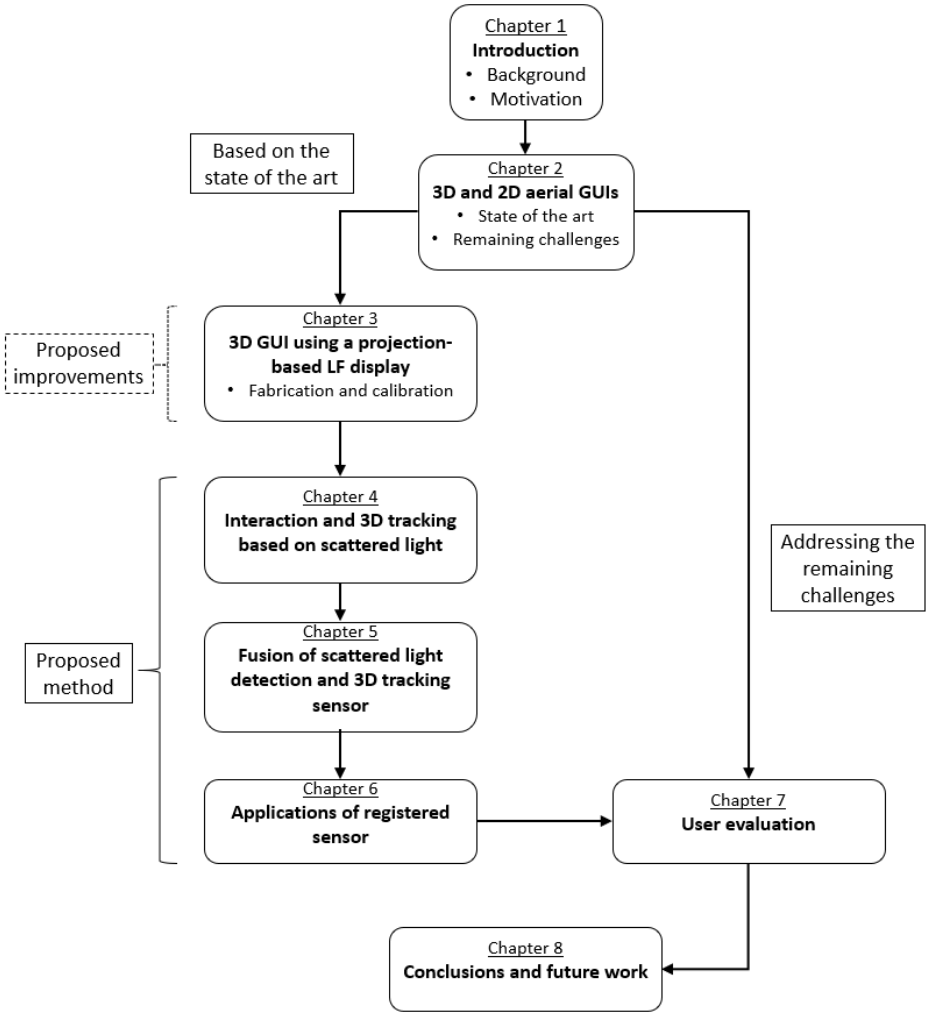


Figure 1.3: Thesis structure.

- Chapter 5 covers the fusion of the scattered light detection with the 3D information provided by a stereocamera based 3D tracking sensor.
- Chapter 6 shows the applications developed using the registration of the GUI using scattered light along with the generation of LF images in real time.
- Chapter 7 provides a user evaluation where we analyze the scenarios in which the registration of the GUI is most beneficial to the completion of a set of tasks. We discuss the advantages and limitations of the proposed system.
- Chapter 8 presents the conclusions obtained by this study, future tasks and a reflection on the future of 3D graphic user interfaces.

A diagram depicting the structure of this thesis is shown in Fig. 1.3

Chapter 2

3D and 2D aerial Graphic User Interfaces (GUIs)

2.1 2D aerial Graphic User Interfaces

Floating images are a first approximation to the development of GUIs more integrated in the 3-dimensional space of users. These devices form a real 2D image in 3D space, which aids in the implementation of gesture interfaces by reconstructing an image in the air, providing more space for movements. They also have the advantage of not requiring hardware in a specific location to show an image in that particular position, which can help in the implementation of AR applications in the future. Aerial GUIs usually work by converging the light rays of a display into a specific 3D location using passive optical components such as micro-lens arrays (MLAs), Fresnel lenses, diffusive screens, retro-reflective arrays (also known as Aerial image by Retro-reflection or AIRR), slit mirror arrays (SMA), dihedral corner reflector arrays (DCRA), among other techniques (see Fig. 2.1).

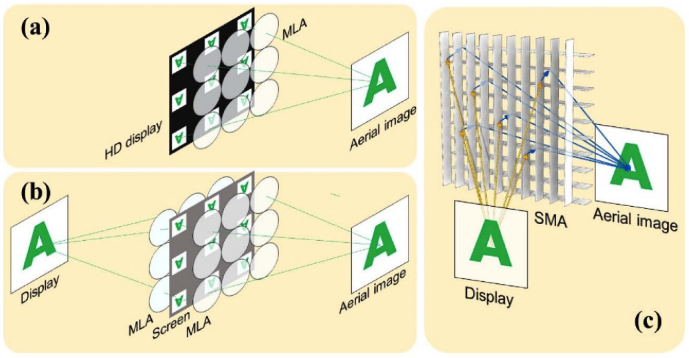


Figure 2.1: Some examples of systems that form aerial images in midair: (a) MLA and high density (HD) display, (b) combination of MLAs, display and a diffusive screen, (c) combination of display and slit-mirror array (SMA). Source: [27]

Since most of the proposed 2D aerial image systems reconstruct a 2D image in midair using as a light source a conventional display, the latency of these devices can be very low even when using current computational power of standard computers [71]. Since these systems are adequate to create hygienic GUIs, their adoption has been progressing and their standardization is already underway [11].

2.2 Autostereosopic 3D image display devices

It is important for this study to locate the nature of the 3D display that we are using, within the extensive research body that covers the 3D imaging and capture field. There exist several ways of creating a 3D display, and they all share the generation of the 3D *perception* by the user. This image might be perceived by only one or many users. Systems that consider only a single user perceiving the depth cues are more common and abundant, since it is easier to develop hardware focusing only in one view than in all the possible views several observers might share simultaneously.

When a system produces an image for a single user, it can be due to two reasons: 1) the system depends on the user to use a hardware piece on the eyes that aids in the creation of the depth cues, or 2) the system tracks the head and/or the eyes of the user and uses that information to direct the imaging resources to a position where the user can perceive the stereopsis effect.

Displays that do not require the user to wear any dedicated hardware, generating the depth cues by themselves, are called *autostereoscopic* displays. Since there is no extra hardware requirement, depending on the display configuration, the 3D effect generated

by these interfaces can be shared by several users.

The generation of 3D imagery without dedicated glasses in the eyes of the user can be achieved by several methods. Some of them include:

- **Parallax barriers**: Displays use a vertical grating that shows odd pixels to one eye and even pixels to another one, creating the stereopsis effect (see Fig. 2.2-(a))
- **Lenticular arrays**: Instead of using barriers, a cylindrical lens can play the role of directing the correct views to the respective eyes (Horizontal Parallax Only - HPO). Alternatively, a lens array can be used to generate the intersection of the light rays based on the principle of integral photography, providing horizontal and vertical parallax (Full Parallax -FP, see Fig. 2.2-(b))
- **Volumetric displays**: Produce imagery by grouping light-emitting physical volumes. This can be achieved by either spinning mirrors at a determined frequency [30], intersecting two invisible lasers to excite a region in space[6], using a varifocal mirror [62], among many other methods.
- **Holography**: Based on reproducing the wavefront of an object by using a diffraction pattern previously generated through exposure or computed by an algorithm based on wave optics.

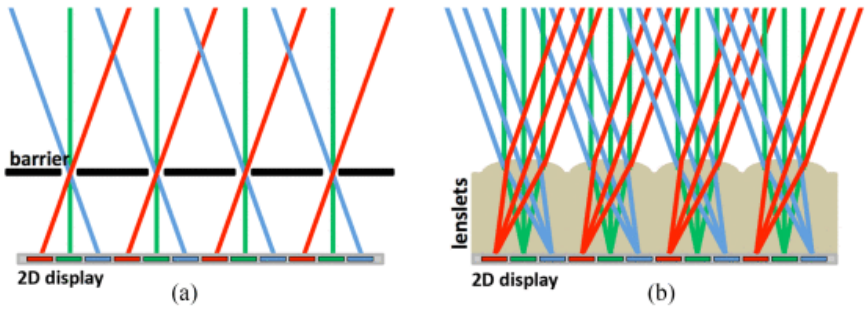


Figure 2.2: Scheme of (a): parallax barrier-based displays and (b): Lens array-based displays

Nowadays, there is a tendency to call *hologram* to any method to produce 3D and floating 2D imagery. This use is essentially incorrect, since a *hologram* is the generation of the LF of an object based on the reconstruction of its wavefront. This reconstruction takes into account the amplitude, wavelength, and phase differences of the light waves emitted by a scene. In that sense, a *hologram* is a more complete representation of the LF of an object, offering to this date the highest resolution of 3D images and the

most natural depth cues of all approaches. However, the complexity of updating the fringe pattern responsible for the generation of LF, makes this approach not suited for a practical and commercially viable 3D display in the near future [58]. The differences between the different mentioned types of interfaces are depicted in Fig. 2.3.

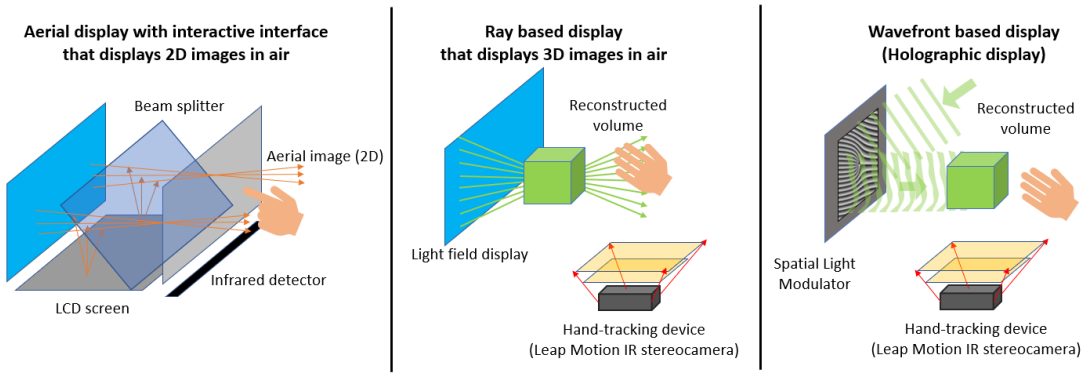


Figure 2.3: Different GUIs, 2D aerial and 3D, that use different principles to create an interface where the user can operate with free hands in the air. However, the 2D aerial is only capable to reproduce 2D content in a floating plane. Although it is potentially possible to incorporate 3D tracking detection to 2D aerial interfaces, the displayed content would remain in a plane.

The technique used by our group is a combination of lenticular arrays and a holographic method known as *holographic stereogram*. Although the display proposed by our group is not *holographic* in the sense that it does not reconstruct the wavefront of a scene, it can be considered as *holographic* or *diffractive* since the impinging angle of light rays is changed using an element produced with a holographic technique.

2.2.1 Holographic Optical Elements (HOEs)

A holographic display usually makes reference to a system capable of updating the wavefront responsible for generating the scene. As mentioned in section 2.2, to update this process is a difficult task and the current achievable sizes of these displays are small and not enough to implement useful functions (see, for example [61]). Therefore, unless a breakthrough development on computational capabilities occurs, this way of creating a 3D display is not very feasible.

Displays based on parallax barriers or lens arrays, on the other hand, can be adapted to work with currently available computational processing power. However, specially in the case of lens arrays, the extra hardware required might be difficult to produce. It might

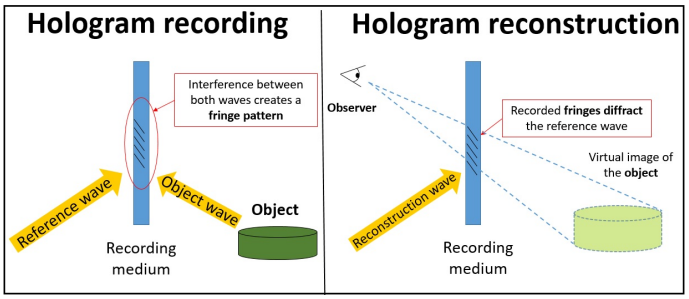


Figure 2.4: Holographic recording and reconstruction of the wavefront of an arbitrary object

also make the device bulky. For that reason, Holographic Optical Elements (HOEs) are considered for these interfaces. Instead of having a lens array, the function that a lens has can be recorded in a diffraction pattern (i.e., the hologram of the wavefront deformed by a lens). This is done by following the holographic process described in section 2.2. Holographic recording and reconstruction are depicted in Fig. 2.4. In ray optics, if we consider a ray of light going through a lens, the output direction of the ray will depend, among other factors, on the shape of the surface of the lens. In the case of HOEs, the direction of the ray will depend on the hologram's fringe structure (i.e., the diffraction pattern).

HOEs have the upside of having the advantages of a hologram (i.e., its thin size and transparency to wavelengths different from the one used as a reference wave). Lens and mirrors, depending on the intended application, can be heavy and difficult to manipulate. HOEs, being thin-films with the same functionality of these devices, can be very advantageous. Since they are films, they can be combined with yet another film to create even more complicated elements (see for example: [74]). On the other hand, some of their downsides are that they can only operate in a narrow bandwidth (corresponding to the coherent light source used to record them) and they are usually used as parts of more complicated optical systems, not being possible to have recorded full optical systems on their own (i.e., there is no "HOE of a telescope") [10].

HOEs, having wavelength selectivity, are transparent to most light that does not fulfill the Bragg condition of the array. Therefore, they can be used as elements of an Augmented Reality (AR) system that combines content in the real world with virtually produced content (see, for example, [38]) (see Figs. 2.5 and 2.6).

In this study, HOEs with the functionality of concave mirrors has been recorded in a photopolymer by using as an object wave the wavefront coming from a camera lens. This provides an optical element that expands the impinging parallel light rays into a

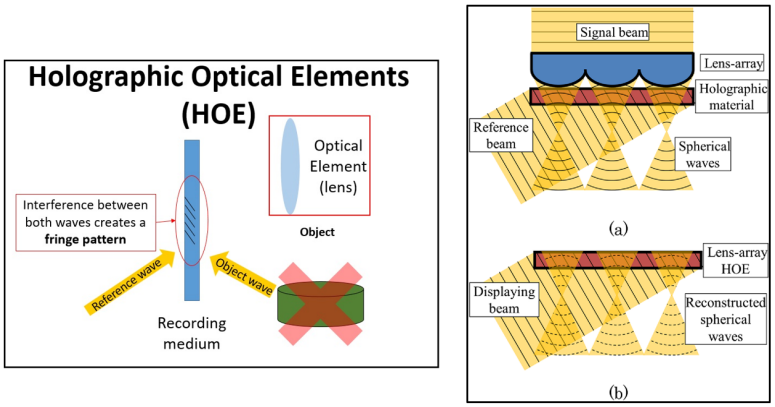


Figure 2.5: Holographic Optical Elements. Left: Use of holographic recording technique to reconstruct the wavefront reproduced by an optical element.Right: Recording of the HOE of a micro lens array with a single shot technique, used in [23]

Figure 2.6: Background combination with the HOE screen to show its potential use as part of an AR system. Source: [38]

fan of rays that depend on the angle of view of the camera lens. By recording several of these elements, we can have an array capable of intersecting the impinged light rays and reconstruct the real image of an object in front of a user (see Fig. 2.3 - center).

2.3 3D input hardware devices

Since the main goal of this research is the development of a user interface to interact with 3-dimensional content, in this chapter we give a brief introduction of some techniques developed for the detection of the 3D movements of a user. The detection of movement in 3D space is something not yet standardized, therefore there is a very large *taxonomy* on the topic. There exist many approaches and many kinds of hardware used to achieve a 3D user interaction. The techniques have also changed a lot in the span of many decades of research. The classification presented here is based on a very extensive survey on 3D interfaces [36], in which more details about hardware, software and detection techniques can be found.

2.4 Extensions of traditional interfaces for 3D interaction

By *traditional*, we refer to the devices commonly used in computers that were initially designed to be used on 2D environments. These devices can be adapted to work as 3D user interfaces, but to work as such they need to be manipulated in unfamiliar ways by the user. Some of them are:

- **Keyboards.** Keyboards are widely used in computer games with 3D cues (e.g., shooting games), having the advantage of providing many buttons (specially the arrow keys) for all the tasks required during the interaction. The obvious drawback is that they are big and impractical for immersive applications. Portable keyboards for use in cellphones have been produced, as well as more portable chord keyboards [36] that emulate the chords of a guitar by pressing different buttons at once to give different commands (see Fig. 2.7-a). According to [36], a user needs a lot of practice to be able to master a chord keyboard.

- **Trackballs.** Trackballs are rotating spheres embedded in the usual mouses that allow the user 2D movement without the need of moving the whole device. Rotating the ball with a finger provides enough 2D cues for interacting with the system. When used in 3D interactions, trackballs are combined with the keyboard to give

more navigational directions (see Fig. 2.7-b).

- **Joysticks.** This kind of interface has a very long history in the computer interaction field. Joysticks differ from mouses in that they continue moving the related cursor as long as they are placed out of the neutral position. They are very used in computer game controllers (see Fig. 2.7-c) and have been implemented as 3D interfaces [36]. Their main problem is that they are not tracked (no orientation and position given), not being very suitable for AR implementations.
- **Desktop 6-degrees of freedom input devices.** This is a mouse designed for 3D design, also named a 'CAD-mouse'. It implements more buttons that enable an adequate 3D interaction in a common desktop computer. They are usually used together with the mouse to choose menu options while designing. Since it is mainly designed to be used in a desktop computer, it is not very used for AR applications (see Fig. 2.7-d).

Figure 2.7: Some traditional 3D input hardware devices to achieve 3D interaction. (a) A 12-key chord keyboard. (b) A mouse with a trackball. (c) Joysticks on a video game controller. (d) Desktop 6-degrees of freedom input device with some of its gestures indicated below. Source: [36]

2.5 3D tracking methods for user interfaces

The interfaces mentioned so far are devices adapted to the shape of the hand that try to make 3D navigation more comfortable to the user by adapting physical buttons. They make the navigation and hardware input more easy, but they do not match the natural behavior that a person has toward the environment.

For example, if a user wants to lift a box within a rendered 3D environment and place them somewhere else using a joystick, the user might need to direct a cursor to the desired box, press a button to let the system know that the user wants to pick that

item, and then proceed to lift the box by moving the lever. This is clearly something the user does not perform within the real world when lifting something, and the user is not capable of performing this task if he does not learn what button should be pressed to select and which item within the 3D environment is the cursor selecting the box. Furthermore, how is the user supposed to drop the box?

All these procedures should be learned by the user, either by trial and error or by learning from a document, a tutorial or from another previous user. In that sense, devices that are able to collect information about the body movements of the user can be used to implement interaction based on gestures and tracking, where the user can profit from its experience in the real world (e.g., grabbing something directly, as usually done in the real world, and not via a series of instructions).

A step further towards implementing that kind of interaction is to track in 3D the movements of a user. To do this, there exist several systems and methods. We provide the next classification:

- **Magnetic sensing.** Based on a magnetic transmitting device and a receiver, some applications to 3D interaction using this technology have been reported [7]. They are accurate to within 0.03 centimeters and 0.01 degree of orientation, but they have the drawback of interacting with ferromagnetic materials present in the room or worn by the user. Besides, they require the user to use a device.

- **Acoustical sensing.** These approaches aim to use the interaction of high-frequency acoustic waves with an object to be able to locate it. The time it takes for a sound wave to travel from the object to one or many receivers (microphones) is computed to determine the object's position. There exist less bulky approaches that use a high-frequency sound in a laptop's speaker and compute the position of a target by using the laptop's built-in microphone. This approach is not very accurate, but it allows the implementation of simple gestures [36]. Other disadvantages of this approach are its interaction with sound sources, acoustically reflective surfaces and low sampling rates.

- **Inertial sensing.** Inertial sensing refers to the use of gyroscopes, linear accelerometers and magnetometers to measure position and orientation, as well as tracking of the position of a determined system. They have been included in ships and planes since the 1950s, but their size in that time limited their application. They had to be implemented into microelectronic mechanical systems (MEMs) before being present in most smartphones nowadays. Although, according to [36], there

have been some research works using these trackers for 3D gesture recognition, they are more used for more rough estimates of position and direction.

2.5.1 Optical 3D sensing

Now that computers and cameras have become ubiquitous, optical sensing based on computer vision is a very wide and active research area. All kinds of approaches using very different types of cameras exist, from high definition to laptop cameras; stereo and depth cameras are also very common. Since there are so many approaches in the literature concerning optical sensing, a first classification of them consists on dividing them into marker-based approaches and markerless approaches [36]. Additionally, depending on where the sensor(s) is (are) located, they can also be classified into *inside-out* in case the sensor is located in the user, or *outside-in* when the sensor is located in the environment in which the user moves (see Fig. 2.8).

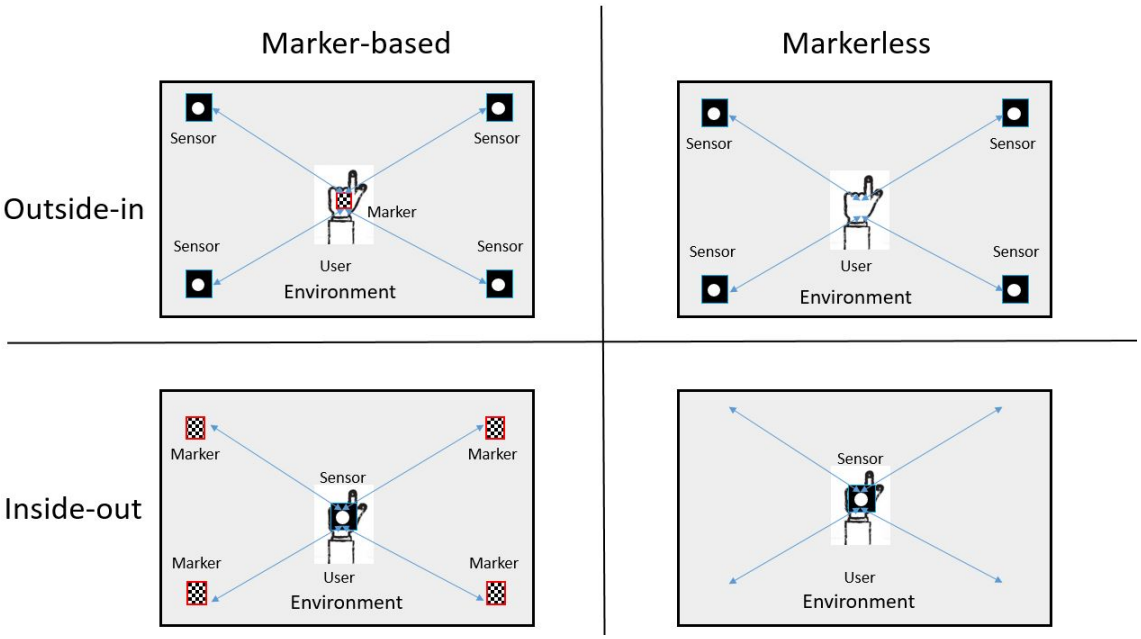


Figure 2.8: Classification of optical sensing approaches depending on the location of the sensor(s) and on the use or not of markers.

When designing a user interface, the ideal approach is to make the user comfortable, not forcing him or her to wear any devices for interacting with the system. As it is illustrated in Fig. 2.8, the only category satisfying this condition is the markerless outside-in one. We will focus on the approaches of this category, in which we can also include the approach followed in this research. An extensive account of the other approaches (i.e., marker-based and markerless inside-out) can be found in reference [36].

There are many approaches applied to 3D interaction that can be classified as markerless outside-in. The most commonly found in recent literature are the ones based on **depth cameras**. Some examples of depth cameras are:

- **Stereo cameras.** This approach aims to imitate the human visual system by using two cameras horizontally separated. The cameras are then calibrated and the depth of a scene is calculated from binocular disparity [36]. Some of these cameras are capable of measuring hand depth and are commercially available, such as Point Grey's Bumblebee Stereo Camera [80]. Recognition of pointing gestures approaches with stereo cameras have also been reported [48]. Maybe the most widespread application of stereo cameras for hand gestures recognition is the **Leap Motion** controller [81]. This controller uses two infrared (IR) cameras along with three separate IR emitters, therefore, it can be classified as a stereo camera approach which also relies on infrared illumination.

- **Structured light.** This approach consists on projecting on a target a known light pattern with a defined structure (fringes, dots or some other patterns). The captured image with the projected pattern gets deformed by the target. When comparing this deformation to the known pattern, a 3D geometrical shape of the target can be retrieved. Some approaches that use structured light to capture 3D information of a scene, although not interaction, can be found in references [29] and [28]. A review on real-time structured light profilometry can be consulted in [63]. A hand gesture recognition method using the structured light-based Intel Real Sense sensor has also been reported in [39].

- **Time of flight (ToF) cameras.** These devices work by measuring the time that light sent towards a target takes to rebound on it and reach the sensor. Since the speed of light is finite, the difference between the impinged illumination and the received one generates a phase shift that is measured and converted into a distance value, effectively estimating the depth of the scene. Just as the structured light-based cameras, the illumination is usually in the near infrared spectrum (NIR). The most used and commercially available ToF camera is the **Kinect** camera to be used with the XBox 360 console, produced by Microsoft Corporation. According to reference [55], just as it happens with the Leap Motion sensor, little technical details have been disclosed by Microsoft about its Kinect camera. However, reverse engineering studies give more details about its functioning [55]. Applications of the Kinect sensor to 3D interfaces have been developed, most notably the one presented in [9]. It should be noted that Microsoft has released its

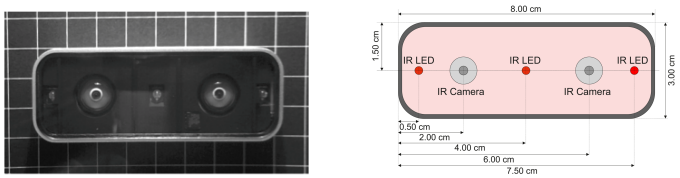
Kinect camera using 2 different technologies: Structured light-based (released in 2010) and ToF-based (released in 2013) [55]. Although the Kinect camera was discontinued by Microsoft in 2017 due to lack of commercial success, the technology was incorporated in 2020 into Azure Kinect, whose main applications are not gaming but robotics and remote health applications.

2.5.2 Leap Motion controller

The Leap Motion (LM) controller will be an important element in this study. Therefore, we dedicate this section to offer more details about its principle. As mentioned in the previous section, the LM controller uses two infrared cameras to measure the position of the hand based on stereo vision. Different from some versions of the Kinect sensor, the LM controller does not use its IR light sources to emit a structured light pattern and measure its deformation to create a depth map, but it rather lights up the scene with these LEDs. A scheme showing the LM structure and the position of its LEDs and cameras is depicted in Fig. 2.9. The required calculations to locate the hands are performed in the host computer of the device, using a proprietary algorithm. The LM hand tracking speed is up to 200 frames per second and the field of view of the sensor can reach to up to 150°[5]. Although that speed is the maximum frame rate achievable by this device, latency variations dependent on the application have been reported previously [18].

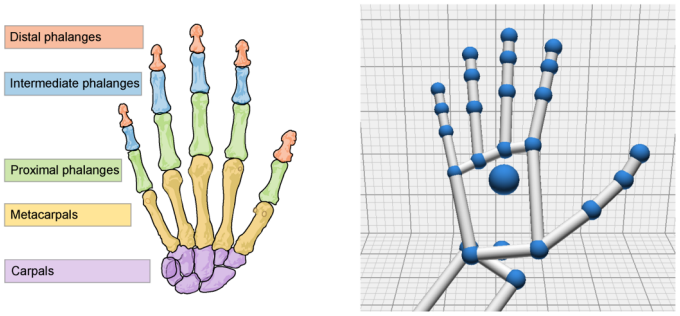
The LM, similar to other marker-less outside-in methods (see Fig. 2.8), counts occlusion of the hands and fingers among its drawbacks. The way to deal with this problem is the inclusion of a skeletal tracking model that provides additional information about hands and fingers. This model assumes a hand has five fingers at all times, what allows them to be detected, although they can be occluded depending on the pose of the hand. Therefore, it is enough for the controller to identify just a part of a hand for it to model the position of all the other fingers. In previous versions of the Leap Motion software, the fingers would be deemed as *vanished* when occluded. The skeletal model adds persistence to the fingers and robustness to the detection [86].

The skeletal model of the LM controller includes the position and orientation of the bones of a hand (see Fig. 2.10). The information of the bones is numbered in a class for its easy integration and use for GUIs.



(a) (b)

Figure 2.9: LM sensor photographed using IR imaging (a), and detailed view of the structure of the sensor (b) Source: [66].



(a) LMC hand model (b) View of detected hand

Figure 2.10: Diagram of the bones recognized by the LM (a), and visualization of the hand model in the LM visualizer application (b) Source: [5]

2.6 Interactive GUIs using 3D displays

After presenting the display techniques (output information) and the capture techniques (input information), we provide a state-of-the-art about some reported 3D GUIs.

Using what we have denoted here as *traditional* interfaces, an interactive full parallax LF display has been demonstrated using wavefront recomposing to increase the angle of view and decrease the image distortion[54]. Real time operation is possible, but the main interaction tool is the keyboard.

Another more recent approach[43] that uses a tabletop LF display along with a Leap Motion controller, captures gestures and modifies the LF in real time accordingly using an OpenGL implementation. Although somehow similar to what we will introduce in coming chapters, the registration problem (matching between the user and the reproduced content) is not addressed.

Something similar about the registration content-user can be said about the research presented in [61] where they demonstrate the generation of the hologram of the path the user traces with the finger using an SLM. However, the gesture is performed in a place different to the LF reconstruction.

An interaction method with LF images where the authors aim for a system robust against occlusion has also been reported in [72], where the authors used a specially tailored light source input along with a retroreflective sheet. Although they also take care of the registration problem and include the measurement of the hand tilt, the system might not be easily adaptable to display more complicated LF images.

3D interaction with registered interaction between the user and content was explored in the system known as Holodesk[21], where the authors used eye tracking combined with depth information provided by the Kinect sensor to create an interface where the user can introduce the hands into the virtual space by placing them under a beamsplitter. Although the user could experience 3D cues such as parallax, they were only visible to one person at a time due to the eye tracking method. In addition, the user had to interact behind the glass of a beamsplitter.

A system similar to Holodesk is one called Vermeer[9], which uses the Kinect motion sensor and an IR sensor. These devices get coupled to a 360° viewable 3D display based on two parabolic mirrors. The system also achieves interaction with the rendered volume when the user touches it, but it is done using an IR sensor that does not necessarily overlap with the light of the displayed content. Besides, the 360° viewable 3D display is based on two parabolic mirrors facing each other, generating a real image floating on top of the mirrors. In the classical demonstration of this kind of interface[2], a small object is placed in the lower mirror to generate its real image above the system. In [9], the image is generated by a spinning stage, a diffuser, and a DMD projector. The system is limited by the size of the parabolic mirrors, which have to be much larger than the reprojected image. This would make the required system too big and bulky for any potential application.

Finally, we have other examples of 3D displays that have been commercially released and can form part of a proposed 3D GUI by using their respective built-in software. The Looking Glass[79], a commercial LF display with a box-like glass structure that enhances the 3D perception of the LF, has a module to develop Leap Motion-based applications. However, it is not possible to directly touch the reconstructed images and the registration problem remains. One example more is the Sony *spatial reality* display[85] released in 2020. It is a parallax barrier based display that uses eye tracking to display the correct views to each eye and give the 3D cues of depth and parallax required for a correct 3D perception. The release of the mentioned displays signals a desire to achieve a 3D display and an interest in developing applications and interactive features with them. Both mentioned commercial examples do not consider registration, which potentially

could improve the acceptance and appeal of a 3D GUI.

There is also an approach based on a Horizontal Parallax Only (HPO) LF display and a Leap Motion sensor that aims to register the content and the gesture of the user [1]. This aim is very similar to the one proposed in this study, but that research does not clarify how to detect the interaction between the user and the LF, which seems to be achieved by a manual input. In addition, this research will also include the real time modification of the LF according to the user position, as well as grabbing and rotating functions that were not addressed in [1].

2.7 Discussion on remaining challenges

2.7.1 Hard-wired systems

Ideas and creativity are very visible in many approaches to create 3D GUIs, but this fact makes the classification of the study very complicated. Several of the proposed systems have been only trialed experimentally to be abandoned afterwards, as discussed in [67]. This is also caused by the nature of many implementations, in which the 3D detection system is hard-wired into the full system, very adapted to the particular display system, and therefore very difficult to be adapted to other scenarios.

On the other hand, in the last decade, the increase of computational power along with the widespread use of imaging sensors, has made the implementation of optical sensing approaches more common. These devices are now relatively inexpensive, and they have good Software Developer Kits (SDKs) that can be used to implement applications with them. The price and software accessibility create a different scenario when compared to the development of 3D interfaces in the latter half of the 20th century. More research development using these devices provides more chances of them to be integrated into 3D GUIs, helping to try them in a variety of scenarios and experimental setups. This will eventually aid into the pursuit of finding an appropriate niche where 3D GUIs can potentially thrive. In this research, we present how the color detection can be integrated to a commercial sensor to increase ts usability.

2.7.2 Content registration

Although there are some developed 3D GUIs that address the problem of content registration, they are very specialized systems, depending on approaches that are not well suited to be mass-produced or commercialized. Content registration is an important is-

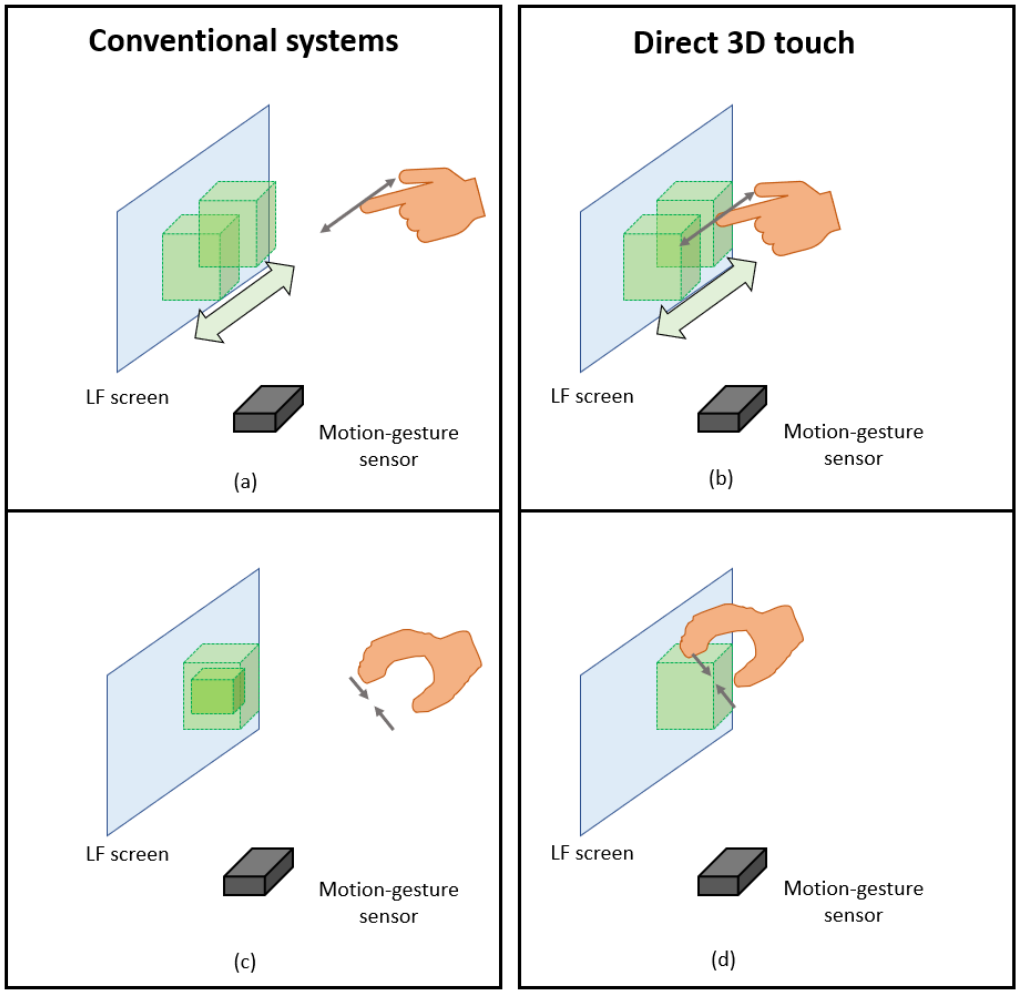


Figure 2.11: Conventional systems compared to direct 3D touch interaction. The user performs gestures a distance away from the displayed content (a and c), while in a registered case, the gestures of the user are matching the content.

sue even for 2D touchable displays, since the failure of adequately registering the position of the user and the content can create user discomfort (see Fig. 2.11).

The *direct* interaction with the reconstructed content refers to interacting with the representation of the content without any mediation between the hands of the user and the content itself. Therefore, no *indirect* ways of interaction such as a widget in the screen or a hardware device (e.g., a mouse) are assumed. As discussed in [34], the *direct* interaction with virtual environments in general (and 3D GUIs in particular) is more useful and preferred when modification of features of the content such as its shape or orientation need to be edited.

2.7.3 Content modification in real time

Although the generation of an integral image to be used for the reconstruction of the LF of an object is by far less computationally expensive than the modification of a Spatial Light Modulator (SLM) in a holographic display of an equivalent size, the real time generation of multiple views of a scene required for the integral image is also an expensive task for a standard desktop computer. There are several 3D rendering programs that are able to generate the views of a digitally designed scene (e.g., Blender, Unity, Unreal Engine, etc.). However, they are not optimized for real time capture of several views simultaneously, and it is not straightforward to use them for that purpose. Depending on the number of views, the required resolution and the computational power at hand, obtaining the integral image of a scene can take from several seconds to even minutes.

There exist published studies (for example, [12]) that adapt some of these programs to generate LF renders in real time. Due to the technical complexity of this task, the research that aims to generate LF in real time based on widely used 3D rendering tools has as an ultimate goal the sole generation of content. Research that aims to generate LF imagery in real time to be used with gestures has started to appear just very recently [43], proving that the topic is of interest within the display community. In this research, the consideration of the content registration will be added to the recently developed concept of a gesture capable LF display.

Chapter 3

3D GUI using a projection based holographic light field display

There are many ways to create an interface where the user can distinguish parallax cues without requiring the use of glasses in the eyes, as mentioned in the previous chapter. A careful distinction should be made between all the different techniques mentioned previously, specially because the manufacturers tend to brand their products as *holographic* when they are not, or *3D* when it is not. In this chapter, we introduce the technique used in this study to create 3D images in midair, which is an advantageous method when considering augmented reality (AR).

3.1 Previous work

One way to create parallax cues without the use of glasses is to use a parallax barrier, displaying different images to each eye of the user. However, if parallax in all directions is to be achieved, then not horizontal parallax barriers but a lenticular array should be placed in front of a screen to generate the required cues in all directions (see Sec. 2.2). Recently, AR is being explored to display information to the users about their environment. Some interesting applications to AR are car and pedestrian navigation, displaying instructions on how to do an activity, acquiring information in a hands-free manner, among many others. For this reason, the display needs to be merged with the real world, combining both virtual and real information. As described in [49], this can be realized by either displaying the real world in a device and modifying it once there, offering to the user the modified view through a device, or by making the display transparent.

Displaying the real world through a camera and then adding modifications tends to disorient the user (apart from being technically complex).The best way to achieve an AR glasses-free system is to have a transparent screen. Most systems that merge virtual and real world scenes depend on a half-mirror with a 2D screen reflecting to it and merging the real and virtual contents (for example, [21]). This does not allow for the creation of 3D content displayed to the user while being aware of the real world, since the displayed views are 2D or projections of 3D renders. There exist some approaches that merge 3D virtual contents with the real world [60, 14], however, they have complicated optical setups that are not easy to reproduce. Using a technique that was first used to expose a holographic stereogram [69], the use of HOEs to create a full parallax LF display was reported in [23]. This method records the spherical wavefronts of a lens array using a photopolymer. Then, the integral image of a scene is impinged on this array, generating a floating image in a transparent screen that merges with the environment. Since the setup consists of a projector, collimation optics, and the recorded HOEs, its implementation is not complicated, and it can be potentially integrated in several scenarios. Different from the implementation first proposed in [23], the approach used by our group is to record each spherical wavefront individually (see Fig. 3.1). This has the advantage of using a chromatic aberration-corrected camera lens for each HOE, but has the disadvantage of being prone to movements of the setup and being time-consuming. Anyhow, we detail the fabrication process and the calibration of the position between the projector and the HOE array in the coming sections.

Since in this 3D display technique we rely on HOEs as the elements that reconstruct a 3D image in midair, we call our display *holographic LF display*. In this sense, the title of our research might also result confusing since, as explained in section 2.2.1, the display we use for this research is not *holographic* in the sense a wavefront reconstruction holographic display is. The display based in HOEs reconstructs the wavefront of the optical element that was used to be record that same element. However, that wavefront does not correspond to a scene but to an optical function we aim to use to create a LF reconstruction. Therefore, it is still holographic, but this phenomenon is related to the generation of LF in a way different to the wavefront reconstruction display (SLM based).

Other relevant studies in which this kind of holographic LF display has been studied are [64], where the view angle is increased by digitally modifying the pattern of the HOE using an SLM; the studies on the calibration of this system made by Jorissen et al.[31, 32], on which some parts of this chapter are inspired, the study on integrating

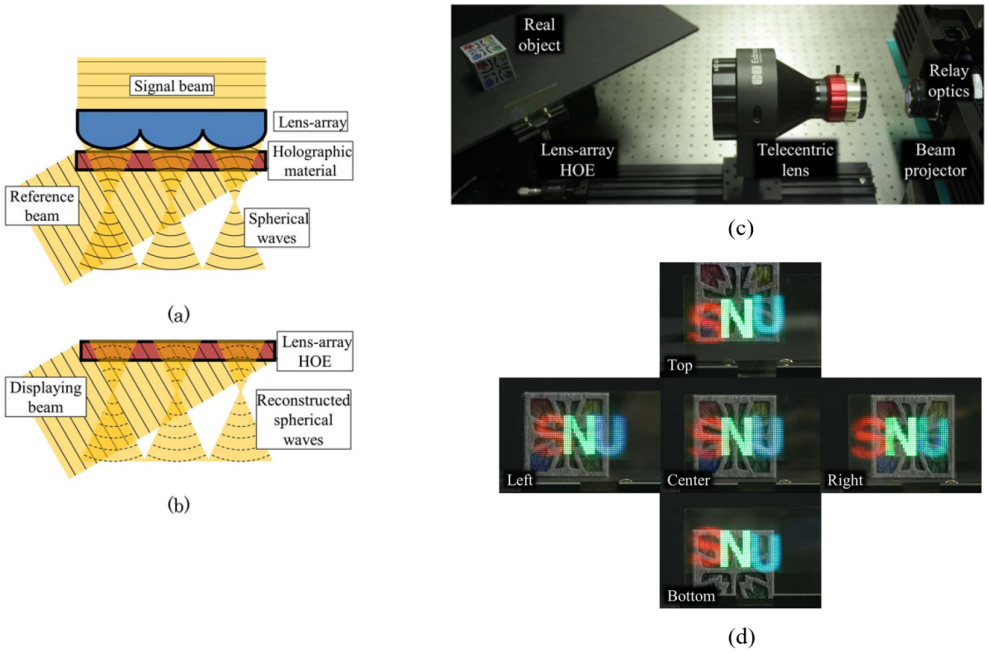


Figure 3.1: HOE-based LF display with a projector and relay optics. One-shot exposure of a lens array (a), reproduction of the spherical wavefront (b), setup to merge virtual and real objects (c), and demonstration of the system (d). Adapted from [23]

the collimation optics to the HOE [25, 26], and certainly the work by our group in which this display was used for implementing a 3D GUI based on scattered light [68, 53]. In this chapter, we will cover some aspects of the fabrication of the HOE based LF display used for this study.

3.2 Fabrication of the HOE-based LF display

HOE is the technique based on holographic principles that allows the recording of diffraction patterns that can be modeled to have the function of an optical element. They are widely used in several applications, because they can perform the function of an optical element with the physical properties of a thin film. Some application examples [33] are hologram memory, holographic printers, 3D head-mounted displays, holographic projection screens, etc.

The kind of HOE that we focus on for this study turns a plane wavefront into a diverging wavefront, the way a convex mirror would do. Using not only one but a dense array of these elements (see Fig. 3.2), intersection between the light rays emerging from adjacent HOEs becomes possible. Therefore, the position of the projector pixel with respect to the center of each HOE in the lattice gets related to the direction each beam will take.

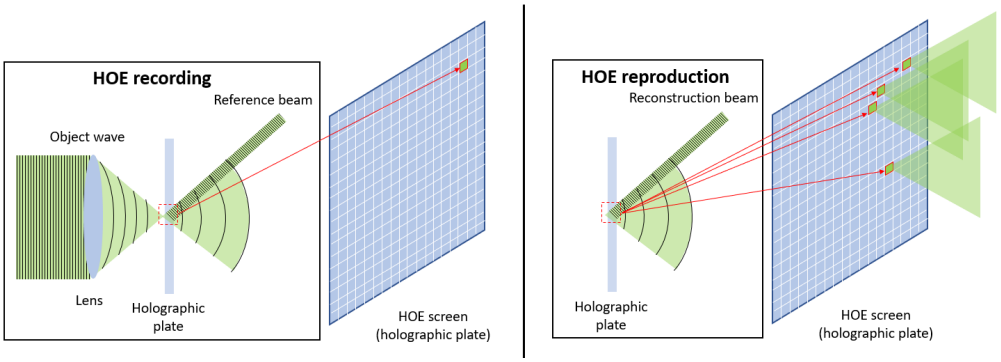


Figure 3.2: HOEs recorded in an array to combine their outputs into a LF reconstruction of a scene.

Using this angle, we can apply the LF theory to generate the LF of a scene (see Fig. 3.3). The use of HOEs as a lens array forming part of a LF display was first proposed in [23], having in mind augmented reality applications.

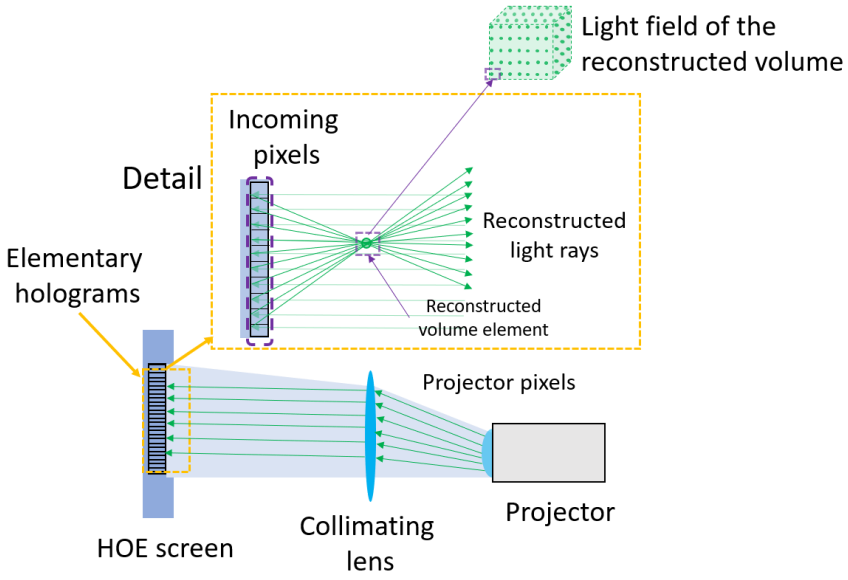


Figure 3.3: Schematic drawing showing the LF reproduction by the holographic LF display

The HOE screen here presented is recorded using an experimental array similar to previously published versions in [69, 68]. To record each elemental hologram in a holographic plate as depicted in Fig. 3.2, the holographic plate is set in an XY moving stage. The experimental setup used for this experiment is depicted in Fig. 3.4 and the experiment specifications are shown in table 3.1.

Red laser	λ=660nm (max power=100mW)
Green laser wavelength	λ=532nm (max power=100mW)
Blue laser wavelength	λ=473nm (max power=50mW)
Glass width	1.85mm
Photo polymer	Bayfol HX-200
Red laser energy dosage	$\approx$15mJ/cm^2
Green laser energy dosage	$\approx$20mJ/cm^2
Blue laser energy dosage	$\approx$25mJ/cm^2
Camera lens	Nikkor 50mm, 1:1.2
Lens of camera used for alignment	Navitar 12x 50486

Table 3.1: HOE screen recording experimental setup.

As shown in Fig. 3.4, the output of a laser source is used and passed through a half-wave plate to change the intensity of the different polarization components and control the intensity ratio between the reference and object waves. After splitting the beam, it is divided and input to two different optical fibers for its manipulation. The use of fibers makes the arrangement of the setup easier, although it induces some losses to the output power of the laser. The output diameter of the beam from both fibers is approximately 1 mm. This output needs to be shaped into a square aperture that matches the desired shape of the HOE. Therefore, a two-lens system is used, where the ratio of the focal lengths expands the size of the beam by 6 (x6 system). By having a larger beam, it can be passed through an adjustable square aperture. The size of the aperture is adjusted to serve as the input of another two-lens system that this time reduces the shape of the beam into an adequate size. The system depicted in figure 3.4 reduces the input beam size by 4 (x0.25 system). The adjustable square aperture has micrometer screws with a resolution of 10 μm, making it possible to adjust the square aperture precisely. The alignment and shape of both beams is verified by using a glass plate with millimeter graph paper and a specialized camera with a zoom lens attached (Navitar 12x 50486) to precisely align both beams. The XY moving stage must also be located perpendicular to both beams in all the screen area, reason why the overlapping of reference and object waves should be tested by moving the stage through all the recording area. Since the reference beam is incoming in a 30deg angle with respect to the perpendicular of the glass plate, the aperture should be corrected in order to obtain exactly a recorded area of 1mm. In Fig. 3.5, it can be seen that using the law of sines, the correction should

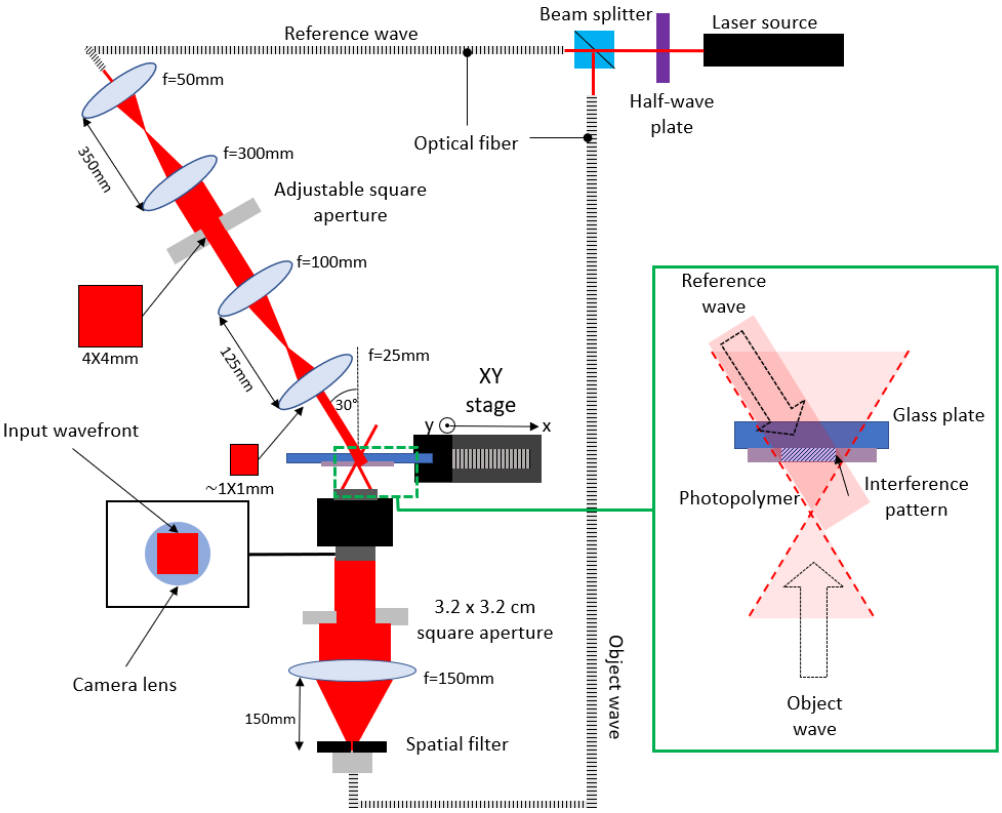


Figure 3.4: Optical setup for the recording of the HOE screen

be:

$$a = \frac{\ell}{\cos\theta}$$

where a stands for the pursued size, ℓ the size of the input reference beam and θ the angle between the reference beam and the normal direction to the glass plate. Therefore, in order to obtain an HOE size with size $a = 1mm$, with an input angle $\theta = 30\,\mathrm{deg}$, the size of the reference beam should be $\ell = 0.866mm$.

The size of the object beam on the polymer is controlled by changing the distance of the camera lens with respect to the holographic plate. As with the reference beam, the correct size on the glass plate is checked using the 12x zoom lens along with a CMOS-sensor camera to monitor both the size and correct overlapping. To finely tune this size, the camera lens is set on a moving base controlled with a micrometer screw.

In previous studies[68, 38], the wavefront for each primary color (red, green, and blue) was multiplexed in the same photosensitive polymer material. That configuration, although it makes the fabrication less complicated, decreases the diffraction efficiency of each of the bands. In this study, we pursue an interface that detects the light scattered by the user to provide the basic signal for a GUI. If the reproduced light does not provide a strong signal, the use of this GUI will be limited to conditions of dark illumination.

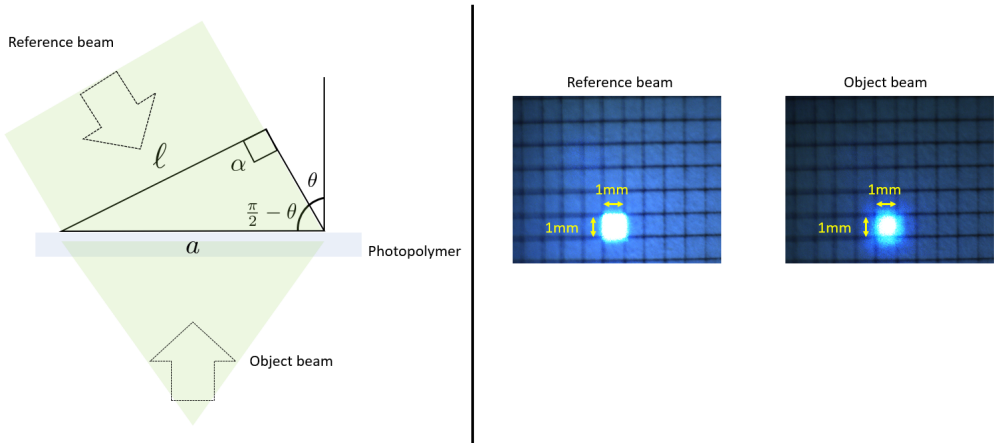


Figure 3.5: Left: Size correction for the input reference beam at a slanted angle. This consideration disregards the refraction caused by the glass plate. Right: However, the actual size is verified using millimeter graph paper stick to a glass plate of the same width as the one used for the HOE screen.

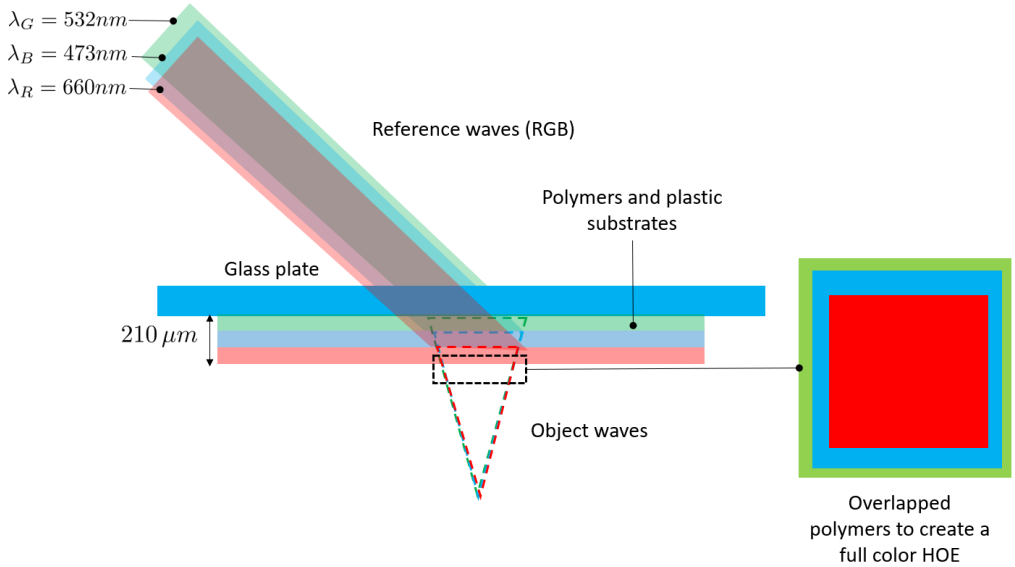


Figure 3.6: Structure of the HOE screen to record each photopolymer with a high diffraction efficiency

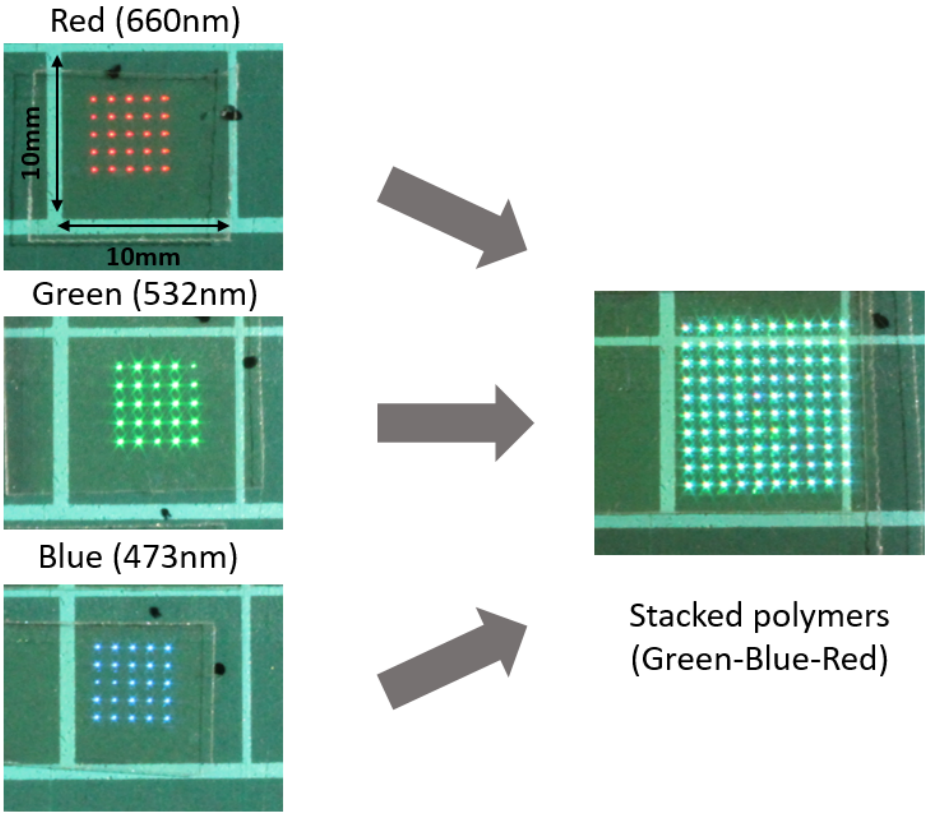


Figure 3.7: HOE samples of each color (size 5×5mm). The conditions for each color were registered to create a stacked sample of 10×10mm size to blend them into white light.

Therefore, a fabrication architecture that uses one polymer for each color was adopted. This use of the Bayfol polymers has been proved to increase the diffraction efficiency in previous studies [65]. Each polymer was recorded in different sessions. Since the green polymer was the one with the highest diffraction efficiency, this polymer was recorded first, so it ends up in the back of the screen. The red, being the one with the lowest diffraction efficiency at the top of the screen, closer to the user (see Fig. 3.6). This procedure also demanded the careful alignment of the reference and object beams in the same position for every color to minimize color aberrations.

3.3 Calibration of the HOE-based LF display

The calibration procedure of the HOE display consists on determining the spatial relation between the center of the HOEs and the projector pixels to achieve the correct reconstruction of the desired LF. This procedure has been reported elsewhere [46, 31]. The difficulty of this procedure has also been reported in [32], where they propose to record on each HOE a special pattern destined for the calibration process. In this research, we describe the approach followed to obtain a calibration with a low amount of error, without requiring the record of a special pattern in each HOE. The precise and practical calibration of this kind of interface is still an open topic, out of the scope of this study. However, we present some differences with respect to the approaches presented in [46] and [31] for future reference. To locate the centers of each of the HOEs recorded in the screen, a camera with a big sensor (5184 $\times$ 3456 pixels) should be used to adequately sample the pixels of the projector. The relation between the camera pixels and projector pixels is obtained by projecting binary descending patterns on the surface of the HOE screen. These patterns are compared with their conjugates using a binary search algorithm that locates the pixel 2D directions (x and y) and saves them in a look-up table (LUT) relating pixel x and y projector directions with respect to the camera pixels. The next step is, with the camera in the same location, retrieve an image where the center of the HOEs can be located using image processing algorithms. This second process computes the LUT relating HOE screen centers with respect to the camera pixels. The third step consists on unifying the results of the previous 2 steps to obtain a projector-HOEs center LUT. The obtained LUT at this point is affected by the deformation caused by the optics of the setup. To correct this deformation, we follow the procedure proposed in [31] of modeling the projector after a reversed camera, and applying the algorithms usually used for camera calibration that compute the internal parameters (focal length, optical center, and radial distortion coefficients) as well as the external parameters (ro-

tation and translation) of a camera (in this case, projector) system with respect to a world coordinate system (in this case, the HOE screen). The obtained matrix is then used to reproject the obtained LUT into the projector plane and obtain a distortion free image.

To summarize, the calibration process can be subdivided itself into:

- Projector-camera LUT computation (binary search algorithm).
- HOE screen-camera LUT computation (HOE center search based on image processing algorithms).
- Fusion of both LUTs to obtain a HOE screen-projector LUT.
- Reprojection of the LUT using camera calibration algorithms, correcting external and internal parameters of the projection system.

3.3.1 Projector-camera LUT computation.

A LUT that indicates the position of the projector pixels with respect to the pixels of a camera is computed. The meaning of measuring the projector pixels with respect to a camera is that this camera will be used later as a reference to detect the HOEs of the array and project the integral photography (LF data) correctly. In principle, it would suffice to project one pixel to the HOE screen, register the projector position and then detect the lit up pixel within the camera screen. However, since the projector resolution is 1920 $\times$ 1080 pixels, this process would require the capture of over 2 million images, which is a very large amount of data. It would also be a very error-prone process since one pixel of the projector is very small, and it might fall into the threshold of noise. To avoid this issue, a projector-camera calibration based on the method proposed in [46], along with some modifications, is proposed.

The used method is based in a binary search algorithm that projects n patterns whose length decreases in powers of 2. n pairs of patterns are used to sort 2^n pixels. Each pattern is projected and compared along with its conjugate pattern to detect the regions that correspond to the power of 2 of the corresponding n pattern. By detecting the largest intensity value in both images and comparing them, a region corresponding to those pixels can be defined. The coefficient 2^m ,where $m \in \{0, 1, 2...n-1\}$, is added to the region identified with the pattern. The LUT of each direction (x and y) is progressively assembled by summing up the contributions of each comparison. An example of how this binary search works in the x axis for a case where the projector has 8 pix-

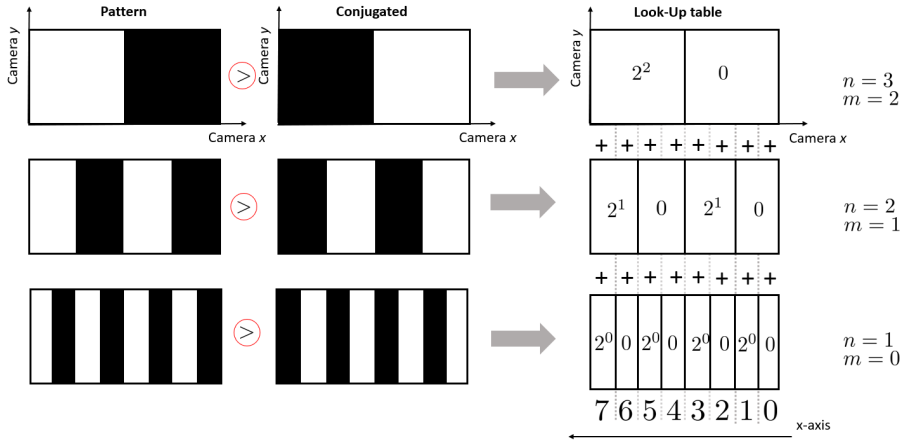


Figure 3.8: Example of the simple calculation of a LUT in the x direction for the case of 8 pixels. Each pattern is compared with its corresponding conjugate and the LUT is progressively assembled. The 2^m coefficient is added whenever the comparison $>$ is true.

els in the x direction is depicted in Fig. 3.8. Using $n = 3$ pairs of patterns, we can sort $2^n = 8$ pixels. Each comparison will add 2^m ($m \in \{0, 1, 2\}$) to the camera area where the comparison between the intensity of each pixel results to be larger in the first projected pattern. Notice that making the opposite comparison (i.e., summing the 2^m coefficient where the conjugated pattern intensity is larger) will reverse the order of the pixels sorting. The same process is applied to the y-axis, obtaining two LUTs that relate the projector-pixel number to the camera plane.

This process is extended to the measurement of the pixels of the projector, where the amount of pairs of patterns is set to $n = 11$ for sorting $2^n = 2048$ pixels. Although this amount is larger than the x direction resolution of 1920 pixels, this is the required amount to sort out the projector pixels using this method. The y-axis also uses the same amount of patterns. The projected parameters are depicted in Fig. 3.9

In this study, we propose for this calibration step two different modifications from the method detailed in [46]: First, the projected binary patterns are not projected on the HOE screen since the projection of these patterns on this reflecting and tiled surface reflects the light unevenly towards the camera. Since the HOEs are designed to deflect the impinged light into a fan of rays, the resulting process of the LUT generation detailed in Fig. 3.8 will result into a capture of binary patterns with many uneven illumination spots. The HOE screen is recorded in a polymer that is stick on a glass plate. This process can result in some air bubbles stuck in the screen that affect the intensity comparison between the pattern and its conjugate. Therefore, we decided to use a glass

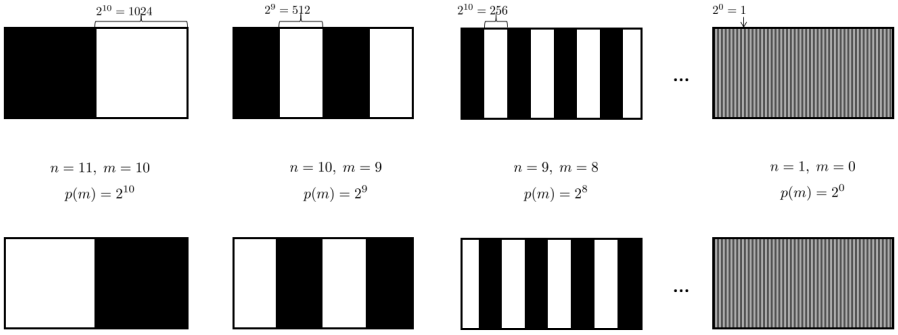


Figure 3.9: Patterns projected for the sorting of the pixels of a projector with a resolution of 1920 $\times$ 1080 pixels

plate of the same width and size as the HOE screen, with a white uniform sticker on it. This white surface provides a uniform and clean surface that results in sharp captures of the binary patterns. Attention should be put on covering the whole area of the screen using this glass plate, in order to have the location of all the required pixels within the HOE screen.

The second modification was applied on the finest projected patterns at the stage of obtaining the smallest pixel pitch of the projector. In reference [46], a 4K projector was calibrated using a camera with inferior resolution, which probably resulted in some accuracy errors. The finest resolution should be captured with a camera with a proper pixel size in order to have an adequate LF reconstruction. In this research, a DSLR camera with 5184 $\times$ 3456 pixels of resolution was used in RAW format.

The blurring of the projected patterns is also an issue in the calculation of the projector-camera LUT step. The collimation optics placed in front of the projector causes the blurring of the smallest pattern widths, specially for the 4, 2, and single pixel-width patterns (see Fig. 3.10).

This problem is solved by dividing the finest calibration patterns in several captures. To avoid the blurring and contrast loss caused by the neighboring lines, the finest patterns are divided as shown in Fig. 3.11. The division of the patterns increases the accuracy of the LUT and avoids mistakes in its estimation. The effect on the contrast enhancement using this method can be appreciated in Fig. 3.12, while the effect on the generation of the LUT in the x direction is also shown in Fig. 3.13.

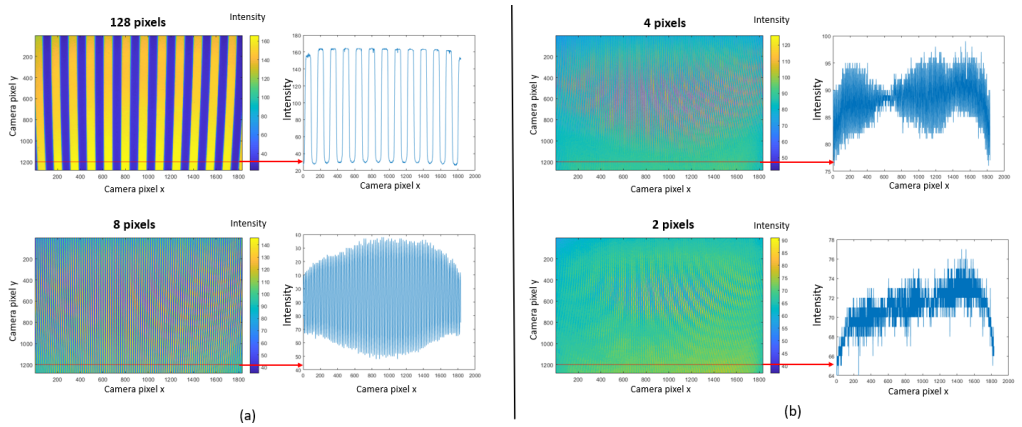


Figure 3.10: Contrast loss with increasing frequency (i.e., smaller pixel period) in the projected pattern. Notice the contrast present in patterns with 128 and 8 pixels (a), while the contrast gets affected as pattern's periods decrease (b).

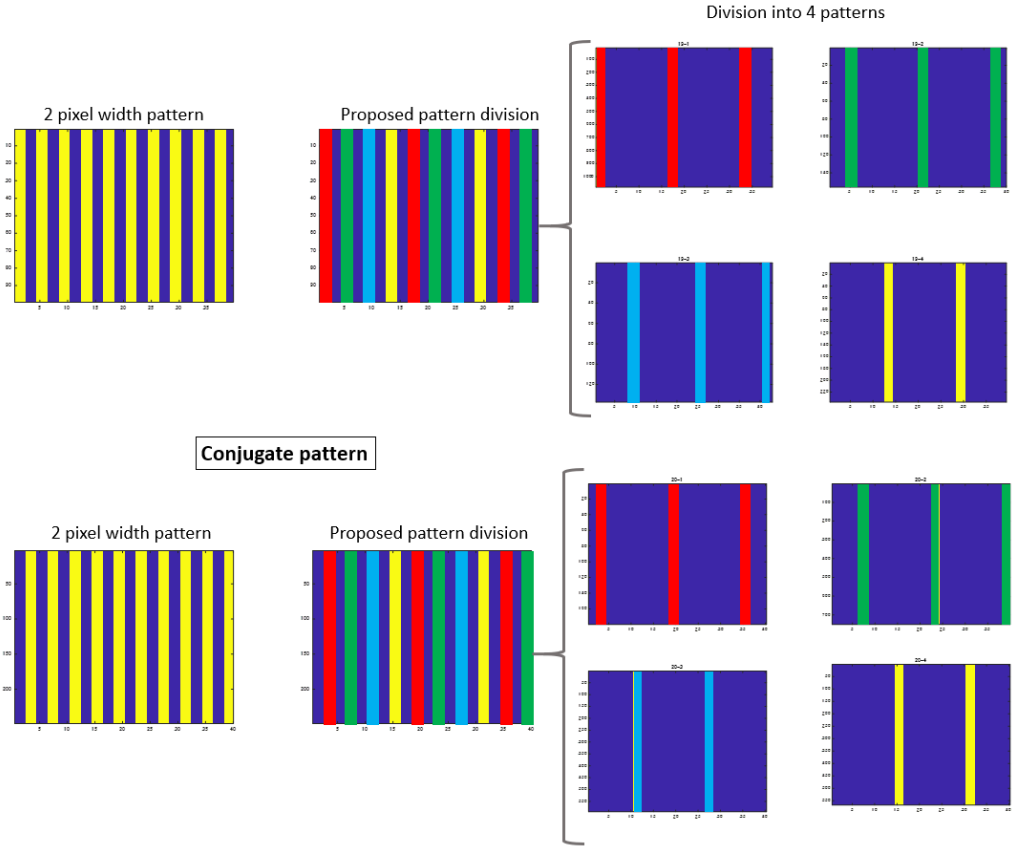


Figure 3.11: Example of proposed pattern split to increase contrast resolution in the calculation of the projector-camera LUT. Each of the corresponding patterns with the same color (normal and conjugate) are compared using separate captures to decrease the amount of error in the pixel measurement.

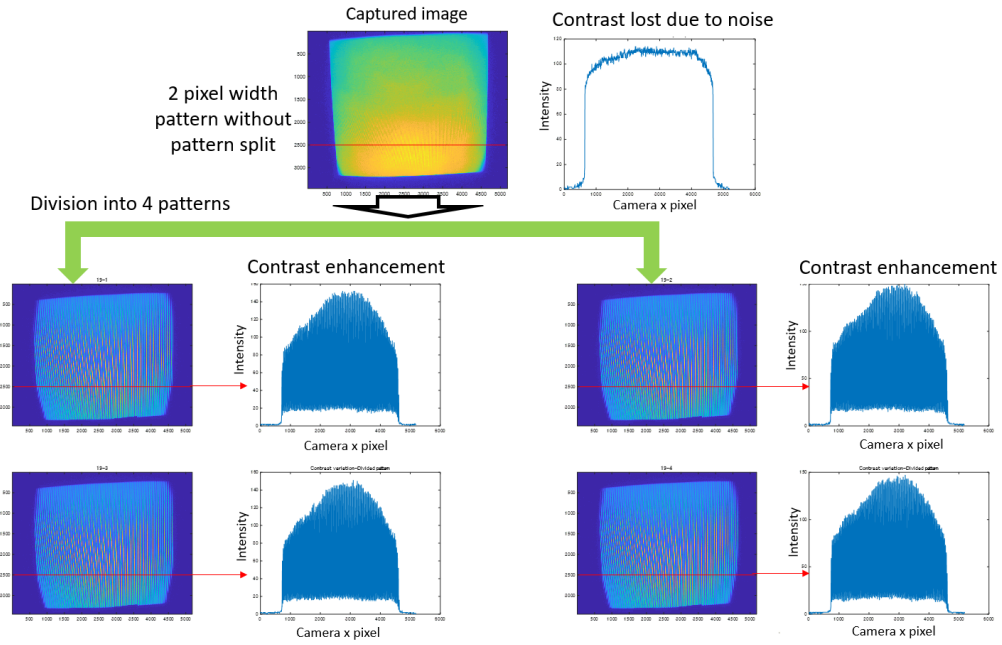


Figure 3.12: Contrast enhancement after dividing the 2 pixel width pattern into different captures. The information, previously not visible, is recovered.

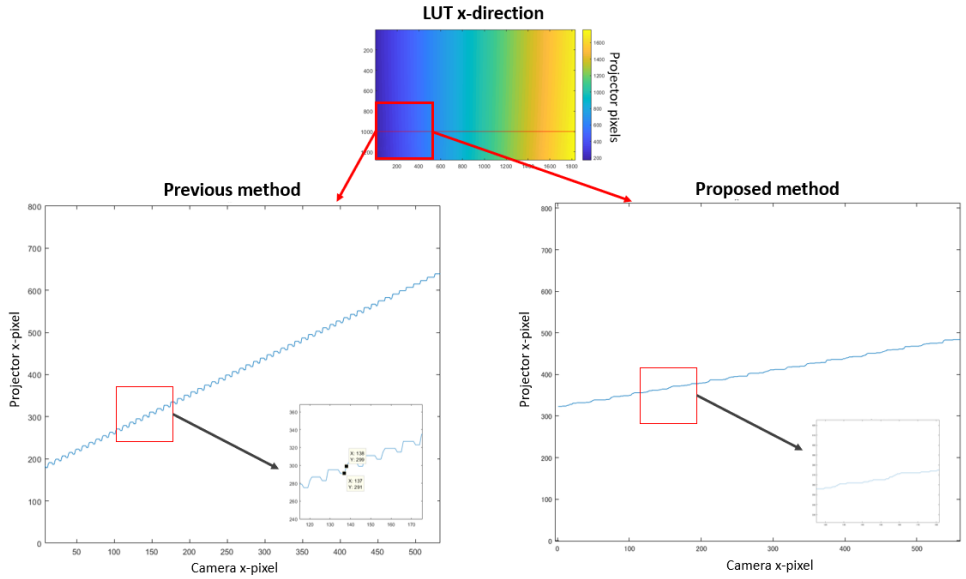
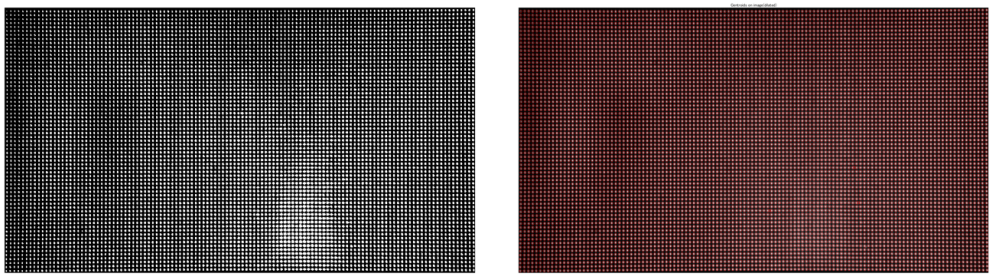


Figure 3.13: Effect on the LUT projector-camera pixels in the x direction



(a) (b)

Figure 3.14: Retrieval of HOE centers: Binarization, erosion and dilation filters to the averaged HOE screen picture (a). The binarized and processed image is input to a centroids search algorithm to retrieve the HOE centers, circled in red (b)

3.3.2 HOE screen-camera LUT computation.

The next step in the calibration process is the generation of the HOE-camera LUT. The process here differs from the one proposed in [46] in the use of the centroid of a binary image that detects the center of the HOE instead of an intensity histogram, which is affected by the camera rotation. By using the centroid detection, along with the projector LUT, the rotation of the camera is naturally included in both LUTs, not requiring a rotation. The detection of the centroids is performed by projecting a white image of near-collimated light to the HOE, with the camera focused on the surface of the HOE screen. This will create an image where the centers of the HOE appear as focused spots if the camera is placed far away from the screen and a low aperture is used, allowing the input of approximately one single light ray into the camera sensor [46].

A background image (average of 10 captures) is subtracted from an averaged image of the HOE screen with collimated light impinged, to reduce the noise artifacts. This image is binarized using a thresholding algorithm. Erosion and dilation filters are applied before the centroids of the remaining spots are obtained. Once obtained, they are sorted according to the known structure of the HOE screen. In this case, the algorithm should correctly find $120 \times 67 = 8040$ centroids. Each centroid is an (x, y) coordinate in the camera space. These centroids are sorted in an array of 120×67 to identify each HOE in the screen (see Fig. 3.14). To increase the accuracy of this process, a polynomial fitting is applied to each line and row of the screen and the location of the centroids is corrected (see Fig. 3.15).

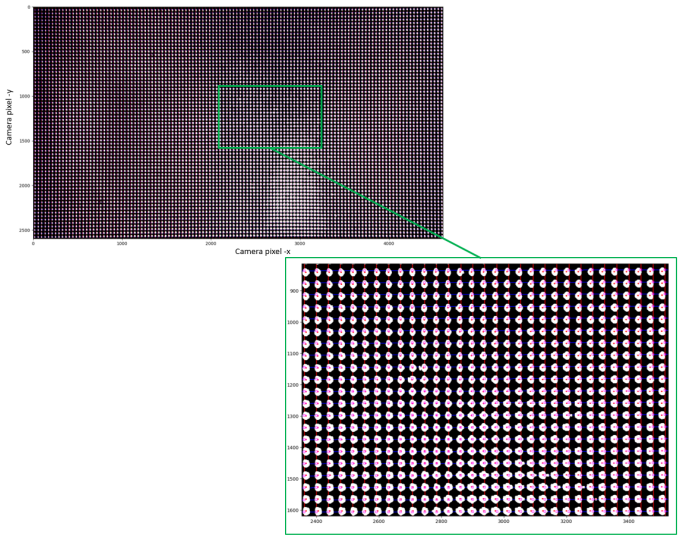


Figure 3.15: Fitting of detected centroids per line (blue lines) and column (red lines) of the HOE screen. The fitting intersection results in a slightly different position from the detected centroid to correct for the possibility of a light ray not coming from the center of the HOE to impinge in the lens.

3.3.3 Fusion of both LUTs to obtain a HOE screen-projector LUT.

The LUTs obtained in the previous steps are both referred to the camera plane. This relation is used to create two new LUTs that relate the HOE screen to the pixels of the projector. However, the obtained result is still affected by noise and by the non-linear effects caused by the optical system (projector lens and collimation optics) that exists between the projector pixels and the HOE screen.

In this stage, an approach inspired on the proposal in [31] is applied. Since the obtained LUT can be thought as the view of a checkerboard pattern seen from the projector, we can regard the projector as a camera from where the corners of a checkerboard pattern (i.e., the centers of the HOE screen) are captured. The checkerboard pattern is a very popular method to correct the non-linear effects caused by the lens of a camera (barrel and pincushion distortion). it also takes into account the difference between the center of the sensor and the optical center of the imaging system (mismatch between projector lens, collimating lens and projector light modulator).

Let (u, v) be a point in the plane of the projector's light modulator, and the point (X, Y, Z) be a 3D point in space. Since we are considering the projector as an inverse camera, we consider that the (X, Y, Z) point is "imaged" by the projector "sensor".

Based on the camera calibration model proposed by [73], the projection model is given by two matrices: intrinsic parameters matrix and extrinsic parameters matrix.

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.1}$$

In Eq. 3.1, (f_x, f_y) indicates the position of the focal point axis in the projector light modulator plane. This point might differ from the actual projector modulator center, indicated by (c_x, c_y). These two points, along with the distortion coefficients, stand for the intrinsic parameters. The extrinsic parameters indicate the position of the projector with respect to the 3D space, and are given in the form $[R|\vec{t}]$, where R is a rotation matrix and $\vec{t}$ a translation vector.

The points in the projector plane would be given by the (u, v) position in the case of a pinhole camera. However, since we have an optical system that introduces radial and tangential distortions, the real coordinates should be corrected into (u', v'). There exist two distortion models of this kind: radial distortion, which is responsible for pincushion and barrel distortion. The other is tangential distortion, which makes the lines coming from the center of projection to wobble as they approach the edge of the image (see Fig. 3.16). The expression for radial distortion is given as:

$$\begin{aligned} u'(u, r) &= u(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ v'(v, r) &= v(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{3.2}$$

In Eq. 3.2, the r variable stands for the distance from the center of the projection plane to the edges of the screen. The tangential distortion model is given by:

$$\begin{aligned} u'(u, v, r) &= u + [2p_1 uv + p_2(r^2 + 2x^2)] \\ v'(u, v, r) &= v + [p_1(r^2 + 2y^2) + 2p_2 uv] \end{aligned} \tag{3.3}$$

The camera-calibration algorithm proposed by [73] consists on giving as an input 3D points(X, Y, Z) with corresponding projected points (u, v) to estimate the matrix R, the translation vector t, the parameters (f_x, f_y), (c_x, c_y) and the distortion coefficients $(k_1, k_2, k_3, p_1, p_2)$. The final undistorted point (u', v') is then set in the projector, where

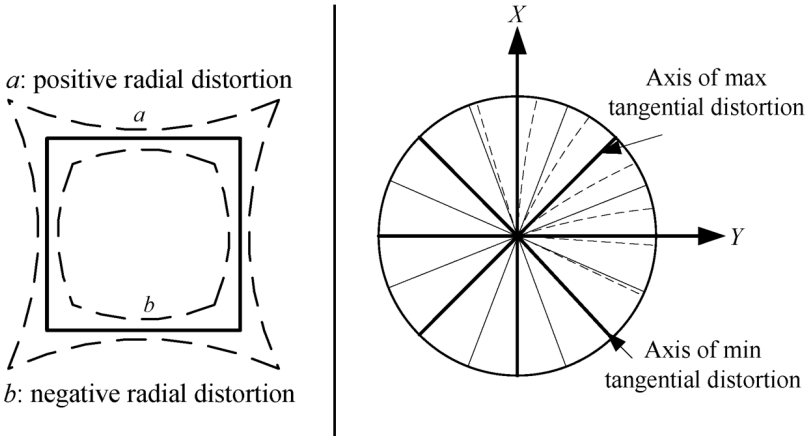


Figure 3.16: Left: Radial distortion, responsible for pincushion (*a*) and barrel distortions (*b*). Right: Tangential distortion, which bends straight lines coming from the center of projection

it will be physically distorted by the optical system to match the position of the center of the HOE.

The optimization process proposed by [73] is usually implemented by taking as an input several images of a checkerboard pattern. However, in this case we take as an input the detected centers of the HOE screen, and we define them as 3D points in a plane with $(X, Y, Z = 0)$, as it is done in the applications of this algorithm. The implementation of OpenCV was used for this study.The effect of this procedure can be appreciated in Fig. 3.17. The image pointing to the estimated HOE centers seen from the projector is shown in 3.17(a). The distortion map, taking into account all the described effects and computed with the procedure mentioned above, is shown in Fig. 3.17(b), while the undistorted view is in Fig. 3.17(c).

3.4 Final device specifications

Once the calibration between the HOEs and the projector pixels has been achieved, the system is ready to reconstruct the LF of a scene. As it is described in the literature [68], the LF display is not capable of displaying a very strong depth sensation because of the lack of coherence in the reconstructed wavefront. However, parallax effects are visible and a shallow depth (around 3 cm) can also be perceived by the user, which allows the evaluation of the use of this kind of interface in the next stages of this study. A scheme depicting the setup is shown in Fig. 3.18 and the specifications of the system used for this research, unless otherwise indicated, are shown in Tab. 3.2. In Fig. 3.18, a mirror

Dimensions of the HOE screen	120 × 67 mm
Dimensions of each HOE	1 × 1 mm
Resolution of the projector	1920 × 1080 pix.
CPU	Intel Core i7-8750H
Clock speed	4.10 GHz
RAM memory	32 GB
GPU	Nvidia Ge-Force GTX1060 6GB
RGB camera	Point Grey Flea 3 1/3" Color CMOS
Camera native resolution	1328 × 1048 pix.
Camera lens	Edmund Optics 18mm f/4-f/12, Mod. 54857
Captured image size	640 × 320 pix.
Software versions	Windows 10 64-bit Python 3.6.2 OpenCV 4.5.1 OpenGL 4.2.0

Table 3.2: Specifications of the system.

is used to place the projector behind the screen and make the LF reconstruction more accessible to the user. However, the light intensity of the reconstruction is affected by this configuration, which should be taken into account at the time of performing the color measurement explained in the next chapter.

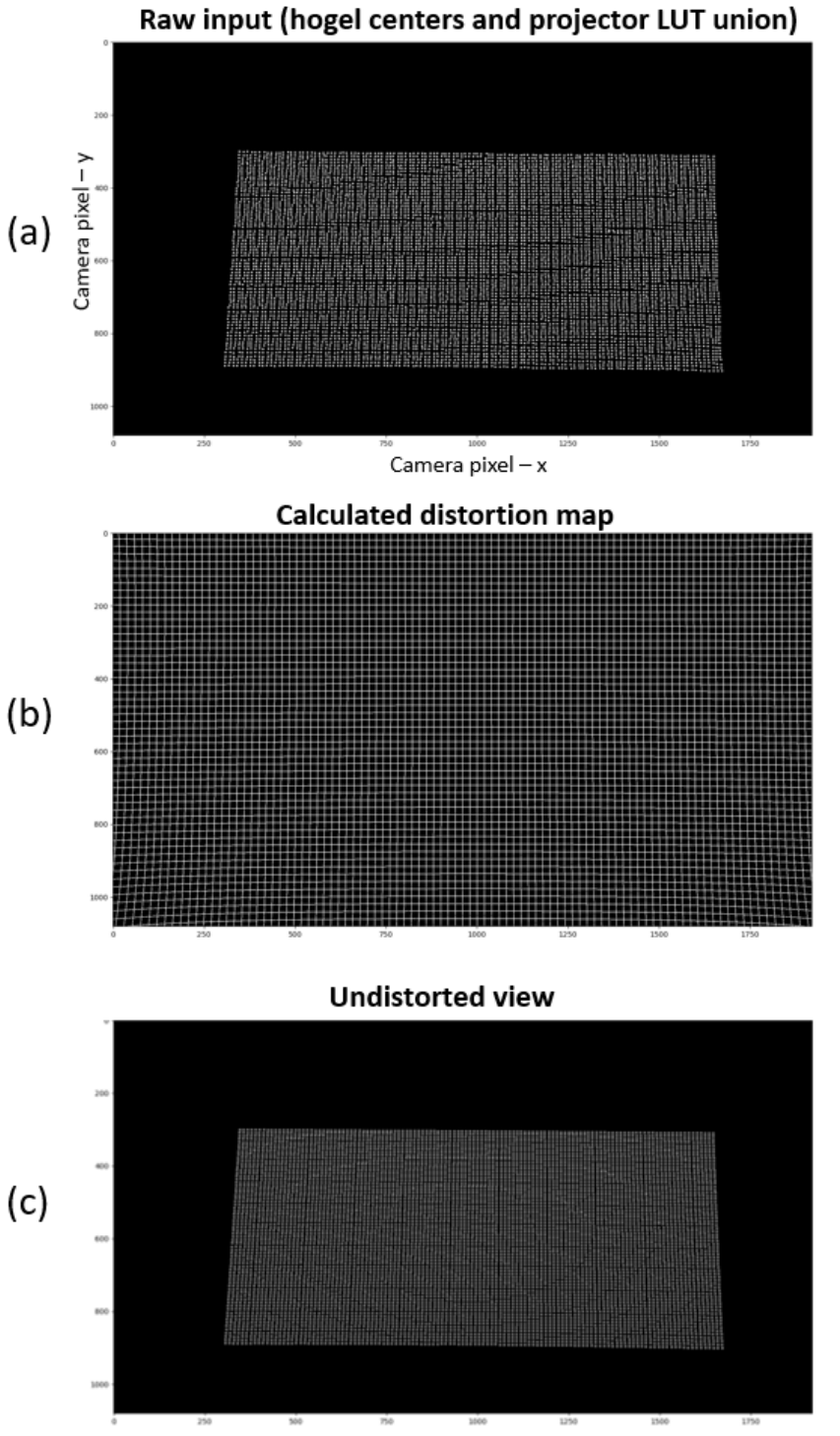


Figure 3.17: Distortion map and its effect on the view of the HOE screen from the projector. Raw input (a), computed distortion map (b) and HOE screen centers with the applied distortion map (c).

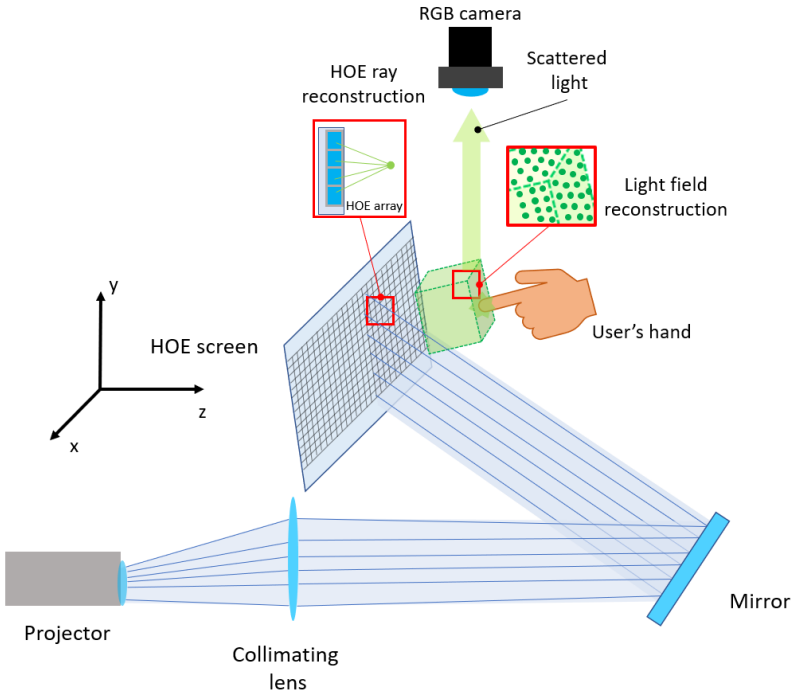


Figure 3.18: Schematic depiction of the display setup used during this study. The mirror might or might not be used.

Chapter 4

Interaction and 3D tracking based on scattered light

4.1 Principle

In order to realize an interface in which the direct interaction with the displayed content can be detected, the proposal of directly using the light scattered by the user when touching the 3D content has been reported by our group [68]. A 3D UI involving a projection-type HLF display (see principle in Fig. 3.3) was used to create a real image that the user can touch with its bare hand. The scattered light by the user is detected by an RGB sensor located behind the transparent screen, eliminating the matching problem and creating a more natural UI (Fig. 4.1 (a)). We refer in this study as light-field touch sensing (LFTS) to the concept of using the light of the reconstructed image as an input signal to create a UI. A related method that uses the LFTS technique has also been reported in [72], where a system using LF for both projection and capture was proposed as the basis of an occlusion-robust hand-tracking method. However, the current LFTS systems are not able of tracking or updating the LF in real time. In addition, one single RGB sensor was used in the system in [68], only capturing a 2D projection of the 3D space. Although it is possible to use other approaches (e.g., stereo camera) to obtain depth information, another strategy to measure the depth information using the LF projection can be adopted.

The method described in [68] consisted on an RGB camera located behind the display (Fig. 4.1(a)). The interaction was realized by subtracting the background from the image captured by the camera and detecting the maximum value between two colors

(blue or green). This interface was demonstrated with a static 3D image of two buttons, in which a binary response (i.e., touch or no-touch) was detected by the system as a preliminary concept verification.

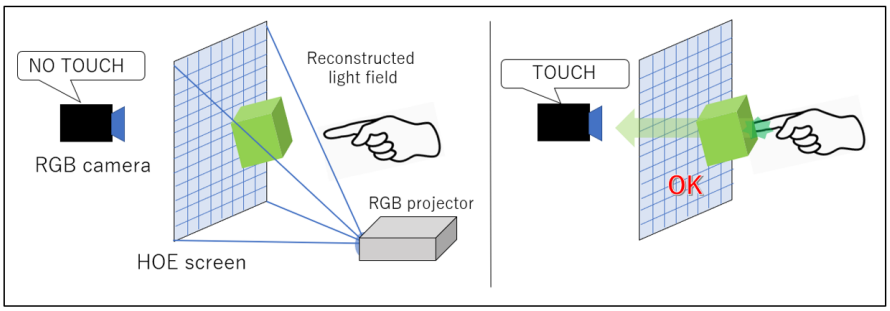
The content of the present chapter is mainly based in a previously published research article [53]. Here we add more details and discussion to the matter exposed in that paper. To summarize, the main advantage points of this method are:

- Direct method that resolves the registration problem between the user and the displayed content. Using scattered light detection, the *touch* signal is only detected once the user enters in contact with the reconstructed LF.
- Use of color information to detect the movement of the user in the axis perpendicular to the RGB camera. This method enables 3D tracking of a user using a single camera, avoiding the use of more complicated techniques like Time of Flight or stereo cameras.
- Compensation of the color of the finger of the user by using a previous sampling of it, increasing the robustness of the detection.
- In addition, we implement two different ways of modifying and updating integral images to generate real-time moving LF images. Coupled with the high-speed implementation of the undistortion algorithm, it allowed for the implementation of interactive demonstrations.

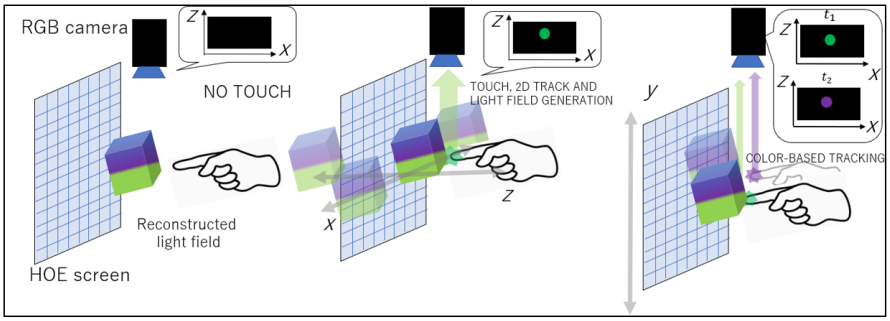
4.2 Limitations of previous approach

The implementation reported in [68], is a system that only provides a binary response (Fig. 4.1 (a)). Since it displays a still LF, functions that require updating the displayed content are not supported. Also, this setup is not fixed, and a calibration between the projector's pixels and the hogels of the HOE screen is required for the integral image to be pre-distorted and correctly displayed [46], a computationally time-consuming operation. All these operations need to be implemented in a time short enough to realize an effective UI.

Regarding the detection of scattered light, in [68] only the maximum value between 2 different color channels and a circle detection was used for discriminating the existence of the interaction. Color detection of a static LF was reported in [52] using this system, but again real-time interaction was missing. Considering that the color and shape measurements can be affected by external light, different touching angles and the use of



(a)



(b)

Figure 4.1: (a): Previous UI based on scattered light detection [68]. The "OK" signal means the interaction has been detected. (b): Proposed UI allowing the touch, tracking and LF modification corresponding to the user's movements.

diverse objects to interact, a more robust detection method is required. Similarly to the process of LF update, the interaction detection should also be realized in a short time to be used in a UI.

In [68], the camera was placed behind the screen, making use of its transparent feature. Although this sensing position allows the detection of the whole screen plane, it can be heavily affected by the light the user's body and the scene might reflect as well as the scattered light by the holographic screen, interfering with the measurement. However, since we are aiming for an UI that tracks the finger in the screen plane, changing the camera location requires the modification of the tracking method as explained in the coming sections.

4.3 GUI based on color detection of scattered light

Although the color detection has already been proposed by [68], we explain in this section how the concept was used to create a real time loop detection and how this technique was expanded to detect 3D information. In the proposed system, the camera position

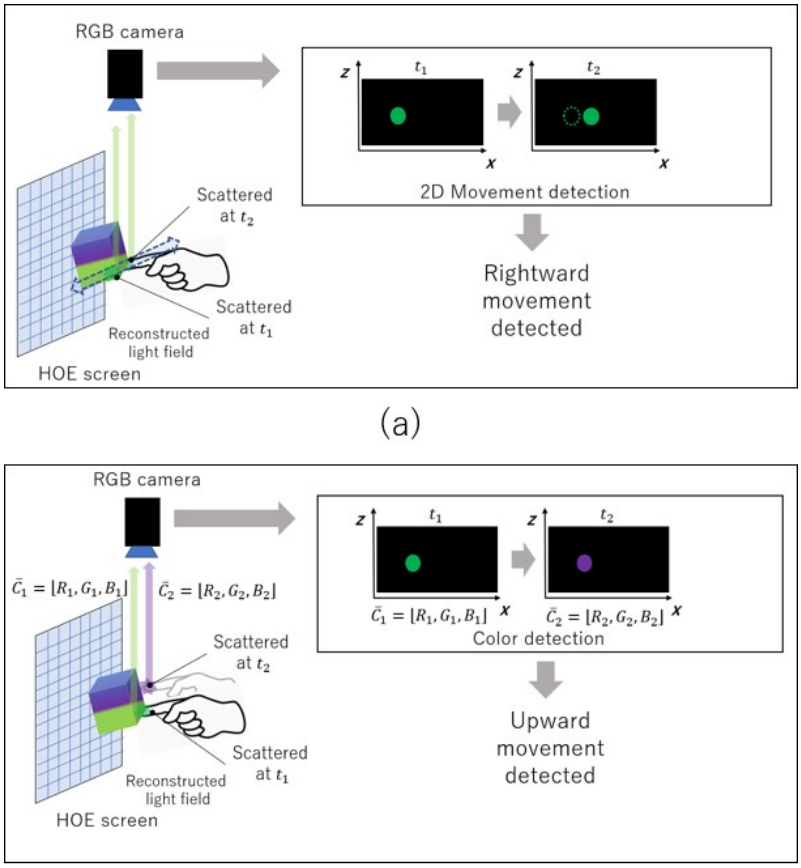


Figure 4.2: (a): **2D tracking**. Detection of scattered light to track the fingertip of the user in a 2D plane (b): **Color tracking**. Detection of different colors along the *y*-axis to add a new measurement dimension. Both (a) and (b) combined realize **3D tracking** of the fingertip

is changed from the back of the HOE screen to the top of the system to reduce the amount of undesired light, which might cause false positives. In the following discussion, we will consider the center of the HOE screen to be placed in the origin of the *xy* plane and the *z*-axis running parallel to the vector perpendicular to the screen. Therefore, the camera placed in the position indicated in Fig. 4.1(b) captures the *xz*-plane. Once a user touches the object reconstructed by the LF display, the position of the finger can be detected in the *xz*-plane from the coordinates in the captured image, realizing 2D tracking of the user's finger (see Fig. 4.2(a))

The image captured by an RGB camera is a 2D projection of the 3D scene into the camera's sensor, in which any depth information (information along the *y*-axis in this case) is lost in the capture process. To realize 3D positioning of an object while avoiding the use of external motion sensors or stereo-camera approaches, we have proposes to

use the color of the LF itself to create an extra cue, so a single RGB camera can be used as a 3D motion sensor. If different colors are used in the object reconstructed by the LF along the *y*-direction, as it is depicted in Fig. 4.2, we can compensate for the information that gets lost after capturing an image (i.e.,*y*-direction distance) by detecting the colors along that direction, so practically turning the combination of an RGB camera and a LF display into a 3D position detector.

4.4 Real time update of LF images

For the interactive 3D-touch display, the LF projected by the system needs to be updated based on the tracking of the user's finger. It will also be important to have LF updates once we implement gestures and free form drawing of LF in the coming chapters. In order to have a LF that follows the contact point of the user, real-time reconstruction of the LF was required. There exist two hurdles in this case: (1) in case the LF needs to be modified, rendering of all necessary views is required and (2) the image needs to be pre-distorted following the data of a Look-Up Table (LUT) like mentioned in chapter 3, to be correctly displayed by our system. First, we discuss the steps taken to update the views of the LF and make them usable for a GUI. In this study, we proposed two different ways to update the LF images in real time and create an interactive GUI: CPU based approach and GPU based approach.

4.4.1 CPU based approach for updating the LF views

The amount of time required to render the views of an integral photography is a widely studied problem [30]. Most research on this topic has been focused on obtaining a fast and good quality image, with a large angle of view and low interpolation, therefore using all or most of the available computer power to achieve this goal. However, in this study our main goal is to implement a GUI, and the update of the LF should remain a modest percentage of the required processing time. A solution was to use an open-source software [82] to generate different views of a scene with an angle of view equal to that of the HOE screen. The result was then multiplexed to create the IP image that was used to generate the LF. The generation of the views and their multiplexing is a time-consuming process that is not suited for real time. To take advantage of the already rendered LF, we decided to displace within the screen space the rendered LF by introducing rows of zeros of the pixel size of the HOEs (16×16 pixels when using a Full HD resolution and 120×67 mm screen size). This method is depicted in Fig. 4.3. This process skips the generation of new views and *recycles* the already sampled views.

A disadvantage is that it only allows displacement on the 2D screen plane and depth update (z-direction) is not possible.

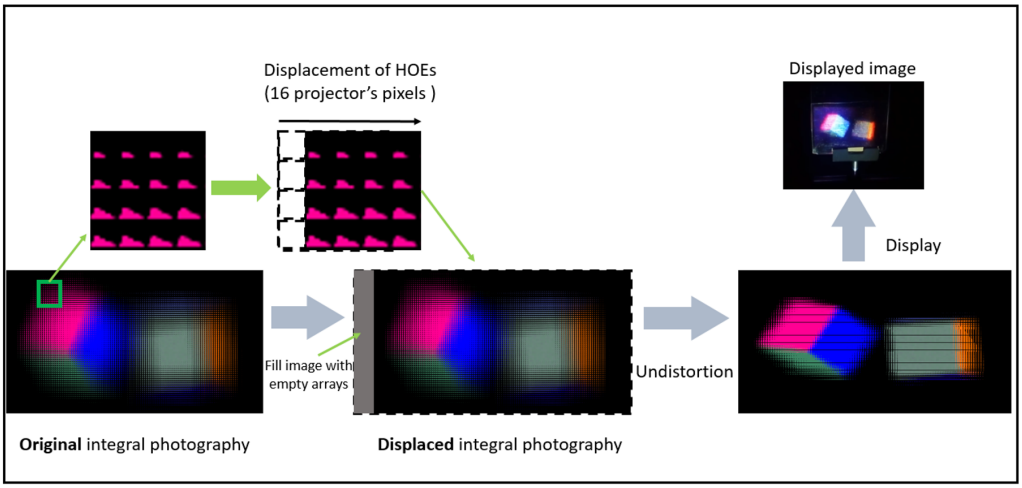


Figure 4.3: Depiction on how the LF is updated without rendering the whole image. Movement toward the right direction using a simple image displacement.

4.4.2 GPU based approach for updating the LF views

Since the CPU approach is not suitable for updating the depth of the LF or displaying a different view from the ones already depicted, we implement a ray-tracing based algorithm based on an OpenGL implementation that can be loaded in the GPU and modified at a very high frame rate. This method models the views of each HOE as a pinhole camera and computes the impact point of each light ray coming from a surface to the rendering plane (see Fig. 4.4). Although this implementation is very fast, the main drawback is that every surface should be mathematically modeled to depict the integral image. This makes the design of complicated images very tedious, not being able to use software packages like [82] to design interactive content. However, we implement it here to show the potential of the use of LF in a GUI, regarding this remaining challenge as a software engineering task that should be completed in the future.

4.4.3 Optimization of LUT-based distortion

Another issue to be addressed to achieve real time operation of this system was the optimization of the LUT assignation of the elemental images to the measured centers of the HOE screen as seen from the projector's view. Although it is a very straightforward process, this process can be very time-consuming, specially when displaying 4K resolution images. This was solved by using a high-speed Python-library [83] that enabled us to

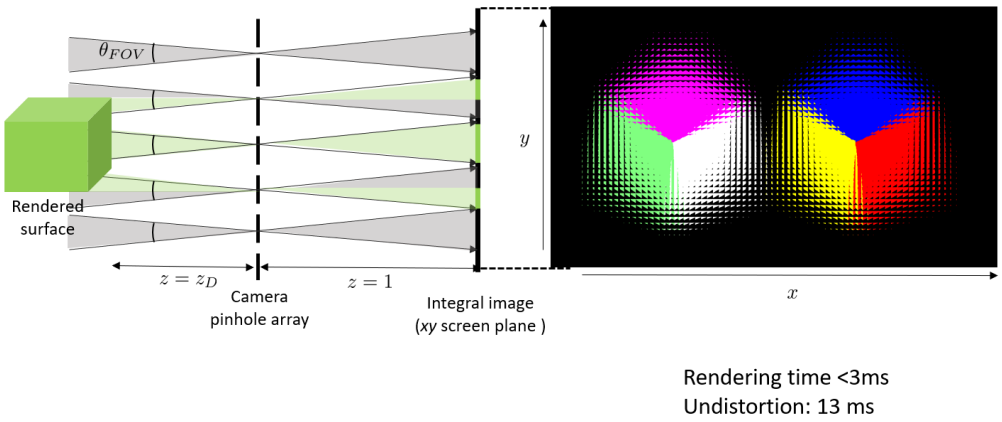


Figure 4.4: Depiction on how the LF is updated by computing each light ray hitting the modeled surface and then the result used to compute the integral image to generate the LF of the desired object.

perform this process in a real-time scale. For the real-time implementation, the finger detection step should also be optimized.

Numba is an open-source, NumPy aware optimizing compiler for Python [**?**], capable of generating machine code using the Low-Level Virtual Machine (LLVM) tool chain. Numba was created by the same author of NumPy, Travis Oliphant, in 2012 [35].

The LLVM tool chain is a set of tools, developed by the University of Illinois [84], used to write compilers for a large range of languages. The aspect of the LLVM that Numba uses is its intermediate representation (IR), which is a low-level language similar to assembly [35]. Numba uses LLVM to inspect Python code and create *type aware* versions of the functions that we wish to optimize, compiles them to the IR, and gains speed by using these optimized versions.

Since our Python implementation uses images as arrays from the very widespread NumPy Python library, which is a library prioritized for optimization when using the Numba library [35], the combination of both allows us to have a faster implementation of the correspondence between the projector's pixels and the large arrays that form the integral photography.

4.5 3D tracking based on scattered light detection

In this chapter, the basic idea is to achieve the 3D tracking of a user using the features of the scattered light. The presented method to detect that light and obtain a 2D position is straightforward, since the camera sensor naturally projects a 3D position into

a 2D plane. However, to complete a 3D tracking system, extra spatial information is required. In this section, we explain the use of color variation of the reconstructed LF along the missing dimension to achieve a complete 3D tracking method based on scattered light.

4.5.1 Tracking of interaction using 2D movement detection

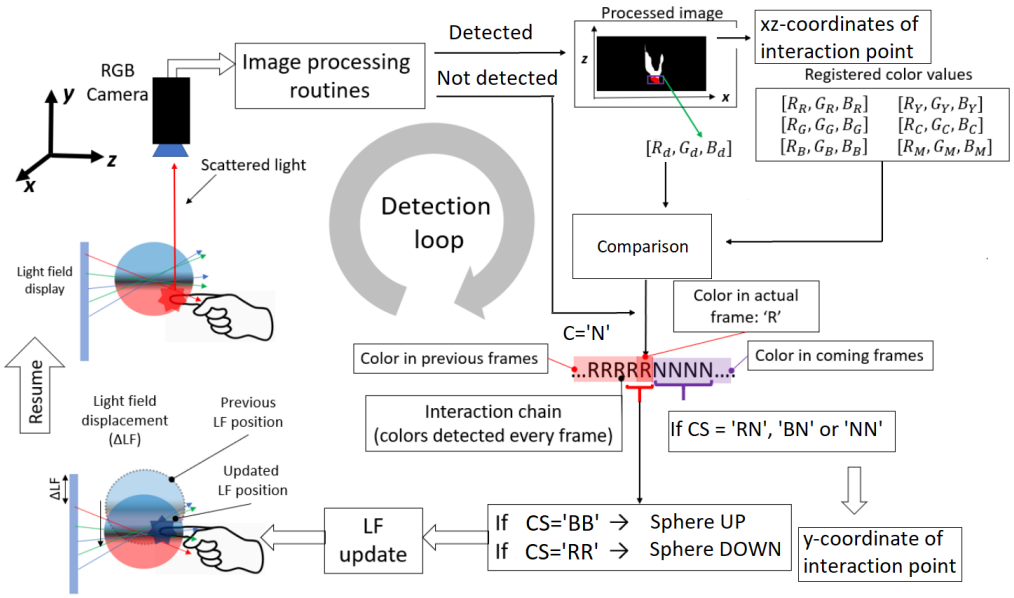


Figure 4.5: Details on how the color identification is used for detecting the movement of the user.

A figure summarizing the whole detection process is shown in Fig. 4.5. First, the touch-event detection is performed. As a first step for the touch detection, background subtraction is performed from the camera's captured image. Background subtraction allows the system to recognize the parts that are moving within the scene, obtain their features and analyze the scattered LF coming from that identified region. The LF scattered by the user is captured by segmenting the ROI from the camera image by the following process. Assuming that the user begins the interaction by pointing at the LF with the finger, we obtain the finger region by applying a background subtraction algorithm. This background subtraction consists of averaging 100 frames of the background captured by the camera and subtracting it from the captured image at every frame. This image is binarized to acquire the finger and possibly some other hand parts. Next, the contours of the main structures are extracted from the binary image acquired. Since the tip of the finger usually corresponds to the pixel closest to the HOE screen in the z-direction (see Fig. 3.3 for axes definition), the minimum z pixel of the obtained contours is the

one we consider to be the fingertip of the user. This ROI extraction method is similar to the one performed in a previous work [53]. Another, more sophisticated method to segment the interaction position and obtain the interaction point, will be described in section 4.7.

The segmentation process will be repeated for a continuous stream of frames to look for the interaction of the user within the region in front of the LF display. When the light is scattered and detected after subtraction, contrast stretching is applied to the image to better locate the interaction point. Next, a binary mask is made from the subtracted image, and multiplied on the captured image to identify the touch-detected region.

As the next step, image processing modules such as small clusters removal and erosion and dilation filters are applied to have a good quality mask that is robust enough to noise. To increase the robustness of the process, the contours of the main structures in the binary mask are extracted. This extraction detects the chain with the largest number of pixels (contours are one pixel width), which usually corresponds to the main interaction point. In this way, the secondary noisy contours shown in the binary mask of Fig. 4.6 can be discarded.

Next, the position of touch-detected area in xz-coordinates is identified. Since the camera does not move with respect to the LF display, a fixed correspondence relating the reference frames of the interaction-detection RGB camera and the display is previously established. This allows the system to know the xz-location inside the HLF display in which the interaction is taking place by using the camera coordinates.

4.5.2 Tracking of interaction using color change detection

The interaction-detection algorithm described in the previous section can provide a good position of the position of the user's finger in the *xz*-plane. However, this method alone is not capable of detecting the movement of the finger in the remaining *y*-axis. To provide a complete interface that is capable to detect the direction of the user in the *y*-axis, we propose a color measurement and classification algorithm that can provide extra information to the system about the user's position.

To detect the movement in the *y*-axis, we used a sphere with different colors in the upper and lower hemispheres (bicolor sphere), separated by a dark region to provide a color transition (see Fig. 4.5). In fig. 4.5, let us denote the detected color in a single loop as C, and the sequence of C in two contiguous frames is denoted as CS.

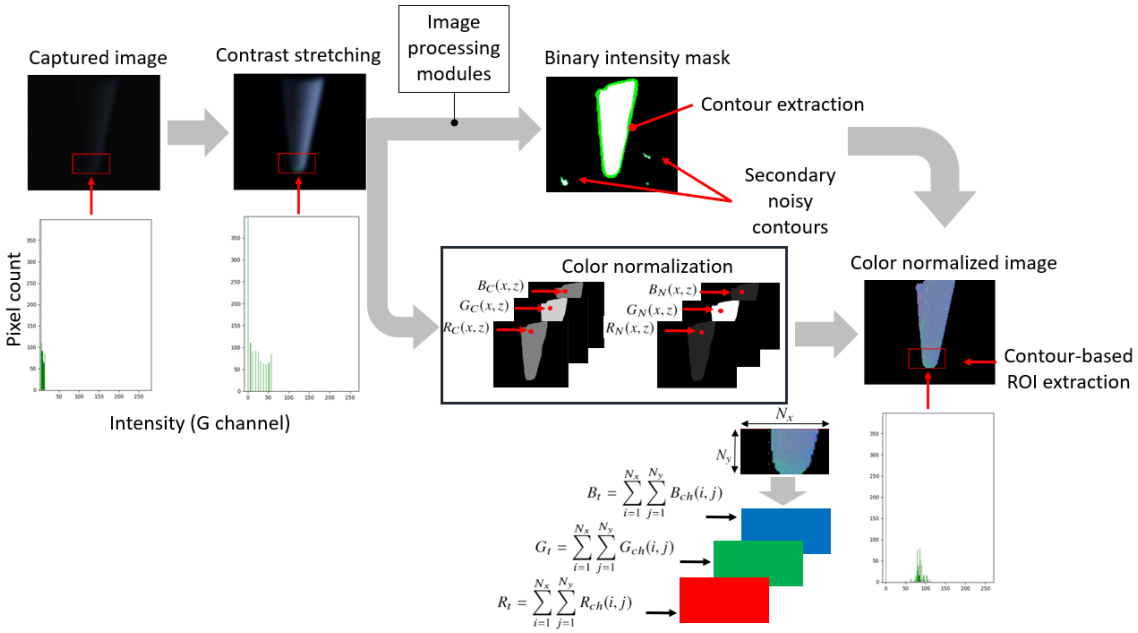


Figure 4.6: Image processing pipeline to extract the color interaction.

If the user scatters the light of the lower hemisphere (red light), the system detects the red color ($C = 'R'$), which indicates that the position of the finger is located below the center of the bicolor sphere. This triggers a displacement of the bicolor sphere to the downward direction (negative *y*-axis). In fact, the decision is done when $CS = 'RR'$ for robustness, as shown at the bottom of fig. 4.5. This displacement step was set to values between 5 and 8mm (approximate finger vertical width). After repeated color measurements and displacement, once the dark center of the bicolor sphere has been displaced to the interaction point (i.e., user's fingertip), the mixture of red and blue lights and dark region will cause no color detection (denoted as $C = 'N'$ in Fig. 4.5). In this case, the system will stop triggering the movement of the LF. This is the cue that indicates the position in the *y*-axis of the user's fingertip, which corresponds to the distance the LF was displaced. Conversely, if the user scatters the light of the upper hemisphere (blue light, i.e., $CS = 'BB'$), the opposite sphere movement takes place until the blue identification fails, and the sphere ceases its movement. In this way, the system uses LF to scan the region of the color ball for the correct location of the interaction point in the *y*-axis. Naturally, if the user moves the interaction point in this axis, the system will detect the displacement when either red or blue scattered light is detected in contiguous frames. Then the ball can track the fingertip movement.

4.5.3 Color identification model

As shown in Fig. 4.5, the color detection process is repeated for each captured frame. This process allows the detection of movement of the user in the *y*-axis, while the detection in the *xz*-plane is achieved by the extraction of the interaction point that corresponds to the pixel located closest to the screen in Fig. 4.6, which is already a 2D value. Although with this method we are not able to get the *y* coordinate with the precision achieved in the *xz*-plane, the change of direction is a cue useful enough for the implementation of some basic gestures and to enhance the interface capabilities.

To extract the color value of the masked region, which is the interaction point in an image, we have to consider some difficulties; for example, the scattered light, can be mixed with other sources of light coming from the environment surrounding our system (office illumination, reflections, etc.). Here we present a first approach to the enhancement of color signal. However, an improved approach will be described in section 4.7. As a first approach, we apply a normalization of the $[R, G, B]$ values of each pixel in the interaction frame. Let the captured image be a 3-dimensional array, each channel having a size of $N_{\mathrm{x}\times\mathrm{N_z}}$ pixels. In the pixel at (x, z), the captured RGB value corresponds to $[R_C(x, z), G_C(x, z), B_C(x, z)]$ (one value for each color channel). In our system, each pixel value is normalized as follows:

$$R_{\mathrm{N}} = \frac{R_{\mathrm{C}}}{R_{\mathrm{C}} + G_{\mathrm{C}} + B_{\mathrm{C}}}, \quad G_{\mathrm{N}} = \frac{G_{\mathrm{C}}}{R_{\mathrm{C}} + G_{\mathrm{C}} + B_{\mathrm{C}}}, \quad B_{\mathrm{N}} = \frac{B_{\mathrm{C}}}{R_{\mathrm{C}} + G_{\mathrm{C}} + B_{\mathrm{C}}}. \tag{4.1}$$

This will provide a stronger color signal over the 3 channels, as shown in Fig. 4.6 (normalization). It can be seen that we have obtained an image with a region of enhanced color at the tip of the touching object (a stylus in case of Fig. 4.6).

Here we are mainly interested in the region with the greatest color change. The interaction point that is more prone to have a color variation related to the generated LF is the one closest to the screen, or the minimum *z*-coordinate, because the fingertip is touching the LF. To avoid for a noisy value to be taken as this *z*-coordinate, the corresponding minimum coordinate of the largest detected contour is used.

For color classification, the color content of a square box around the touch detected position in an image is measured. To extract a color vector from this box, we sum up the pixel values in the box, and then normalize it as follows:

$$\vec{C}_{\mathrm{d}} = \left[\frac{R_{\mathrm{t}}}{R_{\mathrm{t}} + G_{\mathrm{t}} + B_{\mathrm{t}}}, \frac{G_{\mathrm{t}}}{R_{\mathrm{t}} + G_{\mathrm{t}} + B_{\mathrm{t}}}, \frac{B_{\mathrm{t}}}{R_{\mathrm{t}} + G_{\mathrm{t}} + B_{\mathrm{t}}} \right]. \tag{4.2}$$

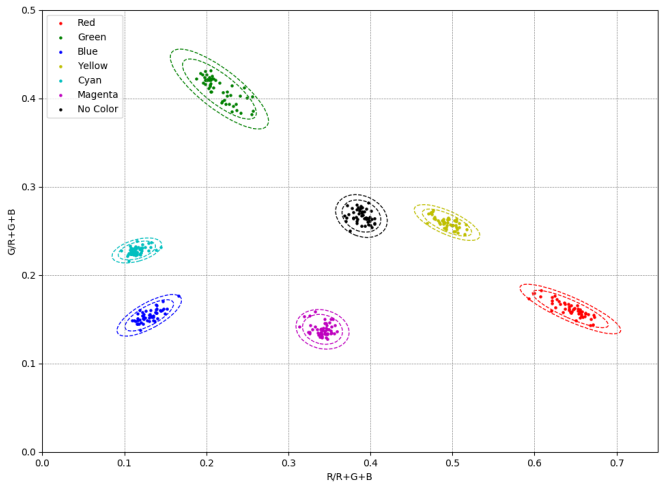


Figure 4.7: Distribution of the sampled color frames in the *RG* normalized color space with confidence ellipses of 95% (inner ellipse) and 99 % (outer ellipse) around each color distribution.

In Eq. (4.2), the subscript t denotes the sum of the respective color channels in the box that defines the *touched* area. After averaging several samples of the vector of the *detected* color $\vec{C}_{\mathrm{d}}$, the averaged value is used to detect each captured color. A 2D projection of the detected color from the normalized *RG* plane is shown in Fig. 4.7. The results of measuring 6 colors with this method as solid LF balls reconstructed by the system (red, green, blue, yellow, cyan, magenta) and scattered by a finger are depicted in Fig. 4.7 along with a "No-Color" (or NC) case that corresponds to the color of the interaction target when is not scattering the LF. The distribution shown in Fig. 4.7 was acquired for 20 touching events for each color. Within the system implementation, this retrieval of color information and saving can be implemented as a photometric calibration prior to the initialization of the UI.

The finger-movement detection process outlined previously gave as a result the color category classes shown in Fig. 4.7. Several measurements of the vector $\vec{C}_{\mathrm{d}}$ in Eq. (4.2) provide us with mean value vectors $\vec{\mu}_{\mathrm{N}}$ ($N \in [R, G, B, Y, C, M, NC]$) and standard deviations σ_{N} of those color clusters. This information is saved into the system as a color database and used for later identification. The use of the normalized RGB space in Eq. (4.2) allows us to have a color measurement unaffected by unevenness in intensity of the captured values. Since we are using this space, it will suffice to use only the R and G components of it without any information loss (since $B = 1 - R - G$). The question that arises now is what is the most adequate way to match a captured color value $\vec{C}_{cap} =$

$(C_{\mathrm{R}}, C_{\mathrm{G}})$ with the mean value of each of the clusters in the database $\vec{\mu}_{\mathrm{N}} = (\mu_{\mathrm{NR}}, \mu_{\mathrm{NG}})$ to implement a movement cue along the *y*-axis. The most straightforward approach is to map a captured color value into the color space of Fig. 4.7 and measure the Euclidean distance from the mean database values to the captured value. The Euclidean distance expression d_{E} is given by:

$$d_{\mathrm{E}}(\vec{C}_{\mathrm{cap}}, \vec{\mu}_{\mathrm{N}}) = \sqrt{(C_{\mathrm{R}} - \mu_{\mathrm{NR}})^2 + (C_{\mathrm{G}} - \mu_{\mathrm{NG}})^2}. \tag{4.3}$$

However, as it is easily seen from the color distribution in Fig. 4.7, the color values are not evenly spread on the different axes of the *RG*-space. To take into account the correlations between the variables [44] and to improve the accuracy of this method, we also measure the Mahalanobis distance that $\vec{C}_{\mathrm{cap}}$ has with respect to each of the N clusters with mean value $\vec{\mu}_{\mathrm{N}}$ [52]. In that case, the expression for the Mahalanobis distance d_{M} is given by:

$$d_{\mathrm{M}}(\vec{C}_{\mathrm{cap}}, \vec{\mu}_{\mathrm{N}}) = \sqrt{(\vec{C}_{\mathrm{cap}} - \vec{\mu}_{\mathrm{N}})^T \Sigma_{\mathrm{N}}^{-1} (\vec{C}_{\mathrm{cap}} - \vec{\mu}_{\mathrm{N}})}. \tag{4.4}$$

In Eq. (4.4), Σ_{N}^{-1} is a 2×2 matrix that denotes the inverse covariance matrix of the data associated to the color cluster N in the RG plane of Fig. 4.7. Using the eigenvectors and eigenvalues of the covariance matrix, we are able to plot confidence ellipses around each distribution. A captured color vector $\vec{C}_{\mathrm{cap}}$ has to be located inside one of the ellipses defined by a color distribution to be classified as a value corresponding to that distribution. Since we consider the values captured are normally distributed, they follow a chi-squared (χ^2) distribution with 2 degrees of freedom [44]. Therefore, to plot the confidence ellipses, as well as the detection thresholds for the detected values of the vector $\vec{C}_{\mathrm{d}}$, the major and minor axes are scaled according to the values of the mentioned distribution for different confidence ranges (see Fig. 4.7).

In order to evaluate the accuracy of the color classification shown in Fig. 4.7, we conduct an experiment in which the LF of six spheres of solid color were reconstructed and displayed one by one in the screen. The user then scattered the light of the LF while the system performed the color identification with both Euclidean and Mahalanobis distances. The scattered light was captured by the RGB camera and then, following the algorithm described in Fig. 4.6 and the details in this section, the scattered light was converted to the vector $\vec{C}_{\mathrm{d}}$ in Eq. 4.2. The results of the color detection for each of the tested colors are shown in table 4.1.

Distance	Red	Green	Blue	Yellow	Cyan	Magenta	No color
Euclidean	66%	30%	91%	39%	40%	94%	79%
Mahalanobis	64%	71%	99%	79%	56%	98%	99%

Table 4.1: Comparison of the detection effectiveness for each of the distance metrics used. The identification effectiveness was tested on a sample of 250 frames per color. Except for red, Mahalanobis distance yields in general better results than identification using Euclidean distance.

To calculate it, the number of correct classifications was considered, counting both misclassified frames and empty frames as failed detections. In the case of Euclidean distance, the detection threshold was set to be a sphere of a radius $r = 2\sigma_c$, with $c \in [R, G, B, Y, C, M, NC]$. For the Mahalanobis distance decision threshold, we used the ellipse that corresponds to the interval with 99% of confidence as illustrated in Fig. 4.7. By comparing both metrics, it is concluded that the Mahalanobis distance is a better fit for the shape that the color distribution acquires in the RG normalized space.

4.6 Interaction examples

The location of the interaction point is readily extracted from the contour extraction described in Fig. 4.6. During the detection of scattered light, the displacement of the finger is measured and once the finger has moved a distance roughly equivalent to the size of the LF sphere in the camera coordinates, the LF displacement is triggered and the finger is re-illuminated by the displaced LF. An application of this approach is shown by realizing a *drag* gesture akin to the one usually used to unlock a mobile device. Since we are detecting a 2D displacement for the *drag* gesture, the color identification at this stage is only used to confirm the user is actually touching the green LF and not other external light.

A frame sequence showing this application is shown in Fig. 4.8. The system is able to follow the position of the fingertip along the horizontal direction to drag the LF of a sphere and *unlock* the screen. The application of the color recognition also allows us to place another informative LF ('unlock' sign) on top of the lower green sphere used as the interaction light. Even if the user scatters the blue light of the upper sign while interacting, this will not be recognized as an input due to the color identification.

By implementing the pipeline detailed in Fig. 4.5 we realized a demonstration in which

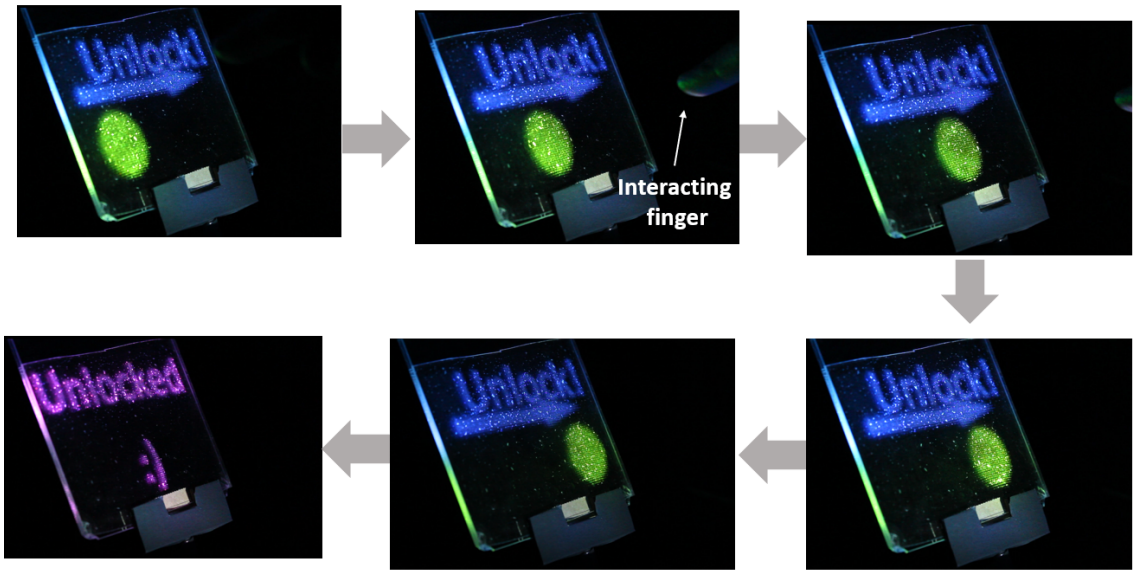


Figure 4.8: Realization of the drag gesture using the light scattered from the LF.

the user is capable of moving the LF of a bicolor sphere in the up and down directions. The implementation of the described method provided an interface that allowed the user to move the ball up and down at will (see Fig. 4.9).

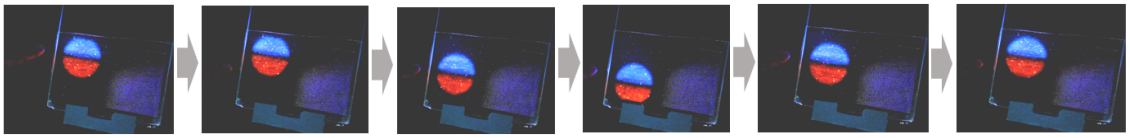


Figure 4.9: Demonstration of user's interaction with the LF along the *y*-axis using color cues. The position of the LF follows the finger that scatters the different reproduced colors.

Another demonstration of how the color can be used to locate different positions along the *y*-axis consisted on a music volume control, with the detection of each color indicating an increase or decrease of sound (see Fig. 4.10). Using colors also permits to have more buttons with different functions in a reduced screen area, by just placing several colors at different heights.

Combining the location in the *xz*-plane, along with the direction in the *y*-axis, we realized a scroll-ball capable of moving a reconstructed cuboid. A detailed scheme of its functions is depicted in Fig. 4.11 and the algorithm shown in Fig. 4.12. Fig. 4.13 depicts how this scroll-ball was able to move the LF in all the indicated directions. As the interactions are only implemented with the bicolor sphere, nothing happens when a finger touches an area different to the bicolor sphere. This is because the cuboid object and the

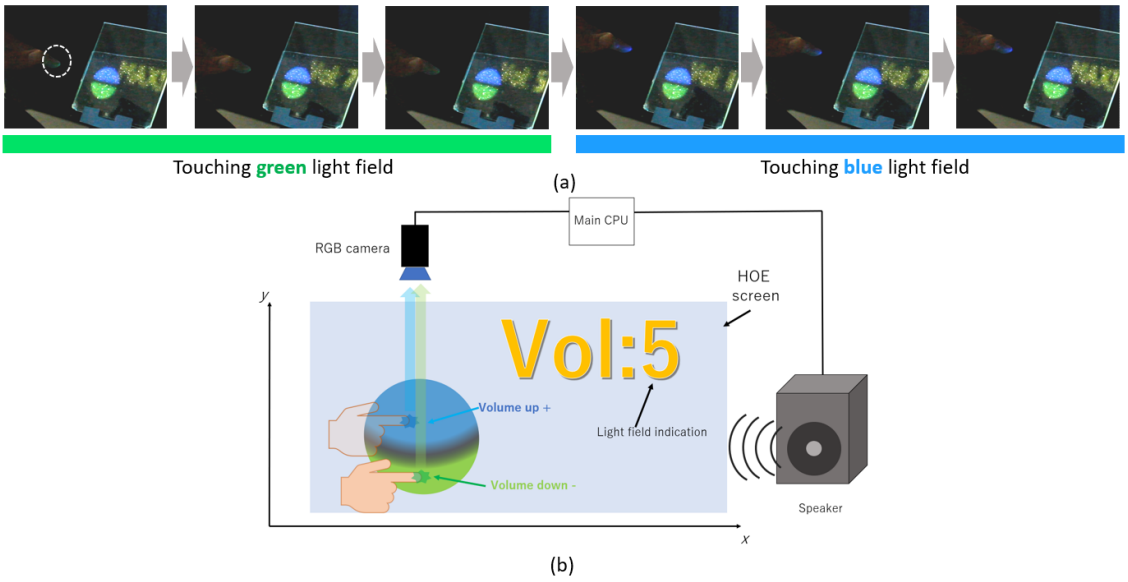


Figure 4.10: Demo for controlling the volume of an audio system. (a) When the volume is placed at 'MAX', the user can lower it by scattering the green LF (see leftmost picture, fingertip's color). Correspondingly, the user can increase the volume once more by scattering the blue LF. This interaction only uses the color values of Fig. 4.7 using the Mahalanobis distance to realize the required task. (b) Front view of the reproduced LF, with the volume indicator placed behind the bicolor sphere.

background content have different colors (orange, magenta and dark blue) from the scroll-ball, therefore minimizing the interference and the errors. Increasing the degrees of freedom in the usage of color is one of the issues to be addressed in future studies.

These demonstrations prove that the concept of reading out the color of the regenerated LF is a very feasible way to create an interface that is not only robust, but also contact-free and requiring nothing more than the use of a single RGB camera. Besides, since it is a technique that uses the LF directly, the problem of calibrating the content with the position of the user gets naturally solved.

4.7 Scattered light detection under arbitrary illumination

The previous demonstrations based on LF detection have been done with exposure times, gain, and lens aperture values that permitted to obtain the binary mask and interaction point only when the user used the finger to scatter the LF. Therefore, without LF scattering, the user would not be detected by our system. This is also related to the

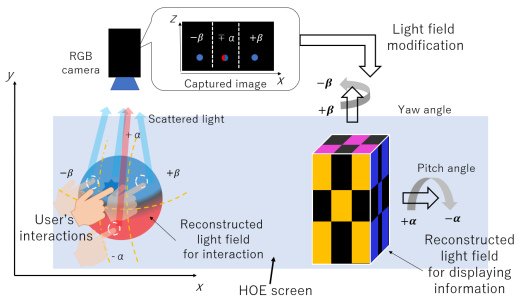


Figure 4.11: Scroll-ball used to spin the reproduced LF.

if $\vec{C}_d == RED$ **then**
  $\alpha - - = 10°$
end
else if $\vec{C}_d == BLUE$ **then**
  if $x < th_{x1}$ **then**
    $\beta - - = 10°$
  end
  else if $th_{x1} < x < th_{x2}$ **then**
    $\alpha + + = 10°$
  end
  else if $x > th_{x2}$ **then**
    $\beta + + = 10°$
  end
end

Figure 4.12: Scroll-ball algorithm

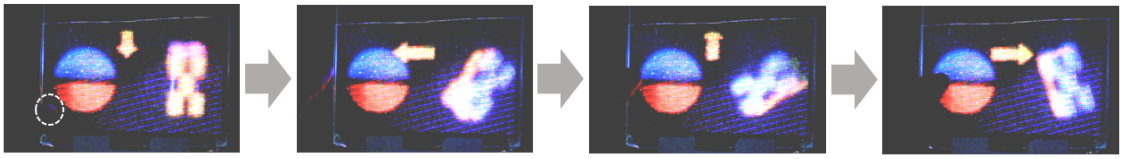


Figure 4.13: Demonstration of how the scroll-ball of Fig. 4.11 is used to interact with a LF reproduction. Arrows in the middle of the screen indicate to the user the direction of the movement. The dotted circle in the leftmost picture shows the finger scattering the LF.

light conditions of the room the system is placed on. If we have a fully illuminated room, then the finger will scatter not only the LF but also the environmental light. Depending on the color of such light, its intensity might significantly affect the color detection. If the intensity of the room illumination is greater than that of the scattered light, then no matter how low the exposure time, gain, and how closed the RGB camera's aperture be, the finger is detected under bright illumination and the system response would be indistinguishable from the LF scattering. Since this system should be used under common illumination conditions and not in a dark room, a more sophisticated method is adapted to detect the scattered light.

4.7.1 Mixture of Gaussians (MOG)-based background subtraction

As previously mentioned, the LF scattered by the user is captured by segmenting the ROI from the camera image by applying a background subtraction algorithm that consists on subtracting a previously captured average of the scene before starting the interaction. The resulting image is binarized to acquire the finger and possibly some other hand parts. Next, the contours of the main structures are extracted from the binary image acquired. Since the tip of the finger usually corresponds to the pixel closest to the HOE screen in the z-direction, the minimum z pixel of the obtained contours is the one we consider to be the fingertip of the user. All these changes of the illumination environment can result in an incorrect segmentation.Therefore, we use a background subtraction based on Mixture of Gaussians (MOG).[75]

However, it should be considered that there might be various changes of illumination that take place during the day, a lamp turned on or off (see Fig. 4.14), or when the user leans towards the system, blocking some reflected light from the environment and modifying the background model previously established. This algorithm models each pixel inside the captured image as a sum of Gaussian distributions based on the values of that same pixel in previous frames over a reasonable time. The model is constantly updated to account for any changes in the illumination of the background, obtaining a more robust segmentation. The generated binary mask is processed using erosion and dilation filters, cleaning the image and obtaining an ROI of the interacting finger, just as it is done with the average-based background subtraction.

The effect of the use of MOG-based background subtraction and a straightforward background subtraction can be seen in Fig. 4.15. In Fig.4.15(a) we use the above described background subtraction where an average frame of 100 frames captured before the beginning of the interaction is subtracted from each captured frame. An LED lamp is lit in both cases, causing false positives in the segmentation mask. On the other hand, using a variance threshold for the pixel model match of 16, 2000 previous frames to train the background model and an automatic learning rate, the mask correctly segments the finger after sustaining the same illumination change as shown in Fig. 4.15(b). The illumination is a fluorescent mercury lamp, the camera gain is set at 17 dB and the exposure time set at 7000 μs.

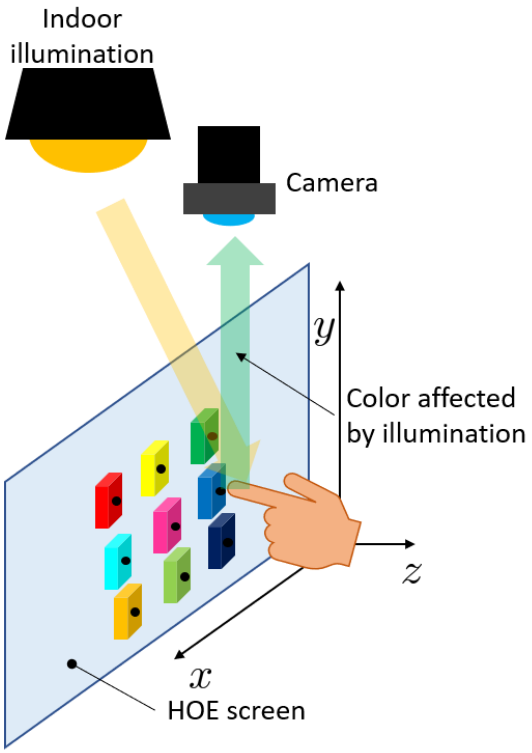


Figure 4.14: Indoor illumination can affect the color detection

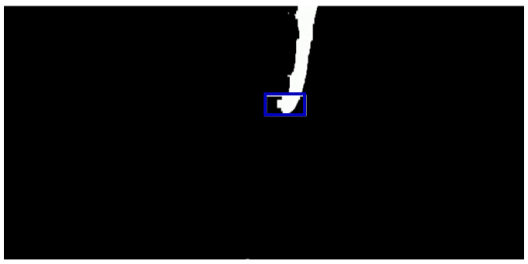
4.7.2 Pre-sampling of the user's finger

Once a correct finger region has been established using the segmentation procedure, the color scattered by the finger has to be accurately detected. One more problem with capturing the light scattered by the fingertip is that the intensity of the LF can be low. This is a problem for detecting the scattered light and the corresponding interaction since a low-intensity signal might get mixed with the environmental light in the space of the interface, causing an incorrect detection or no detection at all.

As mentioned above, the previous method could not detect the finger if it did not scatter LF because of the low light conditions. This time, the problem is to detect if the finger is actually scattering LF or not, but the finger itself can be detected even without the presence of LF. By demonstrating that scattered LF can be captured under bright conditions, we show this method is also applicable in such setting.

To account for the presence of other sources of light, the system will request the user to show the finger under the environmental light without any LF present after obtaining the MOG-based segmentation presented in Fig. 4.15. The user is requested to show the finger for a short time and an average of the captured frames of the finger under environmental light is obtained before the interaction begins.This is what we call *pre-*

(a)



(b)

Figure 4.15: Comparison of finger segmentation mask using conventional background subtraction (a) and MOG-based background subtraction (b) when impinging an LED lamp to the finger. Notice how the ROI shifts to an incorrect area in (a-blue square) but it correctly segments the fingertip in (b).

sampling of the user's finger, which provides the base color the LF scattered colors will be mixed with.

The segmentation of the image of the finger and the computation of its average is realized at a sampling rate of approximately 70 FPS. This process can be completed in approximately 1.43 seconds when sampling 100 frames, requiring the user to only pass his or her finger through the empty screen before starting the interaction for a very short time. Even if the finger is constantly moving, the average can be computed easily because the pixel closest to the screen is used to generate a fixed region around the finger. This region of 50 $\times$ 25 pixels is segmented out from the original captured image of 640 $\times$ 320 pixels. The averaging process is performed only in the region generated around the finger, becoming very fast. Once the system obtains the required 100 frames, the display of LF starts automatically, without any further action required from the user.

After obtaining the average of the finger without any scattered LF, this average image is subsequently subtracted from the captured frames when interaction with the LF starts. This straightforward approach allowed us to increase the distance between the average value of the scattered colors as shown in Fig. 4.16. The graph in the left bottom

part indicates how the means of the colors are clustered in the center of the graph, while its values get more distinguishable from each other after subtracting the averaged finger under indoor illumination (right bottom graph). When compared to the previously obtained distribution in Fig. 4.7, we observe that the same distribution is reproducible under an office-illumination environment when we apply these changes, recovering color information that would be lost otherwise.

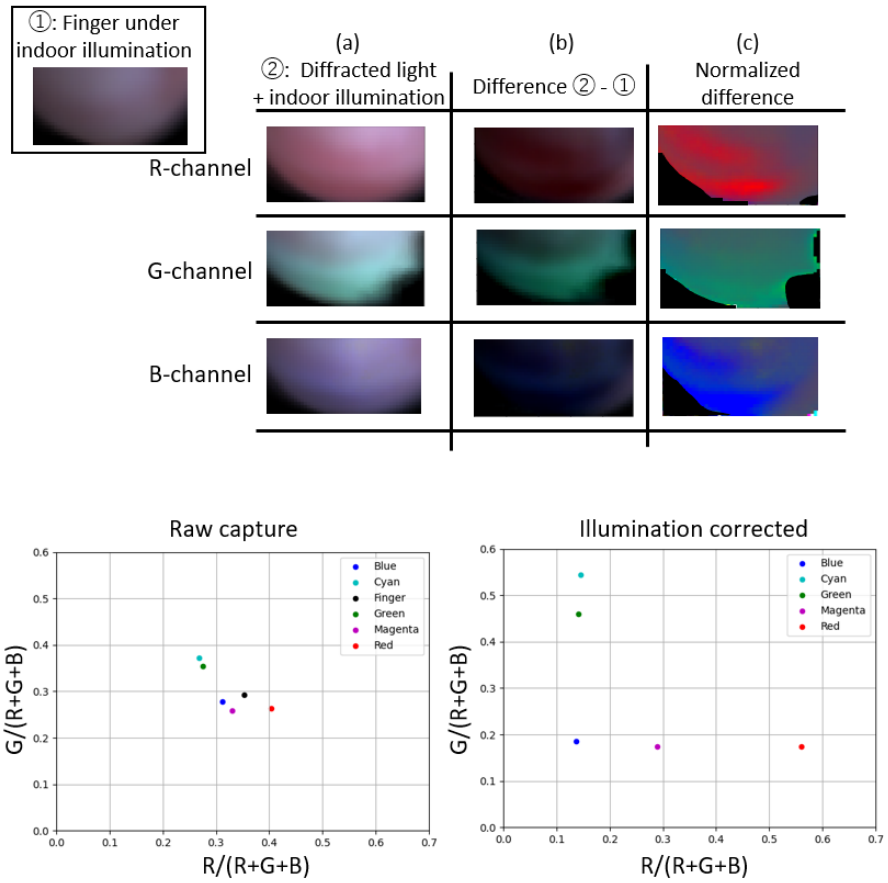


Figure 4.16: Effect of subtracting a finger image without diffracted light (i.e., scattered LF) and normalizing the color result. The separation of the obtained pixel values can be seen in the lower graphs.

This implemented changes show how the color information can be retrieved for its use even in environments where it might get obstructed by other light sources. The here proposed method is based on a capture and subtraction of the finger of the user, but that image can also change depending on the present illumination. A possible solution could be to implement something similar to the MOG subtraction that updates the model depending on the previous captured frames. This would also allow for the capture of scattered light in the hands of people with different reflectance in their skins (e.g., dark skinned people or people with polished fingernails, etc.). The further development to

improve this measurement should be the aim of future studies.

A demonstration that proves the correct color identification under environmental light consists of a button interface that was adapted to represent an ATM (see Fig. 4.17). Depending on the function the user requires, the finger is placed on the required button and that function is executed using the color identification. The system is programmed to execute a certain function depending on the scattered color, similar to the volume interface demonstrated in Fig. 4.10.

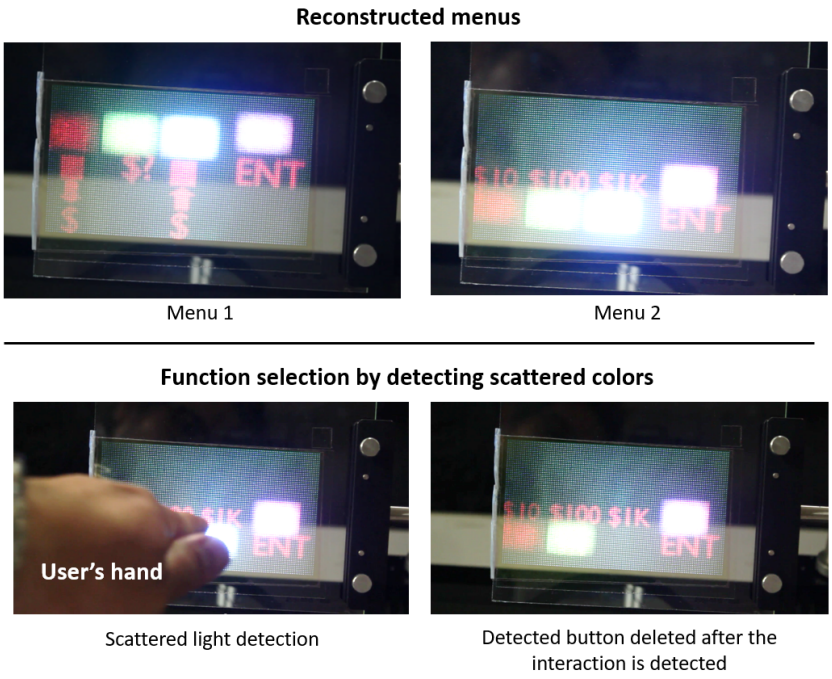


Figure 4.17: Example of the implementation of color detection to reconstruct an interface that resembles an ATM: The user points to a button that corresponds to a function, the color is scattered and the corresponding button vanishes.

The presented applications demonstrate the feasibility of the use of color information in the LF reproduced by a holographic LF display to attain an interactive interface that naturally solves the problem of aligning the displayed content to the signal used to detect the interaction. The use of color permits the creation of floating buttons with several functions and, as it has been shown, very complex tasks can potentially be implemented, such as writing and drawing in a 3D space without any restriction but the area of the screen. This method is an intuitive one, that works directly with the generated content of the screen and that can be easily popularized and promoted as an easy-to-use and contact free interface. Its applications could range from interactive digital signage for cars, kiosks terminals for department stores and scenarios in which hygiene plays an

important role such as hospitals, food-processing facilities, etc. This study is also one step forward into making glasses-free 3D interfaces more widespread, considering that the lack of an adequate application has hindered them from becoming a more popular technology. Finally, although not very exploded in this study, the holographic LF display based on HOEs is a transparent screen that can be blended with the environment and give rise to even more applications (e.g., information display of a moving scenario).

4.8 Future work

The use of the color information as an extra cue when using the detection of scattered light is an easy to implement and useful approach that has many advantages like not requiring infrared detection and keeping the registration of the content and the user. It might be argued that the use of color as a detection cue, specially in the y-axis direction, limits the color of the reconstructed LF because the color should be changed along this direction. That is indeed true, but it can be solved by applying a time-multiplexing scheme in which the color is displayed for a period of time and then the intended LF is displayed, at a frame rate high enough for the user not realizing that color detection is being used.

An approach like this would require a trigger that synchronizes a camera and the projection of the image. Research involving high speed projectors of 947 fps synchronized with cameras at 500 fps has been reported [22]. An approach involving the use of a high speed projector and a high speed camera along with the LF projection to realize 3D tracking based on scattered light can be the focus of a future study. In Fig. 4.18, we show a proposal on how to combine this approach with high-speed devices.

One more point is how can the scattered light be considered to be a *haptic*-like element when interacting with a 3D reconstructed object. The continuous light in the tip of the user as a "haptic" element is, in our personal opinion, very small. In addition, the reflected light is sometimes obstructed by the user and, most of the times, not visible to the user because the finger itself obstructs it, as mentioned by the users in the user evaluation. So it is very common that the user is not even able to realize that light is being scattered by the finger while interacting. Nevertheless, in the case of the ATM interface, we personally felt a stronger feeling of interaction when the LF disappeared after pressing one of the buttons. The sensation of light disappearing from the finger after irradiating the finger gave the sensation of a responsive system. Therefore, in future work it would be interesting to propose how using a blinking light instead of a

continuous stream of light in the finger of the user might increase the user satisfaction with the GUI.

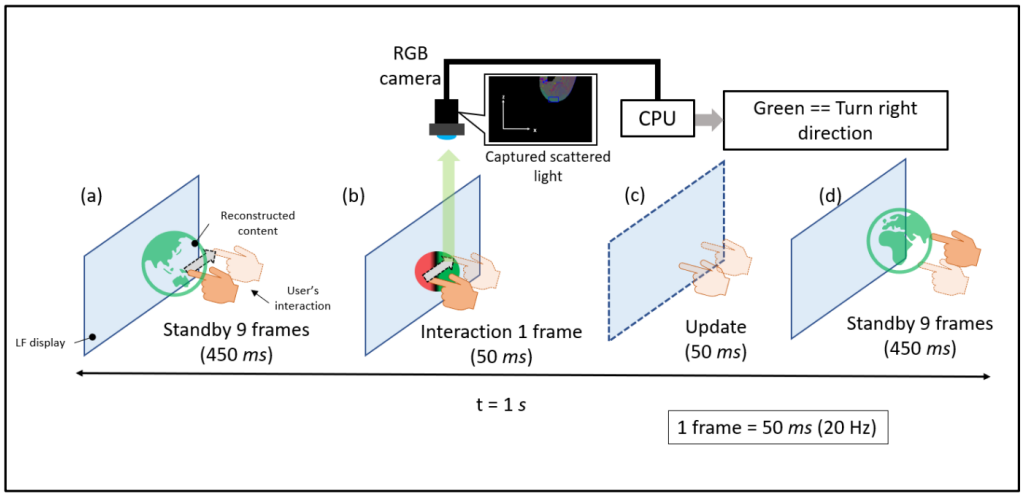


Figure 4.18: Proposal on how to merge color detection with the interaction of a LF image by synchronizing the projector and the camera. Even higher frame rates are potentially possible with the currently reported refresh rates [22].

Finally, to have a more real interaction with the captured objects, a larger display is also desirable. However, since the size of the HOE-based LF display is limited by the size of the collimating lens, creating such a system will result in a very bulky device. Recent proposals [25] on digitally designed HOEs that record in the pattern the required optics using a spatial light modulator (SLM) are also an interesting alternative to further improve the quality of our system and enlarging the device without requiring heavy collimation optics (see Fig. 4.19). Implementation of optical functionalities in a digitally designed HOE could in the future include a correction to the optics of the used projector to improve the quality of the displayed image.

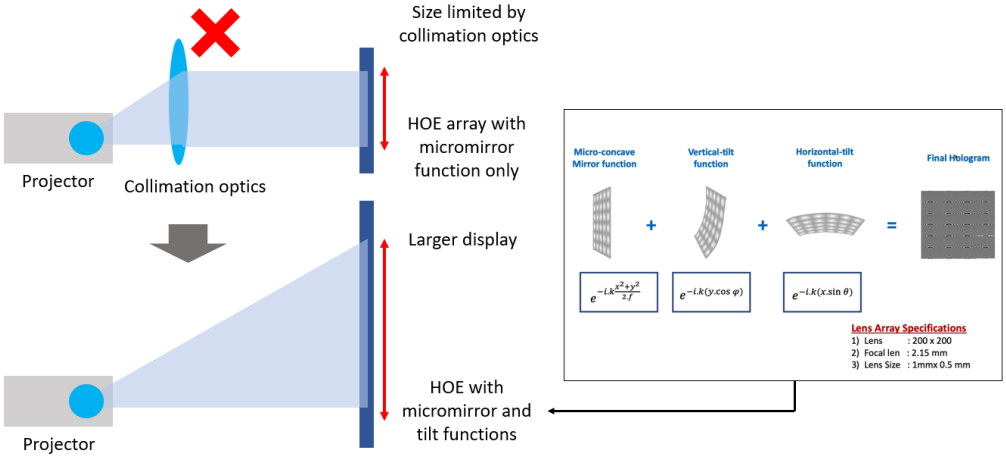


Figure 4.19: Proposal to create a larger display for an interface that better accommodates human interaction

Chapter 5

Fusion of scattered light detection and 3D tracking sensor

The proposed approach in chapter 4 of using the scattered light when touching a LF reconstruction and identifying the scattered light along with its color is an original proposal of our group, naturally registering the user position and the LF, and robust enough under the adequate experimental conditions. It can potentially be blended with other proposed approaches in section 4.8 to potentially realize more complicated tracking routines. Nevertheless, there are already many widely available approaches that track the gestures and position of the hand and fingers of a user without requiring any hardware attached to the user.

5.1 Rationale behind the fusion of scattered light detection and 3D tracking sensors

In recent years, there has been a strong development of several off-the-shelf devices that can track the gestures and the position of one or both hands of a user. Some examples are the Microsoft Kinect [50], Intel RealSense [45], Leap Motion(LM) Controller [66], among others. The described devices have been used to implement gesture interfaces and several other applications, such as the control of a robot arm reported in [66]. They can be used to implement both *gesture* interfaces and *direct-touch* interfaces. A *gesture* interface is an interface where the user mainly inputs information into the system through the use of gestures. These gestures are a movement pattern identified by the corresponding vision system (stereo camera, structured light, etc.) and they may or may

not spatially match with the interactive content. There are many examples of the LM controller used to control actions of a hand displayed on a desktop to modify the content within the display, therefore not accessible to the space where the hand is located.

A *direct touch* interface is an interface where the user directly interacts with the content by physically touching its reproduction. An example of its implementation can be found in the prototype called *Holodesk* [21] and its evaluation has been proved to be helpful for certain tasks in other studies [34, 4]. The direct touch interface is intuitive and easy to use because the user directly touches the 3D image displayed in front. This interface is suitable for systems used by people who are new to or unfamiliar with the system, such as public terminals on the street. Not requiring from the user to imagine beforehand the movement within the interaction space might potentially help to increase the usability of this kind of interfaces.

There exist other approaches related to VR implementations in avatars and physical simulations that consider the realism of the movement and the physical simulation of the contact as important metrics to evaluate the interaction (for example, [57]). An evaluation of this kind approaches compared to the here pursued direct interaction should also be studied to find what is the best way the user can interact with the virtual content.

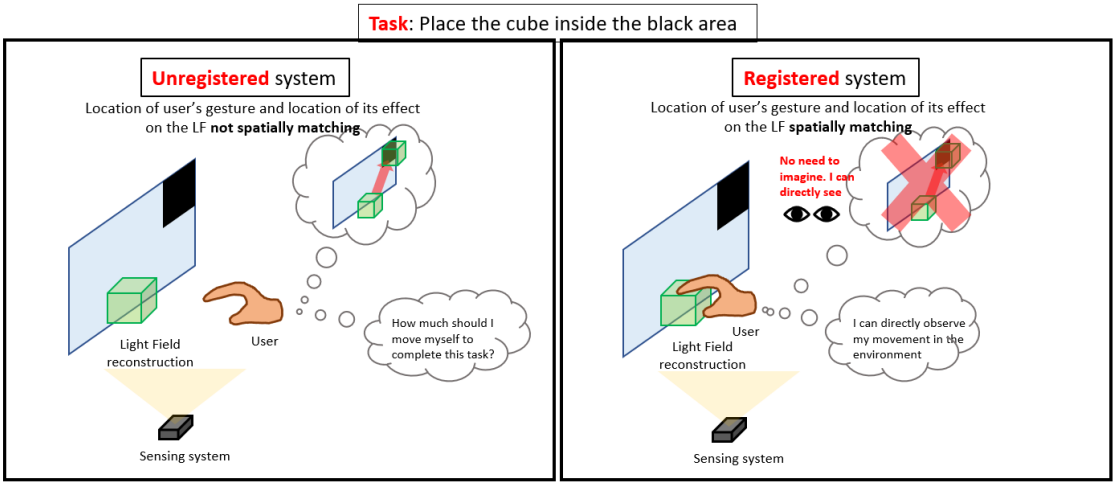


Figure 5.1: Difference between registered and unregistered GUIs. If the position of the user is correctly registered, the user does not need to imagine beforehand how the performed actions will be reflected in the interface when operating it.

Although we consider the approach proposed in chapter 4 of detecting scattered light along with its color could be further developed to be a standalone 3D tracking method, we are also interested in evaluating the potential of this interface with gestures and see how

valuable can the introduction of direct touch be. Since the commercially available devices have already well-developed 3D tracking and gesture detection routines, with functions available in Software Developer Kits (SDKs) for several programming languages, we first propose a way to combine our method with the available technology to harness the capabilities of already existing devices. Once these two approaches (developed gesture sensing and registration of the interaction using scattered light), we can evaluate under what circumstances registering the LF with the position of the hand is meaningful and/or helpful.

Another advantage of using off-the-shelf methods is that they not only track one position of the user, but they are also capable to track several features of the hand such as the position of every finger, the orientation of the palm (yaw, roll, and pitch angles) and even the location of the bones of the hand in the LM case, among other useful information.

Several other methods to create a 3D GUI have already been presented in section 2.6. In general, most of them lack the capacity to register the interaction with the content. The approaches that do solve the registration problem, have either to work on updating the LF in a sufficiently fast way or implement the use of gestures to interact with the reconstructed content. This research aims to solve this problem and present a method both fast, and with registered position and gestures.

5.2 Proposed method

The method we propose to align the position of the user can be summarized as follows:

Consider the LF display reproduces some 3D buttons with different colors so that a user can choose one of them. As shown in Fig. 5.2, the light scattered by the user' s fingertip is captured with a color camera and the region of detected scattered light is segmented. Then, this segmented Region of Interest (ROI) is processed and averaged into an RGB vector $\vec{C}_d = [R, G, B]$ using the pixel values of the captured image. This value is compared to the color vectors of the 3D buttons to identify which button the user touched, just as detailed in section 4.5.3. Since we can control the color and position of the UI objects within the display coordinate system ($\mathbb{R}^3_{Dis.}$), the color can be used to identify additional spatial information. We use an RGB camera located on top of the system and the position of the fingertip within the camera space ($\mathbb{R}^2_{Cam.}$) to find the position of the interaction in 3D space. Therefore, color will let us identify a 3D position

whenever the camera detects a certain color.

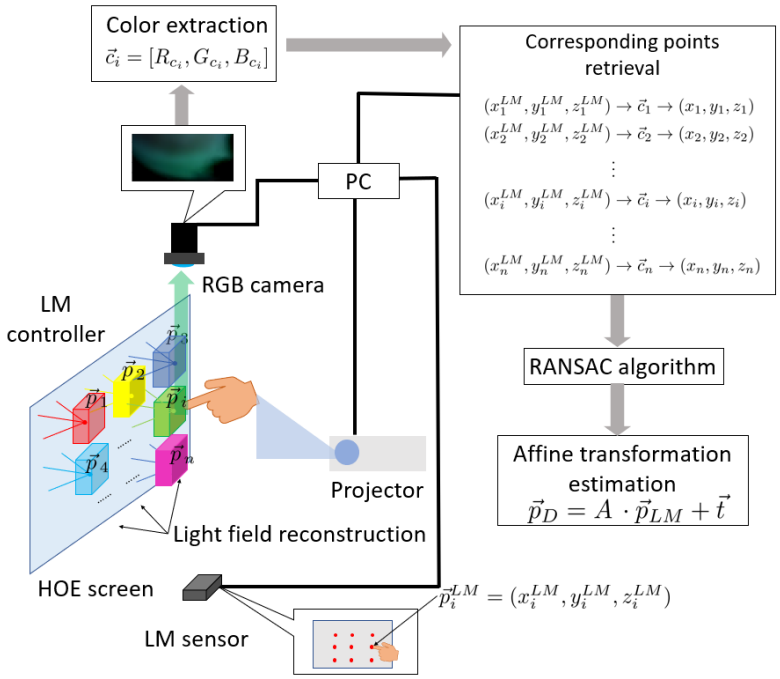


Figure 5.2: Proposed method to register the position of the LM controller using scattered light detection.

5.2.1 Fusion of the scattered light and the LM controller spaces

We detail the integration of the LM controller into our system in this section. We will harness the previously developed scattered light color detection to use this measurement to automatically register the position of the content of the LF display ($\mathbb{R}^3_{Dis.}$) and match it with the coordinate system of the LM ($\mathbb{R}^3_{LM}$) so we can obtain a more natural and direct interface. The theory of the problem reduces itself to the estimation of a geometrical transformation between the display space and the LM controller space. Different spatial points are used to estimate the transformation that takes the measured values in the LM-space to the rendering space of the LF display. Then, once the position of the hand or gesture can match the place of reconstruction of the LF object, the data from the LM controller are used for the recognition of finger movement.

As illustrated in Fig. 5.2, we have the HOE screen and projector to reconstruct the LF, an RGB camera located on top of the setup to detect the scattered light and the LM controller located at the bottom. All devices are connected to a PC that controls the system. Then, n LF objects are reconstructed at known positions ($\vec{p}_i \in [\vec{p}_1, \vec{p}_2, ...\vec{p}_n] \in \mathbb{R}^3_{Dis.}$) in the display rendering space. To each of these $\vec{p}_i$ positions we associate a

color vector $\vec{c}_i = [R_{c_i}, G_{c_i}, B_{c_i}]$, which will be detected when the user interacts with the LF and scatters its color. If we assign a different color to each position, we create a correspondence between the detected color c_i and the position $\vec{p}_i \in \mathbb{R}^3_{Dis.}$. In parallel to this process, the LM controller detects the position of the user in a point $\vec{p}_i^{LM} \in \mathbb{R}^3_{LM}$. The detection of scattered light provides to the system a cue that indicates that the user is directly interacting with the LF in a known position, therefore it creates the association between $\vec{p}_i$(position within the screen space), $\vec{c}_i$(color of the reconstructed button) and $\vec{p}_i^{LM}$(position in the 3D space) via the scattered light detection. By detecting the n points of the reconstructed LF objects, it is possible to estimate the transformation that matches the points in the display and the LM controller. In other words, we look for a transformation T that performs the mapping:

$$T\colon \mathbb{R}^3_{LM} \rightarrow \mathbb{R}^3_{Dis.} \tag{5.1}$$

The transformation in (5.1) can be modeled after a 3D affine transform, as a matrix $A \in \mathbb{R}^{3\times3}$ and a translation vector $\vec{t} = [t_0, t_1, t_2]$ obtaining the correct registration as follows:

$$\vec{p}_i = A\vec{p}_i^{LM} + \vec{t}. \tag{5.2}$$

The matrix A and the vector $\vec{t}$ are used to transform the measurements of the LM sensor into the reference frame of the LF display. The matrix A stands for the rotation of the coordinate system, while $\vec{t}$ is a translation vector from the origin of the coordinated systems. The estimation accuracy of A and $\vec{t}$ depends on the number of correspondences n between LF objects and LM points used to compute it. The method for estimating the matrix A and vector $\vec{t}$ is explained in the next subsection.

5.2.2 Transform estimation

The process we follow to retrieve display coordinates from LM coordinates is summarized in Fig. 5.3. We use an affine camera model to approximate the transformation from the LM space to the display space. This model is a simplification of the perspective camera model, representing the case of the orthogonal projection of a point in 3D space ($\mathbb{R}^3_{LM}$) on a plane in the display space ($\mathbb{R}^2_{Dis.} \subset \mathbb{R}^3_{Dis.}$) [20]. Let us call $\widetilde{p}^{D'} \in \mathbb{R}^2_{Dis.}$ (with tilde) a vector contained in this plane, and distinguish it from the full 3D vector $\widetilde{p}^{D} \in \mathbb{R}^3_{Dis.}$. The LF objects for calibration are reconstructed in the plane that contains $\widetilde{p}^{D'}$ during the transformation-estimation procedure and has a fixed distance z_0 away

from the HOE screen. When using homogeneous coordinates, we will augment the 2D vector $\widetilde{p}^{D'} = (x, y)$ to the notation $\vec{p}^{D'} = (x, y, z_0 = 1)$ (with top arrow), where the last element is a scaling factor. Likewise, the 3D vector $\widetilde{p}^{LM} = (x^{LM}, y^{LM}, z^{LM})$ is augmented to $\vec{p}_A^{LM} = (x^{LM}, y^{LM}, z^{LM}, 1) = (\widetilde{p}^{LM}, 1)$. This is a commonly used notation in projective geometry and computer vision applications [59]. Therefore, this transformation takes the form:

$$\vec{p}^{D'} = \begin{pmatrix} x^D \\ y^D \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x^{LM} \\ y^{LM} \\ z^{LM} \\ 1 \end{pmatrix}. \tag{5.3}$$

Compared with Eq. (5.2), in Eq. (5.3) the translation has been integrated into the matrix A. This transformation, different from a more general perspective transformation, has only 8 unknown parameters. Since each sampled point represents 2 equations, at least 4 points should be retrieved for the problem to have solution. Therefore, to acquire the required points we employ the LF objects reconstructed in a plane parallel to the HOE screen ($z = z_0$). This problem is solved with an implementation of the RANSAC algorithm for robust estimation [15].

Although the option of directly using a 3D version of the RANSAC algorithm is in theory possible, the experimental results of using this algorithm directly turned out to provide an unstable estimation of the mapping between the LM coordinates and the LF display The addition of an extra dimension (z-axis) makes it more challenging to use only RANSAC to estimate this transformation correctly. The reason of the instability of RANSAC for this procedure, as well as the consideration of using a more accurate method should be the topic of future work.

On the other hand, this process can be of aid when RANSAC in 3D does not work and needs to be applied in other scenarios.

Once the parameters of the matrix A have been retrieved, the projection of a point in $\mathbb{R}^3_{Dis.}$ can be located in a plane $\mathbb{R}^2_{Dis.} \in \mathbb{R}^3_{Dis.}$. To obtain the z^D coordinate that provides the full 3D position in $\mathbb{R}^3_{Dis.}$, we first estimate the equation of the plane generated by the N detected points $\vec{p}_1^{LM}, ...\vec{p}_N^{LM}$. This plane is ideally the one that minimizes the sum of Euclidean distances between itself and the set of N points. A plane can be described by its normal vector $\vec{n} = (n_x, n_y, n_z)$ and the equation

$$n_x x + n_y y + n_z z + c = 0, \tag{5.4}$$

with $c \in \mathbb{R}$. Considering a point in the LM space $\widetilde{p}_i^{LM} = (x_i^{LM}, y_i^{LM}, z_i^{LM})$, its distance from the plane is given as

$$r_i(\widetilde{p}_i^{LM}, \vec{n}) = |n_x x_i^{LM} + n_y y_i^{LM} + n_z z_i^{LM} + c|.$$

Therefore, we would like to find the plane whose normal vector $\vec{n}_{min}$ minimizes the sum of distances from all N points, namely

$$\vec{n}_{min} = \arg\min \Big(\sum_{i=1}^{N} r_i^2(\widetilde{p}_i^{LM}, \vec{n}) \Big). \tag{5.5}$$

It can be shown [16] that a Singular Value Decomposition (SVD) of the $3 \times N$ matrix B formed with the $\widetilde{p}_i^{LM}$ points measured can provide the optimal plane within the ℓ_2 norm. The SVD decomposition of the matrix B is given as:

$$B = U\Sigma V^T, \tag{5.6}$$

where U and V are 3×3 and $N \times N$ orthogonal matrices and Σ is the $3 \times N$ matrix of singular values σ_i where the first 3×3 partial matrix is given by $diag(\sigma_1, \sigma_2, \sigma_3)$, and the other part is a zero matrix. The vector $\vec{n}_{min}$, which defines the best plane in the least-square sense as in (5.5), is given by the third column vector of U, which corresponds to the minimum singular value [16]. Once this plane is defined, and assuming an orthographic projection from the LM space to the display plane, we can obtain the z^D position in the display space by computing the distance between the plane defined by $\vec{n}_{min}$ and the read-out point by the LM sensor $\widetilde{p}_i^{LM}$ as follows:

$$z^D = \beta |\vec{v} \cdot \vec{n}_{min}|, \tag{5.7}$$

where $\vec{v} = \widetilde{p}_i^{LM} - \vec{p}^{pl}$, is the vector that goes from a point in the plane $\vec{p}^{pl}$ to the readout point by the LM sensor. This distance is scaled using the β parameter, from the units of the LM space to the ones of the display space.

5.3 Registration error measurement

To evaluate the accuracy of the registration method inspired in the one reported in [1]. The position of the finger detected by the LM (x^{LM}, y^{LM}, z^{LM}) is transformed to a position in the LF space (x^D, y^D, z^D) using the estimated affine transformation along with the z distance from the adjusted plane. A color tile is reconstructed in a previously defined LF position. The accuracy test consists on moving the finger to the tile position

and compare the 3D transformed position with the ideal position used for rendering the tile.

A number L of points are randomly picked for the test. Let $j \in [1, ..., L]$ and each 3D point of the LM transformed to screen coordinates be denoted as $(x_j^{D'}, y_j^{D'}, z_j^{D'})$. The reconstructed tiles are reproduced one each time in the LF space centered at the 3D point (x_j^D, y_j^D, z_j^D), each having a square size of 20mm $\times$ 20mm.

The LM provides a large stream of readings that can change with the tremor of the hand. Another thing to consider is that the finger scatters color within the volume of the reproduced tile, so as long as the finger scatters color inside the tile, the system considers the finger as being in the center of it. Therefore, the reading $(x_j^{D'}, y_j^{D'}, z_j^{D'})$ for the position of the reproduced tile j, is obtained by taking an average of 50 readings of the LM while the finger scatters the color of the tile.

The average error of $L = 10$ points between both ideal and measured positions in this experiment is $E = 4.02$ mm, which is likely to be acceptable for this application. A depiction of this accuracy evaluation is shown in Fig. 5.4.

5.4 Integration into a GUI

The calibration process is a requirement for the system to work correctly, not tied to the task the user wants to perform and certainly not saving time for the user. Therefore, it would be of help that the calibration procedure could be integrated and combined with the tasks the user needs to perform. The use of scattered light in-line with the interaction procedure was proposed as future work in section 4.8. Although in this study we are not considering having in-line correction of the obtained calibration, we implemented an integration with the calibration procedure into an ATM-like GUI that can capture the required spatial position and realize the calibration procedure by acquiring the color-3D space correspondences required. The idea is that the user will input the basic information of the transaction required. In our implementation, as depicted in Fig. 5.5, the user presses the button that corresponds to "withdrawal" and presses an "enter" key. These actions already capture two of the minimum 4 required positions to estimate the affine transformation. In the next stage of the GUI, the buttons are located slightly lower and the options of some withdrawal amounts are presented. By choosing one of them and pressing "enter" once more, the user can finally match the 3D points with the position of the LF images, achieving the registration. Naturally, each of the options are represented by a button of different color, so the system can identify what button is being *pressed*

by scattering the corresponding scattered light.

We could think of several other scenarios in which this registration method is used. For example, a desktop computer usually requires a PIN number to be unlocked. The input of this PIN code by a numeric pad with different colors could also provide this functionality. If this type of displays gets adapted to work in an automobile, then the input information could correspond to keys such as "start engine", choose the position of the gearbox ("Parking", "Drive", "Reverse", etc.), adjust of rearview mirrors by pointing to each of them, which could provide even a safety feature by forcing the user to check the mirrors before start driving.

The GUIs registered with this kind of interface could have many applications: For a desktop-like system (see Fig. 5.6), to have a registered interface enables many creative functions such as drawing and editing pictures and performing organization-related tasks. In the case of the ATM-like interface, it could work as a way to input a 3D pattern for identification, similar to the one used by some smartphones to unlock the screen. To add a pattern of this nature along with the PIN code could provide extra security to the clients of the bank. More details about this possible application will be discussed in the next chapter.

In the case of a registered car interface, it could also provide a way for the driver to point to a certain feature in the road and display information about it, to trace a route on a map to the navigational system, to measure the distance between the origin and the destination, among other possible developments (see Fig. 5.7).

The proposed GUIs (ATM, desktop PIN code, car GUI) are ideas on how we can use the direct touch and light scattering to provide a spatial registration check between the generated content and the position of the user. Early touchscreens in 2D displays required calibration routines[13], specially the ones based on resistive technology due to the deformation of the Indium Tin Oxide (ITO) layer used to detect the touching point. With prolonged use, these layers would deform and warp, changing the performance of the touchscreen. Capacitive touchscreens do not have this problem, not requiring a calibration procedure [24]. In the case of the proposed 3D touch method, depending on where the screen is located, the material and the position of the sensors might be modified by several factors. The HOE polymer is prone to shrinkage and deformations due to temperature and humidity changes. Since there is still not a definitive 3D touch architecture, it is very likely that a calibration routine like the one proposed here is required as these interfaces evolve gradually and become more widespread.

In that sense, the study of the here proposed approach and the study of this problem is of great relevance to the advancement of effective 3D GUIs.

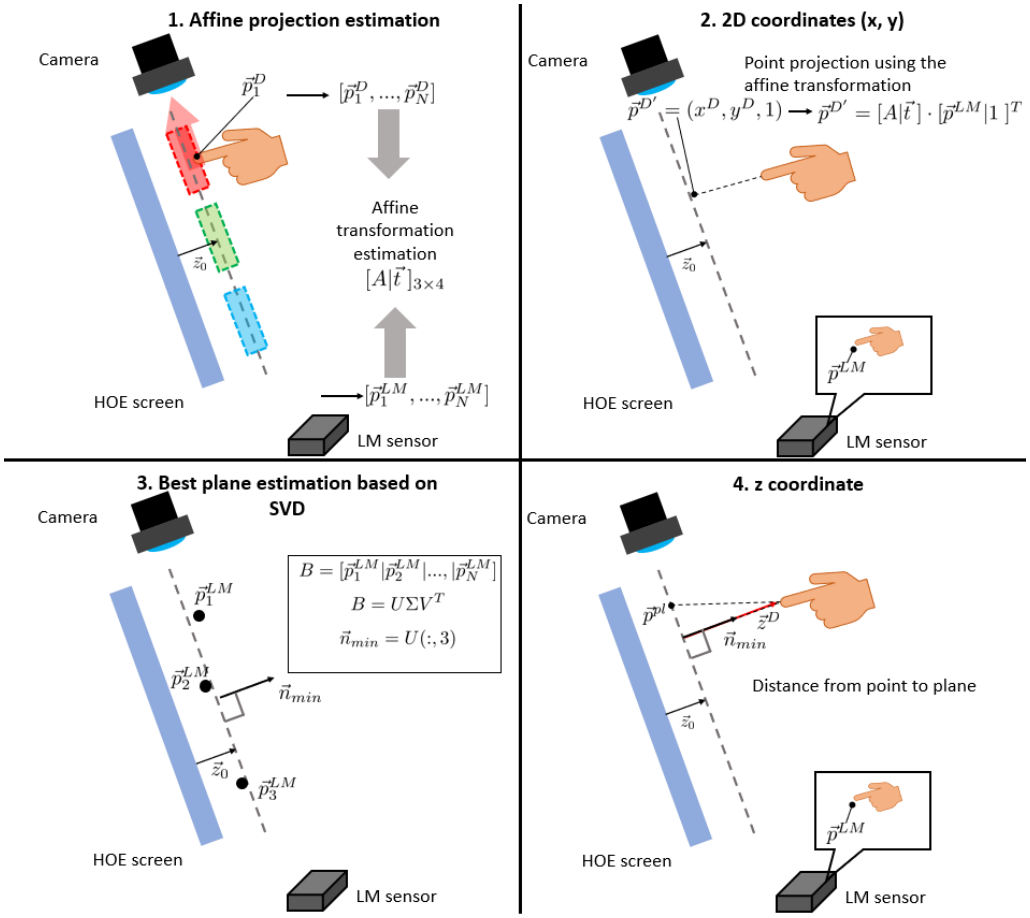


Figure 5.3: Use of the affine transformation estimation and the retrieved points in LM space to obtain the coordinates in the display space.The rendered points $\vec{p}_i^D$ and $\vec{p}_i^{LM}$ are matched using color and the affine transformation computed (1); the transformation projects the 3D points of the LM in the determined plane at z_0 (2); the best fit for a plane using all the LM points is computed using SVD (3); finally, the distance from the $\vec{p}_{LM}$ to the computed plane is used to obtain the z coordinate by transforming this distance into units of the display space (4).

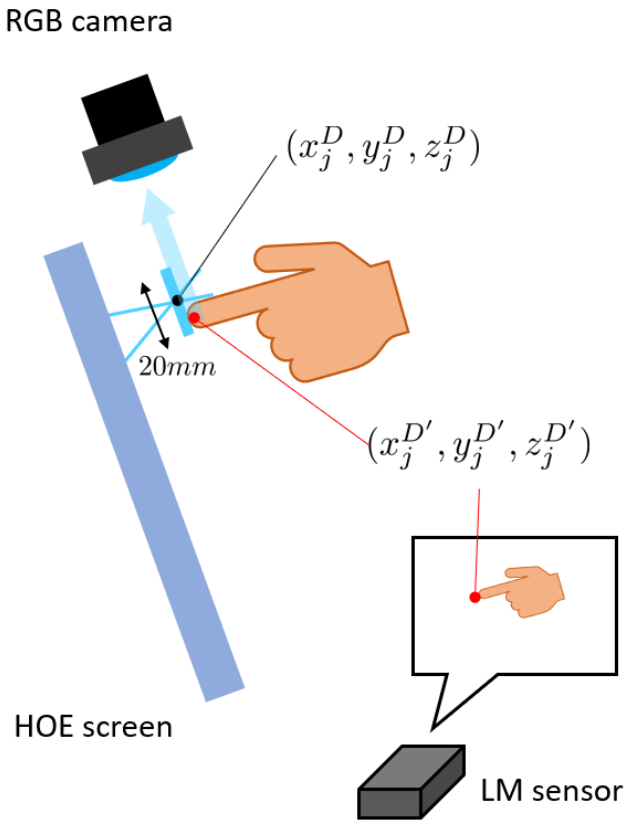


Figure 5.4: Measurement of the registration accuracy: a number of $L = 10$ tiles were reproduced in randomly picked locations within the display space. The discrepancy between the rendered position (ideal position) and the measured one by the LM after being calibrated by the proposed method, was computed.

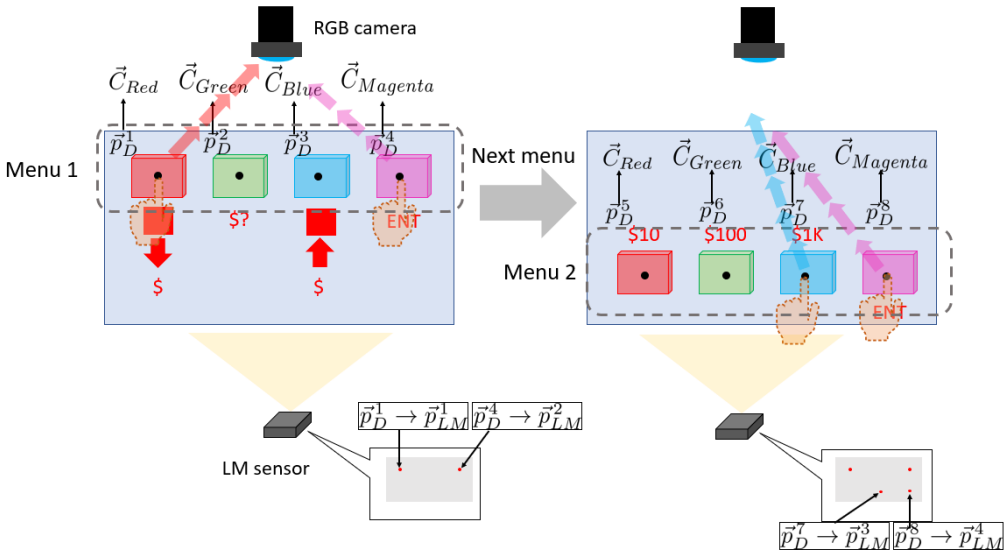


Figure 5.5: Example that shows how a GUI interface is capable of registering the position between the LM controller and the LF display.

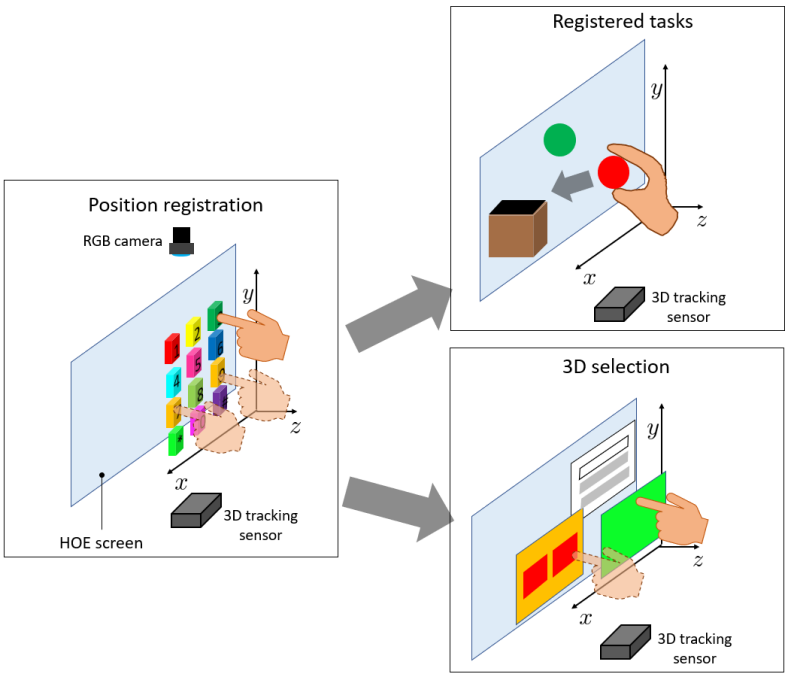


Figure 5.6: The input of the PIN code in a system can be used to register the interaction. Once the interaction is registered, it can be applied to several tasks like sorting out objects or selection of items based on depth.

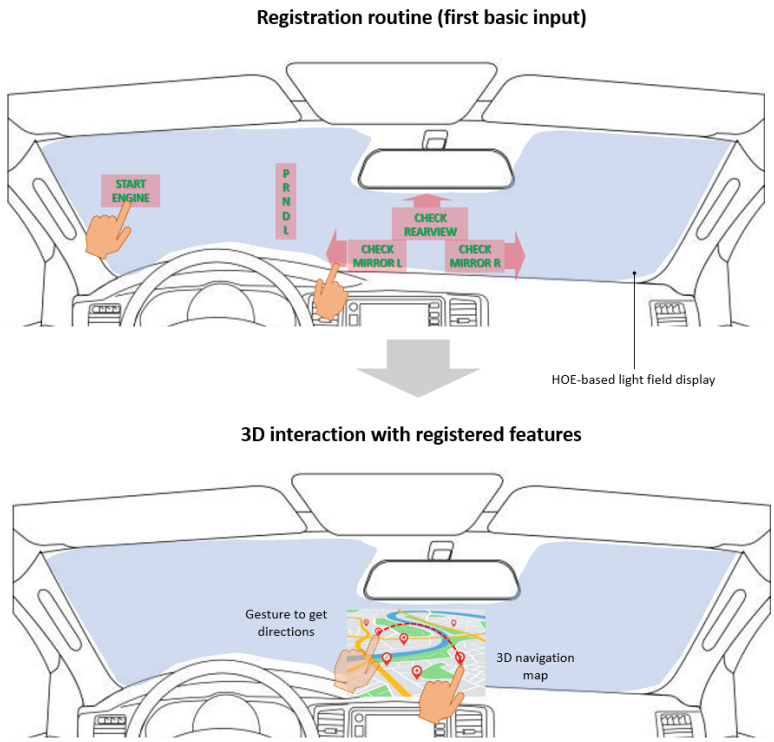


Figure 5.7: Idealization of a car using a 3D display where the user registers the position of the hand and the sensors using the initial required commands to drive. This GUI can also be registered and re-calibrated each time the driver starts a car (top). This registration can be used for several functions in the GUI (bottom).

Chapter 6

Applications of registered sensor

The aim of registering the position of the user end the reconstructed LF is to have an intuitive interface that a user can use immediately and without requiring to learn any complicated operations. To achieve this, we can make use of the daily life interactions of lifting objects and moving them from one place to another, but also of writing, tracing and drawing abilities, as well as the gestures that have been developed and become popular through the use of smartphones and 2D touchable screens (e.g., swiping, pointing, tapping, etc.). However, it could be that some of them are not adequate to implement in a system due to the absence of any physical feedback like a solid screen. Further research on this type of interfaces is required to answer these questions.

In any case, here we introduce some demonstrated interactions using the sensor registered after the procedure described in chapter 5.

6.1 Real time free drawing

We demonstrate 3D free drawing with the user's finger. The user can draw an LF *path* that follows the finger's trajectory within the 3D space in real-time.As mentioned in the previous chapter, the RGB camera goes offline, and the calibrated LM data is used for finger tracking. The effect of the calibration in the free drawing of LF is shown in Fig. 6.1. Although the fingertip and the trajectory drawn in the LF display are separated in the upper sequence before the calibration, the drawing is well registered with the fingertip in the bottom sequence after the calibration. The pictures are taken from the user's viewpoint. Fig. 6.2 shows another example.

This tracing is different from the one reported in aerial displays because it is capable of

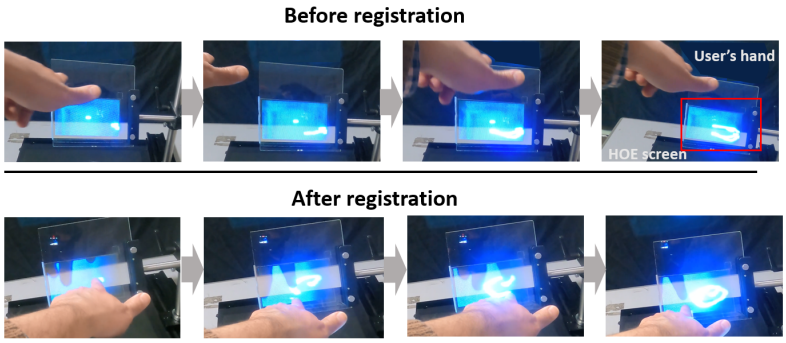


Figure 6.1: **Registration result**: Before the registration procedure, the place of the gesture and of LF reconstruction does not follow the finger of the user(upper sequence), whereas after the registration procedure, the location of the LF reconstruction corresponds to where the user passes the finger, drawing a circular trajectory (lower sequence).

reproducing a 3D trajectory and not just the tracing in a single plane. As mentioned in chapter 5, this can be further used in the future as an additional way to input a pattern into an ATM device or any other system that requires authentication from the user (see Fig . 6.3). This method has already been proposed, and it draws attention as documented in [41, 17, 42]. Of particular relevance is the work published in [41] where the difficulty of not having an input medium is also discussed.

6.2 Registered gestures *grab* and *poke*

Since the LM controller provides not only the position of a finger of the user, but a complete model of the hand and the position of all the fingers and normal vectors, these locations are used to implement *grab* and *poke* gestures and interactions with them (see Fig. 6.4). Although here we introduce only these two examples, the positions of the fingertips, along with the palm location, Euler angles-based orientation and other information provided by the LM controller, can potentially be used to implement more complicated functions (rotating and examining an object, deforming it, disassembling it, etc.) This should be part of the future development of this GUI. In the *grab* gesture, the positions of the index finger ($\vec{p}_{ind.} \in \mathbb{R}^3_{LM}$) and the thumb ($\vec{p}_{th.} \in \mathbb{R}^3_{LM}$) are acquired and the Euclidean distance between them retrieved ($d = \|\vec{p}_{ind.} - \vec{p}_{th.}\|$).

In principle, if the value was under a minimum distance, the system identified this finger position as the user grabbing a LF object. The function would take the form:

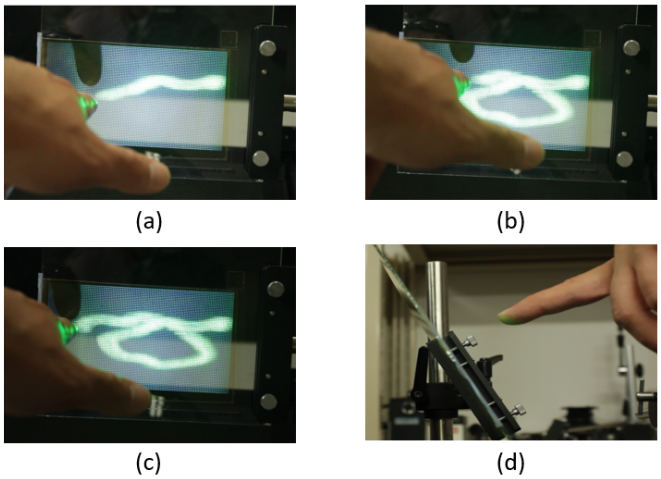


Figure 6.2: Real time free tracing of the finger trajectory into LF using the computed registration (a-c). The registration reconstructs the LF in the tip of the finger of the user (d).

$$grab(d) = \begin{cases} 1 & d \leq th, \\ 0 & d > th \end{cases} \tag{6.1}$$

However, we experimentally found that this way of identifying the gesture was not very robust since the LM controller, due to noisy readings, usually miscalculates the distance between the thumb and the index fingers and indicates a user has stopped grabbing an object even if that is not the case. To account for the measurement errors, a strategy involving 2 different thresholds was tried: Setting a *grab* threshold $th_{gr.}$ and a *release* threshold $th_{rel.}$, and redefining the gesture function as:

$$grab(d) = \begin{cases} 1 & d \leq th_{grab}, \\ 0 & d > th_{rel.} \end{cases} \tag{6.2}$$

With these couple of thresholds, the user can grab the LF object and the system would not consider the user dropped it until the fingers had a significantly larger distance than the one used for grabbing. Therefore, when displacing the LF object, the tracking errors of the LM during the displacement of the hand would affect less severely to the threshold-based strategy. The buffer zone created by the couple of thresholds provided robustness to the gesture identification (see Fig. 6.5). The picture of the actually implemented interaction is depicted in Fig. 6.6.

In the same fashion, the information provided by the LM controller was used to show

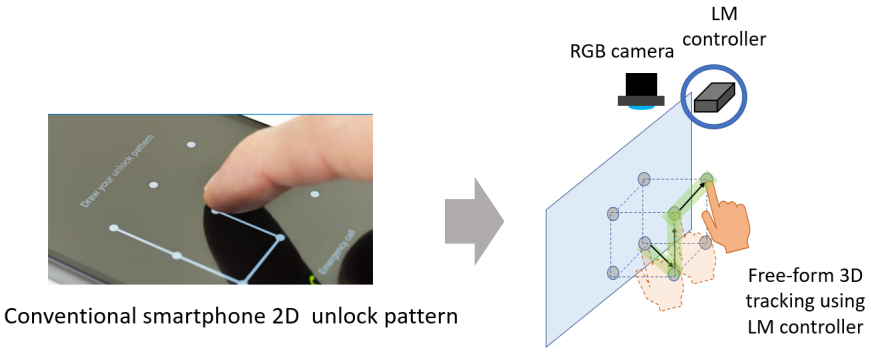


Figure 6.3: Commonly used 2D unlock patterns for smartphones. The proposed method could be used as a way to input a 3D unlock pattern for personal identification. Source: [78]

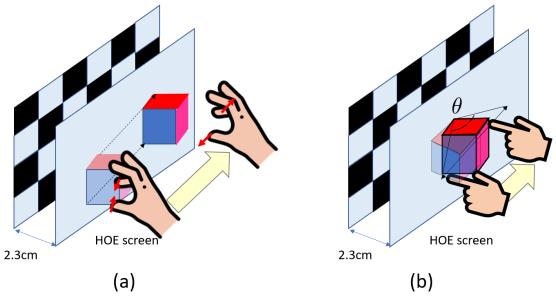


Figure 6.4: Illustration of the implemented gestures in the holographic LF screen: Grab, move, and drop gesture (a) and poke and spin gesture (b).

a *poke*-like interaction that consisted on the user poking the face of the LF cube and spinning it, as if it were laying on a table. The way to implement it was by measuring the distance between the fingertip and the center of the cube (surrounding sphere, Fig. 6.7). The considered distance corresponded to the one from the fingertip to a sphere enclosing the reconstructed cube, which differs slightly to the distance from the fingertip to the actual face of the cube. However, we consider this model as a simplification to demonstrate the system capabilities. To spin the cube, the angle formed by the vector formed by the center of the cube and the fingertip and the one formed by the center and the perpendicular to the face of the cube, was considered. Then, the cube moved to the direction of the vector resulting from the subtraction of both, so both vectors get aligned.

Similar to the *grab* gesture, this rotation was implemented to the render of the cube whenever the distance d from the surface of the surrounding sphere of radius r to the center of the cube satisfied:

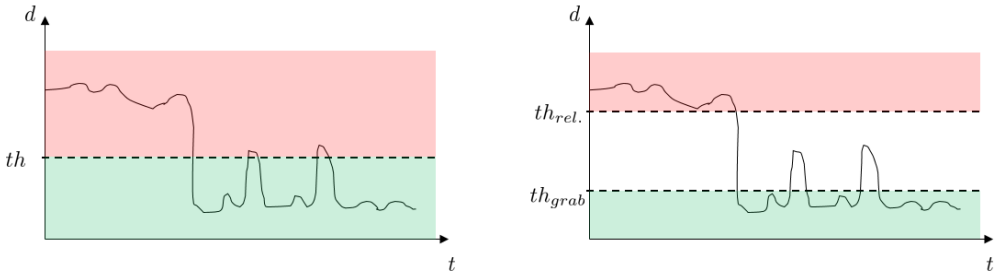


Figure 6.5: Comparison of the detection of *grab* gesture when using two thresholds and when using a single threshold. The value d stands for the thumb-index fingers distance. Notice how the noisy readings are better managed by the created buffer zone.

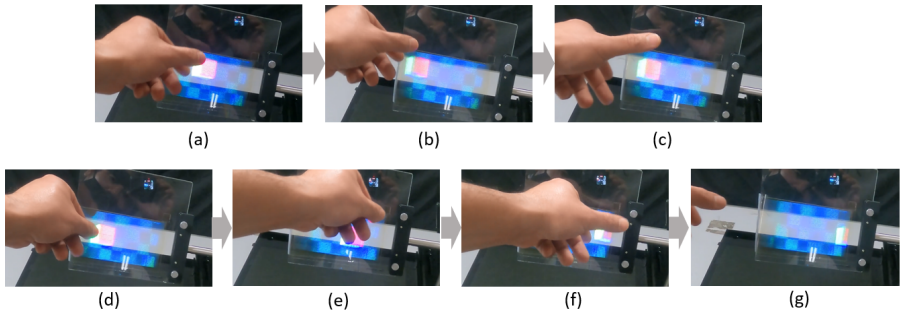


Figure 6.6: Gesture detection after registration with the LF screen (compare with Fig. 6.4(a)): **Grab** gesture where the user is able to grab the reconstructed cube (a), take it to the left edge of the screen (b), drop it (c), grab it once more (d), take it to the right edge of the screen (e), drop it (f) and remove the hand leaving the cube in the right edge (g).

$$poke(d) = \begin{cases} 1 & d \leq r, \\ 0 & d > r \end{cases} \tag{6.3}$$

If $poke(d) = 1$, then the render is turned to the direction of the difference between both vectors, as shown in the diagram. The video extract of the real implementation of this gesture is shown in Fig. 6.8

More sophisticated approaches that include the full rotation of the render and the modification of features of the object with the use of gestures can be potentially implemented. The accuracy of the registration approach can also be improved if we consider some proposals of using the LF detection along with the LM controller from section 4.8. However, the significance of advancing that research is still unclear if first the requirement of the registration is not evaluated. For that end, some other tasks that implement the

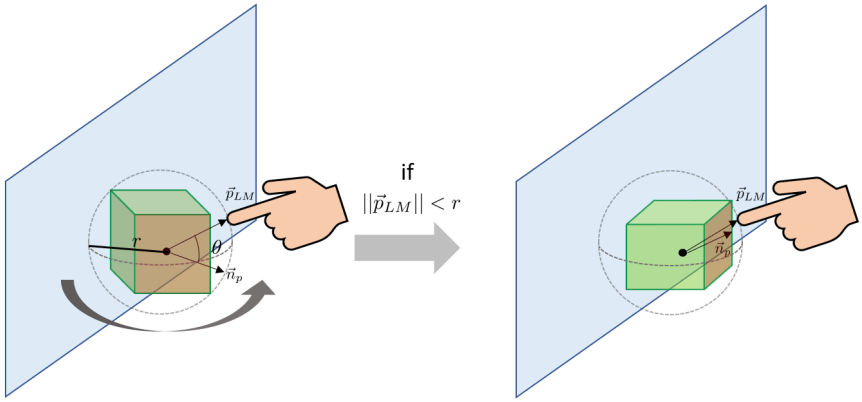


Figure 6.7: Measurement of the angle that pushes the LF cube

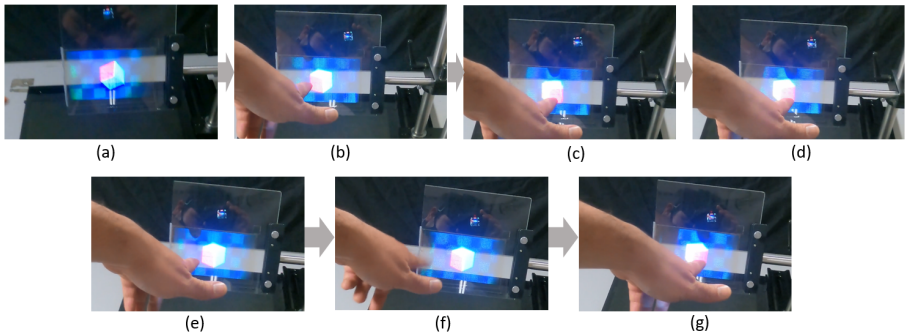


Figure 6.8: Gesture detection after registration with the LF screen (compare with Fig. 6.4(b)): **Poke** gesture where the user can move the cube by poking it and inspecting its different faces, as one would do with a real cube. First, the cube is reconstructed (a), then the user pokes it and the red face comes forward (b), the user moves the finger to the right and the red face moves towards the right (c and d), poking it so the the red side faces back to the left (e and f) and back again to the right (g).

grab gesture and the tracking of the finger were also developed in order to elaborate a set of them for evaluating the system.

6.3 Evaluation tasks

To measure the impact of having registered gestures, we designed some tasks whose completion time and subjective impression of the users would provide feedback to know how necessary this procedure is. The tasks used the tracking of the finger in the horizontal direction (unlock screen), the 3D tracking and the *grab* gesture (boxes picking-up) and the tracking of the finger in 3D to write a pattern in air (circle tracing). The details of their implementations are presented in the next sections.

6.3.1 Unlock task

Similar to the task presented in section 4.6, this task mimics the *swipe* gesture used to unlock conventional smartphones screens. In the referred section, this tracking was performed using the detection of the scattered light in the camera space. This time, using the registered LM sensor, the same gesture function was implemented. To make it more engaging, the color of the LF sphere was changed each time the user moved the ball to the edge of the screen. Besides, the user was required to swipe the ball 5 times (left→ right → left → ...) before the unlocked screen appeared (see Fig. 6.9).

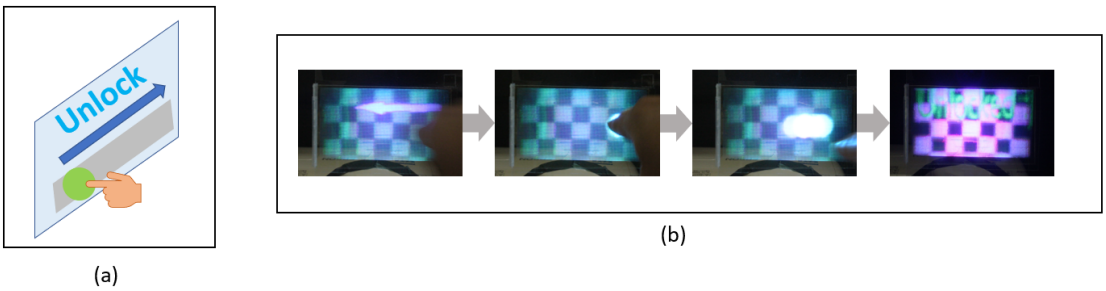


Figure 6.9: Screen unlock task: Diagram (a) and real implementation (b)

6.3.2 Box sorting

Using the *grab* gesture explained in the previous section, the render program was changed to depict three boxes in different locations and take them to a goal located in the upper right part of the screen, which was drawn using a blue torus. Whenever the center of the boxes reached the coordinates of the goal, then the box would be reduced and erased, mimicking a *vanishing* or an *absorption*. The box then would be counted as *goal* and the program would end once the user collected the 3 boxes. Since the area of the screen is rather small (120 × 67mm), the size of the boxes was reduced so three boxes could fit and be displaced within this small area (see Fig. 6.10).

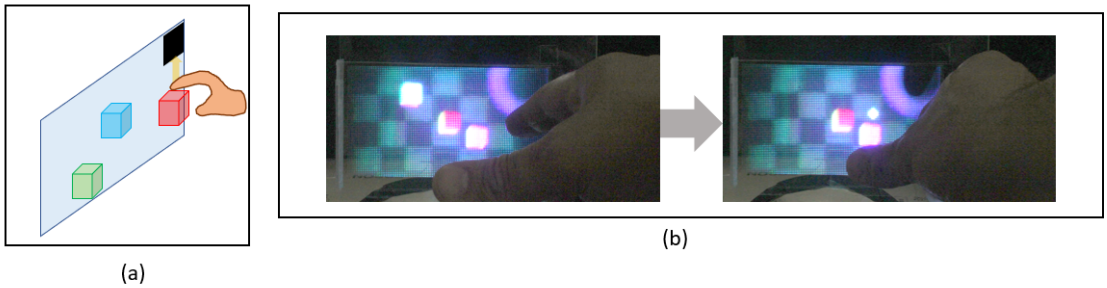


Figure 6.10: Box sorting task: Diagram (a) and real implementation (b)

6.3.3 Circle tracing

Finally, inspired by the free drawing demonstration from section 6.1, a task was designed to evaluate the difficulty of tracing a determined path when the system was registered and when it was not. The LF a circle was reconstructed in a certain color (purple) and the trajectory of the fingertip of the user was reconstructed in another color (green). The user in this task needs to pass the finger near the 3D coordinates of the circle. The coordinates are stored in a dictionary-like data structure, and whenever one coordinate of the circle matched the tracked finger within a certain tolerance, the coordinate is erased. After successfully matching a certain number of coordinates, the program finished (see Fig. 6.11).

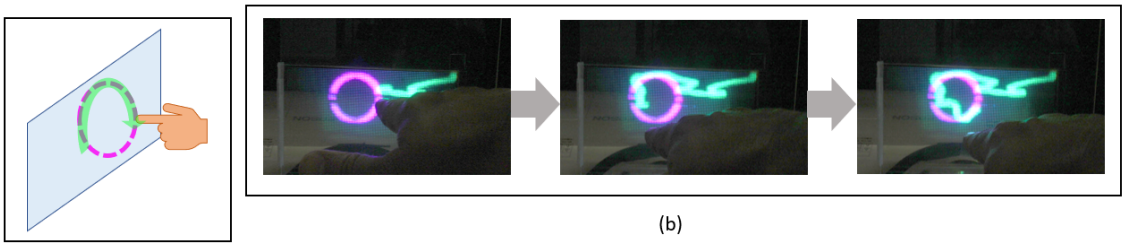


Figure 6.11: Circle tracing task: Diagram (a) and real implementation (b)

Chapter 7

User evaluation

There exist several studies that evaluate 3D interfaces using different paradigms and different tasks with a wide range of characteristics [1, 21, 4]. Early research on the need of registering the position of a user with the virtual contents can be traced back to [34], as it has been mentioned previously in this study. There is also later research that considers different scenarios where this *need* for registration is tested [36], finding that the isomorphic mapping (direct interaction) can at times be more demanding for the user when compared to a non-isomorphic mapping (indirect interaction) in the case where the non-isomorphic mapping helps the user to reach out for objects that are far to reach or can reduce the effort required for tasks. The use of different selection metaphors such as ray casting, virtual hands or voice commands has also been studied [3] and implemented in several studies, proving again to be more efficient than direct interaction under certain circumstances. In the last instance, the adequacy of using isomorphic or not isomorphic mappings will depend on the application, being isomorphic mappings more suitable in environments where strict realism of the manipulation is the major requirement [36]. Anyhow, the registration of the virtually generated content and the physical space (related to isomorphic mapping) is a necessary element for a mixed reality system. Considering all the previous research regarding the applicability of the direct interaction metaphor, there have been other studies that look at the significance of this registration when implementing a GUI. An early study on direct and indirect interaction using an eye tracking camera and an LCD with a beamsplitter was reported in [21]. In this prototype, as described in section 2.6, information from the Kinect sensor was used to detect gestures and 3D tracking of the hand of the user. The user, rather than having the LF reconstruction of an object, could observe a projection of a 3D view of a scene from different positions while moving the head. The results were a shorter

completion time when the user could interact directly with the content than when this was not the case. In the user evaluation reported in [1], the users had to locate their fingers in either 2D or 3D modes using a HPO LF display and a calibrated LM controller. Tiles were displayed in random positions and the user would require to locate its finger only in the right 2D coordinates to select a position in 2D mode or would be required to also match the 3D coordinate under the 3D mode version. Although a small significant difference between the accurate spatial location between each of the modes (2D and 3D) was found, being the 3D mode slightly more challenging to the users because of the existence of an extra dimension, no significant difference between the completion tasks was reported. This study is very useful for our evaluation, since it includes a LM controller and a LF display, although an HPO one.

In this study, our contribution is also to explore how the registration and the mismatch between the position of the user and the reconstructed LF can affect the completion of a task, but this time using an FP display, where the user can more accurately locate the finger thanks to the extra vertical parallax cues present. Also, different from [1], explore how this registration aids the effectiveness of gestures by testing versions of our GUI where the coordinates match and where they do not. We used for registering the position of the user and the content the method reported in chapter 5. The task-completion time (TCT) and the subjective user evaluation using a likert scale were retrieved. In this chapter we describe the experiment, present the results and discuss them to find a conclusion.

7.1 Experiment

To prove the effectiveness of the registration in both the TCT and user experience, we designed an experiment where the user performed different tasks with three different scenarios: (a) unregistered interaction, (b) registered interaction and (c) registered interaction with a randomized mismatch. The difference of each of the tasks is described in the following paragraphs.

7.1.1 Unregistered case

In the unregistered case, the user would be required to perform a task using the LM controller and using a cursor in the screen as a way to see where the position of the finger was reconstructed inside the 3D display space. No spatial matching was considered at all, but the user could not move the controller from its fixed position either. Therefore,

the user needed to imagine the movement and the speed of the interaction required to complete the task before performing it, as depicted in Fig. 5.1. After the user was required to press a button in the main PC, the program of the required task would run and a 5 seconds-long beep sound was played. After the beep, the time would star running and the user was instructed to perform the task. Once the system detected the user had finished the task, the time would be stopped and the same beep would be played to indicate to the user the task had been successfully completed (see Fig. 7.1).

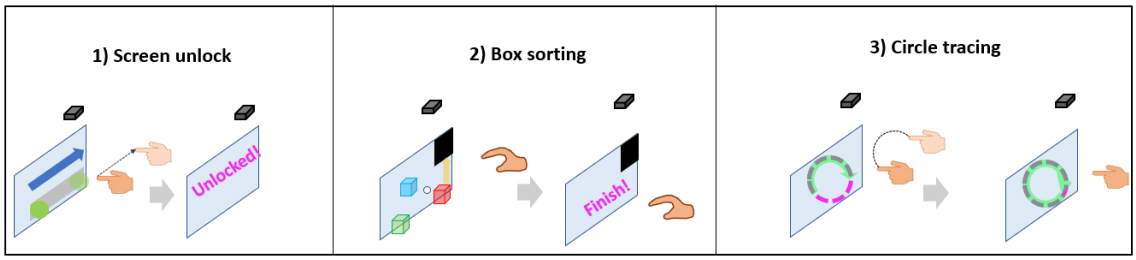


Figure 7.1: Tasks in the unregistered case

7.1.2 Registered case

In the registered case, a system exactly the same as the one for the previous version, but with the estimated transformation obtained as detailed in 5, was used to perform the same task. This time, the position of the user and the reconstructed object were matching more precisely. The same execution process was performed (i.e., press button→beep sound→time starts→task completed→time stops→beep sound). An illustration showing the tasks in the registered case is shown in Fig. 7.2.

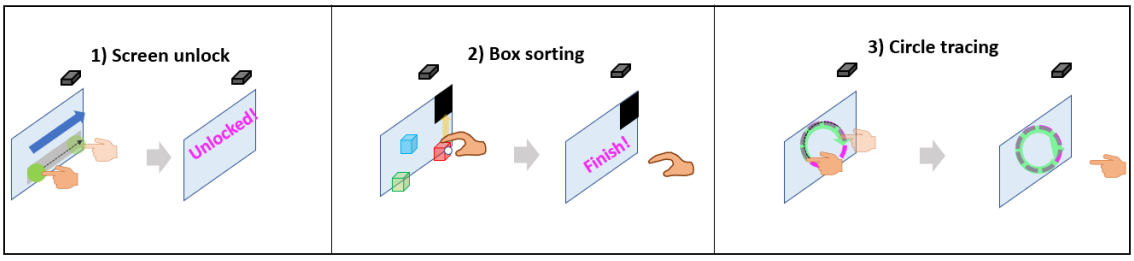


Figure 7.2: Tasks in the registered case

7.1.3 Registered case with random mismatch

Finally, in the registered case with a random mismatch included, the same transformation as in the case of the registered case, was used. However, the position $\vec{p}_{LM}$ of the user in the LF space was displaced by adding a random displacement vector $\vec{\Delta p}_{mis.} =$

$(\delta x, \delta y, \delta z)$ in each trial. The random displacement was limited to be no larger than half the total displacement possible in each dimension of the screen. In the case of the z-axis, it was limited to $\pm$20mm. Adding a random displacement vector while keeping the affine transformation allowed the system to keep the scaling of the interaction space, although affecting the exact matching between the user and the content for evaluation purposes. This configuration can be considered as an intermediate step between the registered and the unregistered cases.

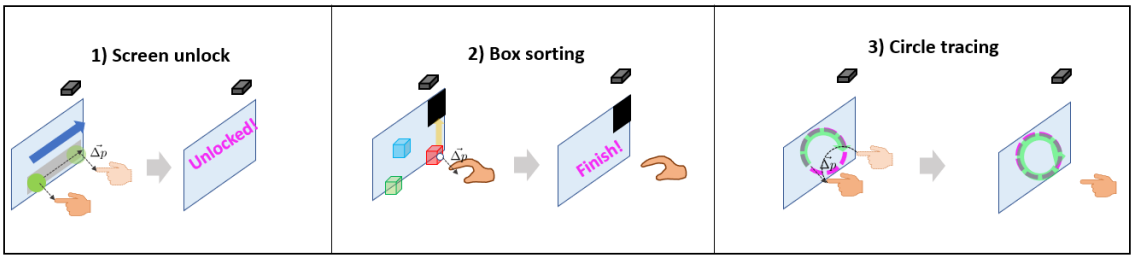


Figure 7.3: Tasks in the registered case with a random mismatch

7.1.4 Experimental procedure

The subjects were members of the Tokyo Institute of Technology, affiliated to the Yamaguchi Masahiro laboratory. 11 users performed the experiment. All of them had normal vision and no important visual problems that impaired them of seeing the 3D visual reconstructions. The users came to the experiment room, and they were explained that they would perform 3 kinds of tasks, each of them with 3 different versions, and 3 trials of each of them (a total of 27 trials, see Fig. 7.4). However, it was not disclosed what was the difference between each of the cases and the experimenter let the users find out by themselves. First, the experimenter himself performed the required tasks in front of them in the registered version, and they were instructed to observe. Secondly, the experimenter asked them to perform each of the tasks in the registered version as a practice. Then, the actual experiment began with each of the 3 versions (unregistered, registered and registered with random mismatch) of the 3 tasks (swipe, box grabbing and circle tracing) randomized in order. It was indicated to the users that they could press a button whenever they were ready to begin. The users performed each of the 9 version-tasks 3 times each, and the average time of the 3 trials was obtained. After the 27 tasks were completed, in order for the participants to fill the questionnaire, it was explained to them that the versions where the hand and the interaction had an important mismatch were the *unregistered* versions, while the versions where there was no mismatch or only a low mismatch were the *registered* versions. The participants were required to fill a

questionnaire that was used for elaborating the likert scale, where they would rate in a scale from 0 to 4 the tasks that they had performed. The questions that were asked for each of the tasks(screen unlock, box sorting and circle tracing) were:

1. How would you grade the *naturalness* of the interaction for: unregistered/registered interaction?
   →Very natural 0 1 2 3 4 Very unnatural
2. How would you grade the *difficulty* of the interaction for: unregistered/registered interaction?
   →Very natural 0 1 2 3 4 Very unnatural
3. How would you grade the *fatigue* of the interaction for: unregistered/registered interaction?
   →Very natural 0 1 2 3 4 Very unnatural
4. How would you grade the *novelty* of the interaction for: unregistered/registered interaction?
   →Very natural 0 1 2 3 4 Very unnatural

There was also a section where the participants could write their opinion about each of the tasks, to obtain a better understanding of the user experience with this GUI. Some pictures of the users performing the tests are depicted in Fig. 7.5.

7.2 Results

The TCT of the 3 trials of each task was averaged, to obtain 9 TCTs from each participant (1 for each version of the 3 tasks). The average times for each case and each of the tasks are depicted in Figs. 7.6,7.7 and 7.8.

It is noticeable that the unregistered task is the task with the highest TCT regardless of the task. The difference between the registered task and the registered task with a random mismatch if not as noticeable, but the registered version still has a small advantage.

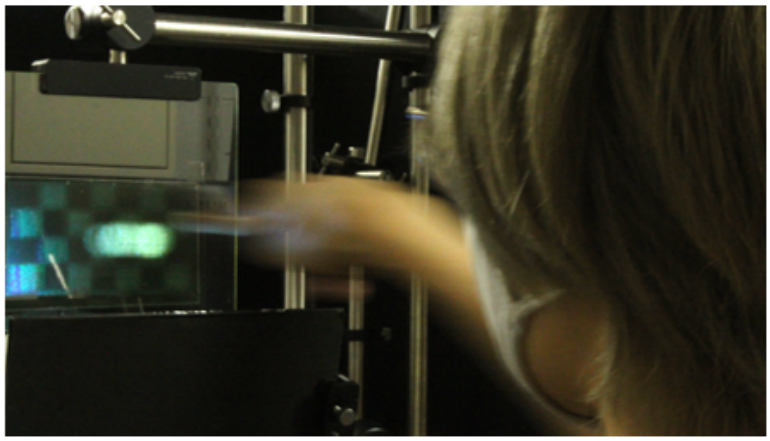
To test this results for statistical significance we performed an analysis of variance (ANOVA) study. By having the mean TCTs of all the participants in the three GUI conditions, we performed an ANOVA for each of the tasks. For each of the tasks, let $G = [1, ..., k]$ the k groups of the TCT for each condition. Let X be a TCT from any of the three groups, and n the number of variables (number of available data through

Figure 7.4: Example of how the required tasks and their versions were shuffled to minimize the learning effect among the participants

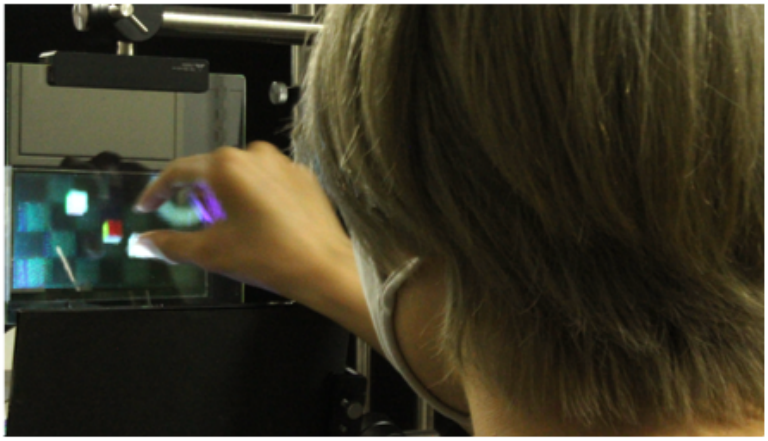
all groups). We first computed the mean sum of squares within groups as:

$$MSS_w = \frac{\sum_{g \in G}(X - \bar{X}_g)^2}{n - k}. \tag{7.1}$$

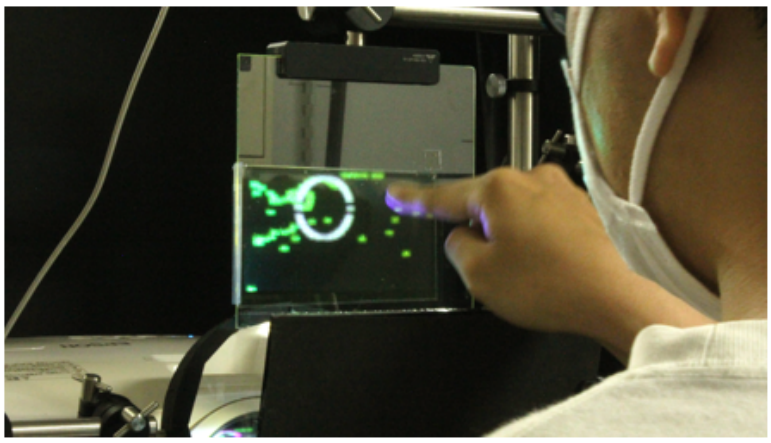
In Eq. 7.1, the value $\bar{X}_g$ represents the mean value of each group. Apart from this value,



(a)



(b)



(c)

Figure 7.5: Users performing the tasks: Swipe task (a), box sorting task (b) and circle tracing task (c)

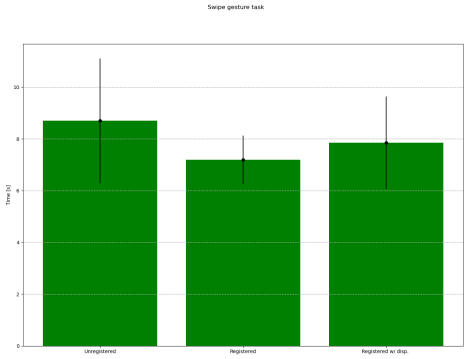


Figure 7.6: Completion time for the swipe task in each of its versions

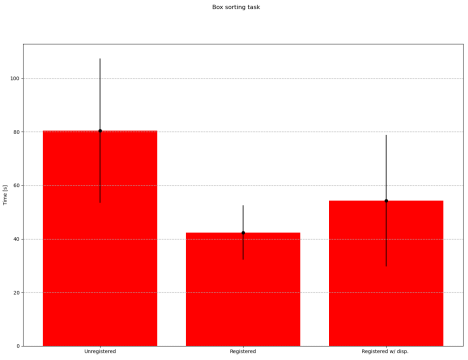


Figure 7.7: Completion time for the box sorting task in each of its versions

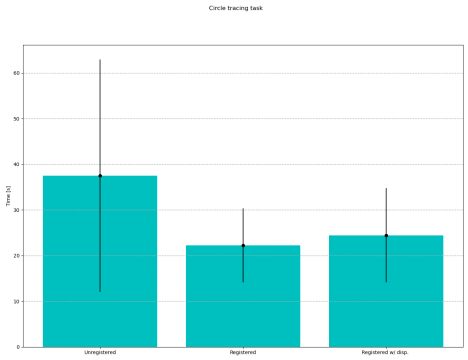


Figure 7.8: Completion time for the circle tracing task in each of its versions

we require the mean sum of squares between groups, which can be computed as:

$$MSS_b = \frac{\sum_{g \in G} n_g (\bar{X}_g - \bar{X}_G)^2}{k - 1}. \tag{7.2}$$

In Eq. 7.2, the value $\bar{X}_G$ represents the mean over all values of all k groups, while n_g is the number of variables in each group. The F statistic to test for the significance to

reject the null hypothesis is computed as the ratio:

$$F = \frac{MSS_b}{MSS_w} \tag{7.3}$$

In this case, the null hypothesis states that the mean values of the k groups are the same (i.e., there is no significant difference between each of the configurations). If the F statistic obtained is such that:

$$F > F_{(df_w, df_b)}. \tag{7.4}$$

In Eq. 7.4, $F_{(df_w, df_b)}$ stands for the critical value of the F distribution with df_w being the degrees of freedom in the denominator and df_b the degrees of freedom in the numerator. They are defined as:

$$df_w = n - k$$
$$df_b = k - 1$$

The value of $F_{(df_w, df_b)}$ is also dependent on the significance level α, from where the p-value is also calculated. In this case the number of degrees of freedom in the numerator are $df_b = 2$ and in the denominator are $df_w = 30$. We set the value $\alpha = 0.05$ The obtained values of the ANOVA test are presented in Tab. 7.1.

Task	**F-value**	p**-value**
Swipe	1.748	0.1914
Box sorting	7.924	**0.0018**
Circle tracing	2.487	0.1001

Table 7.1: F-statistic and p-value for the three conditions of the three different tasks.

The graphs in Figs. 7.9,7.10 and 7.11 show the averaged values of the likert scores to the questions for each task, described in section 7.1.4. The 'UR' acronym stands for the unregistered version while the 'R' stands for the registered version. Apart from the presented information, the subjective opinion on the user experience of each task was collected and some comments are discussed in the next section. The computed p-values shown in Tab. 7.1 show that the mean TCTs for each of the version of the box task are not the same ($p < \alpha$ →rejection of null hypothesis). Therefore, to clarify what version is different from the other, we perform a Bonferroni correction using the . The

p-value of a T-test is computed for every set of TCTs of the box version (unregistered-UR, registered-R and registered with a random displacement-R+Ran.). Since there are $k_T = 3$ tests, we obtain the Bonferroni-corrected $\alpha^* = \alpha/k_T = 0.0167$ and compare with the p-values of each pairwise comparison. The result of the comparison of each group are shown in Tab. 7.2.

Version	**p-value from T-test**
UR vs R	**0.0005**
UR vs R+Ran.	0.0349
R vs R+Ran.	0.1700

Table 7.2: p-value from the T-test of pairwise comparisons to perform the Bonferroni correction in the different versions of the box sorting task

It can be observed from Tab. 7.2 that the condition $p_{T-test} < \alpha^*$ is fulfilled for the comparison between the registered and the unregistered versions, therefore demonstrating a statistical significance between these two versions.

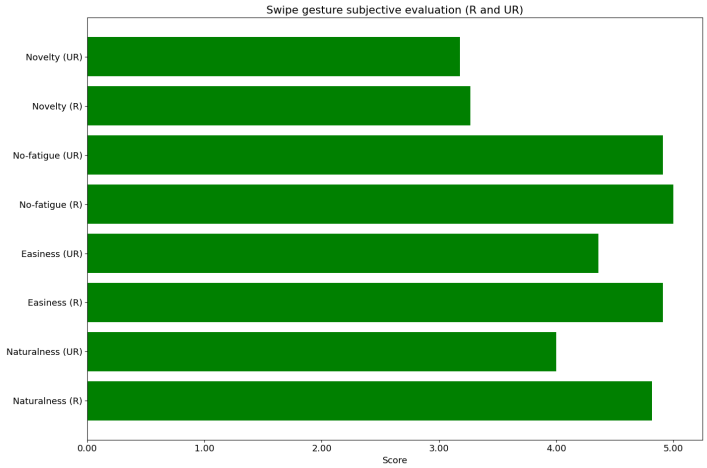


Figure 7.9: Likert averaged scores for swipe task

7.3 Discussion

The results in general are positive towards the registered interaction when compared to the unregistered interaction, although the gain depends on the performed task. We discuss each of the tasks, along with the objective results (TCT change depending on the configuration) and the subjective results.

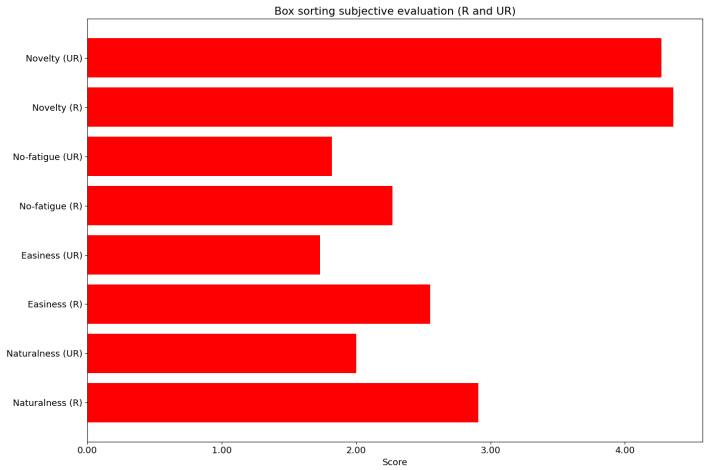


Figure 7.10: Likert averaged scores for box sorting task

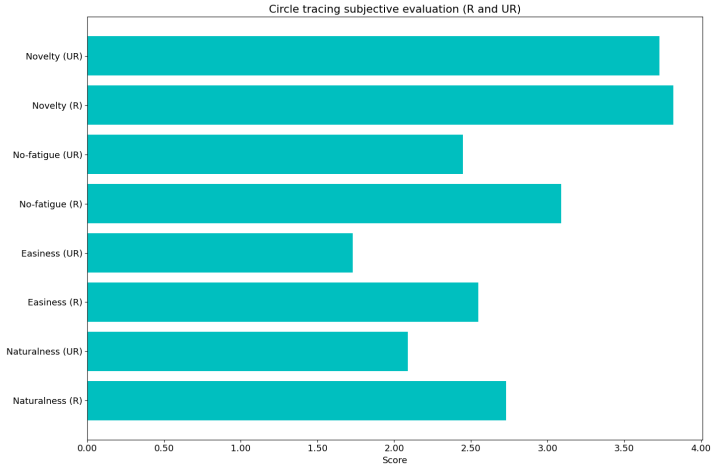


Figure 7.11: Likert averaged scores for circle tracing task

7.3.1 Swipe task

In terms of the TCT, this task was overall the fastest for every user across the three configurations. It is probably because this kind of gesture is very used, it is limited to a movement in one dimension, and it is not intellectually demanding. The average TCT for all the participants was in average shorter for the registered versions. However, the ANOVA test failed to show any significant difference ($p = 0.1914 > 0.10$), probably because of the small difference in the completion times along with the small size of the sample group.

It was observed during the experiment that, when using the unregistered version of

the program, the users would tend to hesitate if the system was responding to their movement, what made them move slowly and therefore increase the TCT. Although, as mentioned previously, it was not disclosed to the users what version they were using, some comments to the experimenter while performing the test while using the registered version were *'this version is smoother'* or *'this one feels better'*. The subjective evaluation also shows that this is the task the users felt more comfortable and where the system felt more responsive. Some factors contributing to these opinions were the fact of the relatively big touchable LF volume, where the user had a large volume to locate the finger, and the limitation of a single axis movement.

7.3.2 Box sorting task

The TCT difference of each of the versions for this task was the highest across all the participants. This test showed the highest significance in the ANOVA test ($p = 0.0018 < 0.05$),and the Bonferroni correction also showed that the comparison between the registered and the unregistered versions was the most significant one, indicating an important difference between the versions. The registered interaction proved to be the one with the lowest TCT, confirming that the registration is helpful in a more demanding task. One of the possible reasons of why this test represented the most significant difference in the TCT is that this test required the combination of a gesture with the spatial location of the user, requiring more focus by the user to match the cursor and perform the gesture when using one of the unregistered versions. Another factor was that this task, different to the other two, required to use a larger area of the screen. When using the unregistered version, the users tend to extend the arm to trying to reach to the goal located in the upper right position. Since they did not know how much they needed to extend the arm due to the lack of registration, they might sometimes stretch it out of the LM controller angle of view, therefore *dropping* the box and being required to pick it up again. The failure of the sensor to recognize the grabbing gesture at times, caused an extra difficulty that increased the TCT in the unregistered version.

Some observations during the execution of this experiment were the different ways the users had for interacting with the LF reconstructions. There were participants that turned their hand upside down to avoid touching the mirror that directs the light rays to the HOE screen. It was noticed that the implemented criteria for detecting when a user is grabbing an object (see section 6.2) tended to fail specially when the user tried to grab the objects with both index and thumb fingers in separate planes of the y-axis (see Fig. 7.12). This occlusion of the fingers made it challenging for the LM controller

to detect the user's gesture, and the user failed to grab the reconstructed box, which frustrated them and made them give a low score on *easiness* and *no-fatigue* categories. There were times in which the user achieved to grab the LF object and moved it, but the LM controller noisy readings provided data that indicated the user had dropped the LF object, which in consequence stopped moving. The user then had to retry to grab the object. This was specially the case when the user moved very fast, and some of them adjusted their speed in the next trials to try to complete the task and trying not to drop the LF objects.

The registered version appeared to be the least *painful* one, with users performing the task more easily. In the unregistered version, the users moved more slowly trying to match their movements to the movement of the LF. This made them invest a longer time on completing this task. In doing so, due to the absence of registration, the users moved their hands in excess, at times to a position outside the LM angle of view range, dropping the LF reconstruction once more.

Some comments were that, although the registered version felt more natural, sometimes the finger would obstruct the light that reconstructs the object, making it more difficult to perform the task. The failure in recognizing the grab gesture also made them hesitate about the depth at which the LF was reconstructed, since they located their hands at the perceived depth but the recognition of the gesture made them think that was not the right depth.

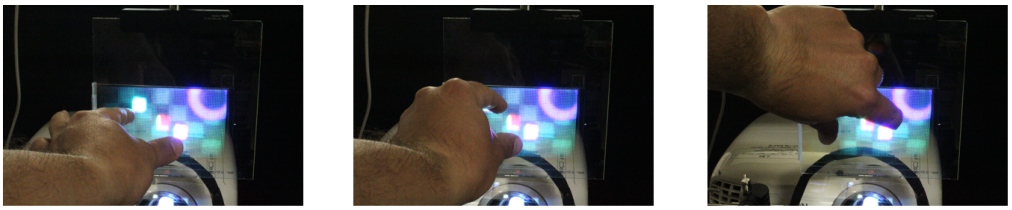
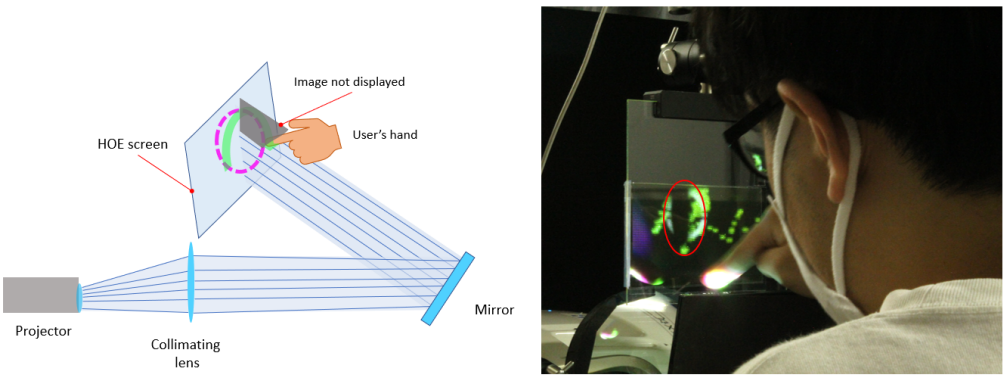


Figure 7.12: Illustration of the different ways the users employed to grab the reconstructed boxes. (a) and (c) were still detected by the LM controller, but the gesture (b) was more difficult to be correctly detected due to occlusion of the fingers.

7.3.3 Circle tracing task

This task, similar to the one involving the swipe gesture, presented less difference between the unregistered and the registered versions in the TCT, although still the registered versions showed a smaller TCT in average. The ANOVA test shows a value that falls short from being significant ($p = 0.1001 \approx 0.1$), indicating that the difference is still

not significant with this number of participants but the tendency seems to benefit the registered version. Since in this task all the movements are registered in the interaction space by marking the trajectory followed, the HOE screen ended up full of trajectories that were not intended to be the circle trajectory, what might probably decreased the performance of the participants. Some of them stated that the sensor was *too sensitive* to the movements of the finger. The users also noticed that moving the finger slowly was better for controlling the trajectory of the marked path. During the experiment, some participants mentioned that the registered version was more comfortable to use. Finally, some of them stated that placing the finger on top of the trajectory made it difficult to interact with the GUI, because then the user was not able to see what was happening (see Fig. 7.13). This observation was also made with the box sorting task, but interestingly it was not the case with the swiping task.



(a) (b)

Figure 7.13: Illustration showing when a user blocks the light coming from the projector with the finger. Depiction of the problem in the experimental setup (a) and during the evaluation test (b). The shadow of the finger is circled in red.

Chapter 8

Conclusions

In this final chapter, we summarize the performed work and discuss the pending tasks to achieve a 3D user interface that can provide a significant impact to the public. We reflect on the benefits and the drawbacks of our proposed approach, outline the future tasks and make some final remarks.

8.1 Summary

In this work, based on the previous work of our group where a LF display using HOEs was recorded and a basic GUI based on scattered color light was proposed [68], both hardware and software improvements were explored and implemented to increase the robustness of the interaction and generate content in real time to create a more interactive GUI. This technique was merged with a commercial 3D tracking sensor specialized on detecting the position of the different structures of the hand. Using this information, we proposed a system where both scattered light detection and hand spatial information were used to have an interface with registered gestures and LF generation in the position of the finger of the user. The effectiveness of registering the position of the user was evaluated by measuring the task-completion time (TCT) of a group of participants in three different tasks, finding an important significance for the reduction of the TCT in the task that required a relatively more complicated manipulation (i.e., grabbing and moving). The other tracking-based tasks, although subjectively evaluated by the users as easier to perform under a position-registered environment, did not show a significant TCT reduction. However, a favorable tendency towards the registered versions of the interface was statistically detected.

Software and hardware improvements to achieve an interactive GUI from the basic

demonstration reported in [68] to the here presented system are:

- Increase of the accuracy of the projector look-up table by taking separate shots of the projected binary patterns
- Real-time implementation of the undistortion function to locate the elementary images in the correct HOE positions
- Proposal of two different ways to have interactive LF reconstructions by either using the system's CPU or GPU, achieving moving LF images in real time
- Modification of the structure of the HOE array from a single multiplexed photopolymer sheet to three aligned RGB photopolymer layers to increase the diffraction efficiency of the screen
- Improvement of the finger detection and the color detection algorithms for them to be used in bright environments. The compensation of the color of the skin of the user to achieve a more robust signal was also proposed
- Proposal of a color-based algorithm to detect the movement and position of the finger of a user using a single RGB camera
- Demonstrations of several 3D-GUIs using scattered light detection (e.g., screen unlock, 3D object spinning, y-axis color based movement, sound volume, grab and spinning of one cube, grab of several cubes, ATM interface and 3D free drawing)
- Proposal of a method to register the 3D content reconstructed by the LF display with the position of the user by using scattered light detection and the LM sensor
- Development of methods to use the 3D tracking information to detect the gestures of a user.

In chapter 4, we proposed for the first time the use of color detection in a LF display and its combination with 2D tracking to detect the movement in 3D of a user interacting with the content. The use of color information makes it possible to use a single RGB camera to track in 3D the position of the user, and this idea can potentially be applied to other scenarios where the number of sensors needs to be reduced.

Exploiting the techniques developed in chapter 4, the proposed registration method using scattered light detection in chapter 5 provides, for the first time, a method to systematically match the position of the user and the reconstruction of a LF image. Methods proposed to this day to provide of interactive interfaces to LF displays have not taken in consideration this problem, which is important to address to create a functional

3D GUI. The development of 3D displays requires a method of this kind to improve the human computer interaction (HCI), as shown by the evaluation test performed in this study.

Finally, in the elementary techniques level, we have provided a new model to use RANSAC 2D and SVD to match the detected position, which is an alternative to the more unstable version of the RANSAC 3D version. By using LF images reconstructed at the same depth, a plane parallel to the HOE screen is estimated. The distance from this plane to the position of the user, along with the 2D version of the RANSAC estimation, proved to be a more practical alternative. This technique, combined with the detection of scattered light to provide the required data, is the most important contribution of this work to the development of a 3D touchable interface.

8.2 Future work

This topic is challenging since building an interface that gets adopted by the public requires overcoming not only the quality of present GUIs, but also a number of *cultural* or *educational* instances, because the public has already developed certain habits towards 2D interfaces and 2D GUIs. The quality of the images displayed by modern 2D GUIs can have up to 4K-pixels resolution or even higher sometimes, what represents a very high detail and image quality that makes the user feel comfortable. One of the biggest challenges of 3D interfaces is that, to this day, that level of detail cannot be reached for a 3D real image in real-time without using an extraordinary computational power.

In a world where internet and wireless communications are omnipresent, a successful 3D GUI would also require sending in real-time a good quality reconstruction of a 3D image, what is an even more burdensome process than displaying a 3D reconstruction accessible from on-site hardware. Since the transference of information can be realized more efficiently (by several orders of magnitude) in 2D GUIs with the current network bandwidth available, some technological breakthrough in the field of information processing is required before 3D GUIs are able to have the same wireless capabilities the current 2D GUIs have. This is why, in the author's opinion, it is currently too early to expect 3D GUIs and interactions very similar to real life to become widespread and replace *all* the current technologies based on 2D displays.

However, there are many applications using 3D display and sensing technology that can be achieved with the current computer power. Applications in 2D GUIs and interaction in 3D can become complementary, maybe by using hybrid interfaces that harness both

the benefits of 2D and 3D interfaces [56]. In that sense, it is important to explore different kinds of interactions to confirm in which scenarios the 3D GUI can bring a benefit to the public. That is the contribution of the present study, where by registering the content with the position of the user and performing some tasks, we have been able to confirm that content registration effectively decreases the time a user requires to finish a task, specially when gestures and displacement through the available LF space is needed. Registration allows for a more efficient use of the available detection space than having the LM interaction volume separated from the content.

To this day, there are several demonstrations of the new company that owns the LM technology, Ultraleap, mixing its interaction technology with the Looking Glass LF display [77]. It is noticeable how many applications are depicted where gestures are used, but they still rely on the paradigm of showing a cursor or the shape of a hand within the screen, and not registering the content (see Fig. 8.1). In that sense, the demonstrated interactions in this study and the effect of the registration can help to give more direction to the development of a really effective 3D GUI.

Figure 8.1: Implemented gesture using the LM controller and the Looking Glass LF display, where the user pinches a portrait and moves the nose of a person, implemented by UX designer Albert Hwang. Source: [77]. Notice how the gesture and the interaction are not registered.

Some drawbacks of the proposed GUI are that it requires a big space for the location of the projector and collimation optics to produce an approximately-flat wavefront. We also confirmed an important drawback during the user evaluation when users blocked the reconstruction light while interacting with the LF objects (see Fig. 7.13), what made this interface uncomfortable. The presence of the mirror at the bottom of the screen, although implemented for the users to have an easier access to the interface, also blocked the user from using the GUI freely. A better design, or even possibly a change of the passive optical element that creates a LF (the HOE array) can be considered to increase the easiness of use and preserve the transparency and potential of application in augmented reality.

Although some complaints from the users included a wrong identification of the per-

formed gestures and faulty tracking, these problems can probably be solved by improving the gesture recognition through artificial intelligence (AI) strategies, as well as using more updated versions of the LM commercial program which reportedly has a better accuracy while tracking a user (software compatibility problems held us back from using the latest controller version). Software development tasks are required to explore the capabilities of this interactive 3D GUI. Something similar can be said about the color detection, which could include a more in-line and dynamic approach to detect the different tones of skin of the user, the different illuminations of the environment sampled at regular intervals, and also AI algorithms to better sample the scattered light.

Finally, it should be said that we can propose and develop the design of several kinds of interaction (grabbing, deforming, spinning, throwing, etc.), implement them and refine them. However, what will be the purpose of implementing any given gesture if there will be no significant application that can be improved out of it? Collaboration with professionals of different areas (e.g., architects, biologists, chemists, designers, artists, etc.) that can potentially take advantage from 3D GUIs and 3D interaction should be actively pursued to find the right niches where this technology can be useful.

Development of 3D GUIs is the natural step from 2D interaction, which has proved to be very beneficial to HCI and has improved what people can do with computers. There is absolutely no reason to think 3D interaction will not be as beneficial for HCI as 2D interaction has been. However, several tasks should still be addressed before that happens.

Publication list

Publications related to this thesis

Journal papers

- Sánchez Salazar Chavarría, Iván A., Shimomura, K., Takeyama, S., & Yamaguchi, M. (2022). Interactive 3D touch and gesture capable holographic light field display with automatic registration between user and content. Journal of the Society of Information Display. (Accepted: August 15, 2022).

- Sánchez Salazar Chavarría, Iván A., Nakamura, T., & Yamaguchi, M. (2020). Interactive optical3D-touch user interface using a holographic light-field display and color information. Optics Express, 28(24), 36740-36755

International conferences

- Sánchez Salazar Chavarría, Iván A., Nakamura, T., & Yamaguchi, M. (2021). Automatic registration of gesture-sensor data and light-field for aerial 3D-touch interface. In Imaging and Applied Optics Congress (3Th7E-2). Optical Society of America.

- Sánchez Salazar Chavarría, Iván A.,& Yamaguchi, M. (2021b). Interactive and gesture-capable 3d holographic light field display with registered interaction between user and light volume. In Proceedings of the 28th International Display Workshops (IDW ' 21) (FMC3/INP6-3)

- Sánchez Salazar Chavarría, Iván A., Nakamura, T., & Yamaguchi, M. (2019a). An interactive holographic light-field display color-aided 3d-touch user interface. In Proceedings of the International Display Workshops (IDW '19), Sapporo, Japan (INP5-4).

Domestic conferences

- Sánchez Salazar Chavarría, Iván A., Shimomura, K., & Yamaguchi, M. (2021). Registration of gesture-sensor data and light field display based on colored scattered light detection. In Conference on 3D imaging 2021 (pp. 3-3).
- Sánchez Salazar Chavarría, Iván A.,& Yamaguchi, M. (2021a). Interactive 3D touch holographic light field display. In 15th Forum on New Imaging Systems and Photonic Information.
- Sánchez Salazar Chavarría, Iván A., Nakamura, T., & Yamaguchi, M. (2020). Interactive capabilities of a 3D holographic light-field display based on scattered light detection, In Conference on 3D imaging 2020 (pp. 10-3).
- Sánchez Salazar Chavarría, Iván A., Nakamura, T., & Yamaguchi, M. (2019b). Development of interactive real-time color-aided capabilities for 3d touch-detection in a holographic light field. In Proceedings of the ITE annual convention (22B-1). The Institute of Image Information and Television Engineers (ITE).

Other publications

- Shimomura, K., Sánchez Salazar Chavarría, Iván A.,Takeyama, S., & Yamaguchi, M. (2022). Integration of holographic light field display and 2d display for 3d-touch user interface. (p. 10). International Workshop on Holography and Related Technologies (IWH2022).
- Yamaguchi, M., Igarashi, S., Sánchez Salazar Chavarría, Iván A., Kakinuma, K., & Nakamura, T. (2019). Holography enables ultrahigh-reality light-field display and 3d-touch user interface. In Proceedings of the Annual Meeting of the Japanese Optical Society (Invited 2aBJ2). Optics and Photonics Japan (OPJ)

Bibliography

[1] Vamsi Kiran Adhikarla, Jaka Sodnik, Peter Szolgay, and Grega Jakus. Exploring direct 3d interaction for full horizontal parallax light field displays using leap motion controller. *Sensors*, 15(4):8642–8663, 2015.

[2] Sriya Adhya and John Noé. A complete ray-trace analysis of the mirage toy. *Proc. SPIE ETOP*, pages 966518–1, 2007.

[3] Ferran Argelaguet and Carlos Andujar. A survey of 3d object selection techniques for virtual environments. *Computers & Graphics*, 37(3):121–136, 2013.

[4] Benjamin Bach, Ronell Sicat, Johanna Beyer, Maxime Cordeil, and Hanspeter Pfister. The hologram in my hand: How effective is interactive exploration of 3d visualizations in immersive tangible augmented reality? *IEEE transactions on visualization and computer graphics*, 24(1):457–467, 2017.

[5] Daniel Bachmann, Frank Weichert, and Gerhard Rinkenauer. Review of three-dimensional human-computer interaction with focus on the leap motion controller. *Sensors*, 18(7):2194, 2018.

[6] Detlef Bahr, Knut Langhans, Martin Gerken, Carsten Vogt, Daniel Bezecny, and Dennis Homann. Felix: A volumetric 3d laser display. In *Projection Displays II*, volume 2650, pages 265–273. SPIE, 1996.

[7] Aryabrata Basu, Christian Saupe, Eric Refour, Andrew Raij, and Kyle Johnsen. Immersive 3dui on one dollar a day. In *2012 IEEE symposium on 3D user interfaces (3DUI)*, pages 97–100. IEEE, 2012.

[8] Stephen A Benton and V Michael Bove Jr. *Holographic imaging*. John Wiley & Sons, 2008.

[9] Alex Butler, Otmar Hilliges, Shahram Izadi, Steve Hodges, David Molyneaux, David Kim, and Danny Kong. Vermeer: direct interaction with a 360 viewable 3d display.

In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 569–576, 2011.

[10] DH Close. Holographic optical elements. *Optical Engineering*, 14(5):145408, 1975.

[11] International Electrotechnical Commission et al. 3d display devices-part 51-1: Generic introduction of aerial display. Technical report, Tech. Rep. IEC TR 62629-51-1: 2020, 2020.

[12] Grégoire Dupont de Dinechin and Alexis Paljic. From real to virtual: An image-based rendering toolkit to help bring the world around us into virtual reality. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, March 2020.

[13] Wendy Fang and Tony Chang. Calibration in touch-screen systems. *Texas Instruments Incorporated*, 10, 2007.

[14] David Fattal, Zhen Peng, Tho Tran, Sonny Vo, Marco Fiorentino, Jim Brug, and Raymond G Beausoleil. A multi-directional backlight for a wide-angle, glasses-free three-dimensional display. *Nature*, 495(7441):348–351, 2013.

[15] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[16] Walter Gander and Jiri Hrebicek. *Solving problems in scientific computing using Maple and Matlab®*. Springer Science & Business Media, 2011.

[17] Souvik Ghosh, Spandan Ghosh, Pradeep Kumar, Erik Scheme, and Partha Pratim Roy. A novel spatio-temporal siamese network for 3d signature recognition. *Pattern Recognition Letters*, 144:13–20, 2021.

[18] Tibor Guzsvinecz, Veronika Szucs, and Cecilia Sik-Lanyi. Suitability of the kinect sensor and leap motion controller—a literature review. *Sensors*, 19(5):1072, 2019.

[19] Parameswaran Hariharan. *Basics of holography*. Cambridge university press, 2002.

[20] R Hartley and A Zisserman. Multiple view geometry in computer vision (cambridge university, 2003). *Chapter 6*, 2.

[21] Otmar Hilliges, David Kim, Shahram Izadi, Malte Weiss, and Andrew Wilson. Holodesk: direct 3d interactions with a situated see-through display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2421–2430. ACM, 2012.

[22] Sora Hisaichi, Kiwamu Sumino, Kunihiro Ueda, Hidenori Kasebe, Tohru Yamashita, Takeshi Yuasa, Uwe Lippmann, Petra Aswendt, Roland Höfling, and Yoshihiro Watanabe. Depth-aware dynamic projection mapping using high-speed rgb and ir projectors. In *SIGGRAPH Asia 2021 Emerging Technologies*, pages 1–2. 2021.

[23] Keehoon Hong, Jiwoon Yeom, Changwon Jang, Jisoo Hong, and Byoungho Lee. Full-color lens-array holographic optical element for three-dimensional optical see-through augmented reality. *Optics letters*, 39(1):127–130, 2014.

[24] Timothy Hoye and Joseph Kozak. Touch screens: A pressing technology. In *Tenth Annual Freshman Engineering Sustainability in the New Millennium Conference April*, volume 10, 2010.

[25] Boaz Jessie Jackin. Digitally designed holographic optical elements for large-size light field displays. In *Ultra-High-Definition Imaging Systems III*, volume 11305, pages 38–44. SPIE, 2020.

[26] Boaz Jessie Jackin, Lode Jorissen, Ryutaro Oi, Jui Yi Wu, Koki Wakunami, Makoto Okui, Yasuyuki Ichihashi, Philippe Bekaert, Yi Pai Huang, and Kenji Yamamoto. Digitally designed holographic optical element for light field displays. *Optics Letters*, 43(15):3738–3741, 2018.

[27] Bahram Javidi, Artur Carnicer, Jun Arai, Toshiaki Fujii, Hong Hua, Hongen Liao, Manuel Martínez-Corral, Filiberto Pla, Adrian Stern, Laura Waller, et al. Roadmap on 3d integral imaging: sensing, processing, and display. *Optics Express*, 28(22):32266–32293, 2020.

[28] Tong Jia, DongYue Chen, LiQun Bai, ZhuoQun Fang, and HongYu Wang. Depth perception based on omnidirectional ring structured light. *Optics & Laser Technology*, 104:123–132, 2018.

[29] Chufan Jiang, Beatrice Lim, and Song Zhang. Three-dimensional shape measurement using a structured light system with dual projectors. *Applied optics*, 57(14):3983–3990, 2018.

[30] Andrew Jones, Ian McDowall, Hideshi Yamada, Mark Bolas, and Paul Debevec. Rendering for an interactive 360 light field display. In *ACM SIGGRAPH 2007 papers*, pages 40–es. 2007.

[31] Lode Jorissen, Boaz Jessie Jackin, Ryutaro Oi, Koki Wakunami, Makoto Okui, Yasuyuki Ichihashi, Gauthier Lafruit, Kenji Yamamoto, and Philippe Bekaert. Ho-

mography based identification for automatic and robust calibration of projection integral imaging displays. *Applied optics*, 58(4):1200–1209, 2019.

[32] Lode Jorissen, Ryutaro Oi, Koki Wakunami, Yasuyuki Ichihashi, Gauthier Lafruit, Kenji Yamamoto, Philippe Bekaert, and Boaz Jessie Jackin. Holographic micromirror array with diffuse areas for accurate calibration of 3d light-field display. *Applied Sciences*, 10(20):7188, 2020.

[33] Nam Kim, Yan-Ling Piao, and Hui-Ying Wu. Holographic optical elements and application. *Holographic Materials and Optical Systems*, 5:99–131, 2017.

[34] James L Knight. Manual control and tracking. *Handbook of human factors*, pages 182–218, 1987.

[35] Gabriele Lanaro. *Python High Performance*. Packt Publishing Ltd, 2017.

[36] Joseph J LaViola Jr, Ernst Kruijff, Ryan P McMahan, Doug Bowman, and Ivan P Poupyrev. *3D user interfaces: theory and practice*. Addison-Wesley Professional, 2017.

[37] George Lawton. 3d displays without glasses: coming to a screen near you. *Computer*, 44(01):17–19, 2011.

[38] Gang Li, Dukho Lee, Youngmo Jeong, Jaebum Cho, and Byoungho Lee. Holographic display for see-through augmented reality using mirror-lens holographic optical element. *Optics letters*, 41(11):2486–2489, 2016.

[39] Bo Liao, Jing Li, Zhaojie Ju, and Gaoxiang Ouyang. Hand gesture recognition with generalized hough transform and dc-cnn using realsense. In *2018 Eighth International Conference on Information Science and Technology (ICIST)*, pages 84–90. IEEE, 2018.

[40] Gabriel Lippmann. Epreuves réversibles photographies integrals. *Comptes-Rendus Academie des Sciences*, 146:446–451, 1908.

[41] Duo Lu and Dijiang Huang. Fmcode: A 3d in-the-air finger motion based user login framework for gesture interface. *arXiv preprint arXiv:1808.00130*, 2018.

[42] Yujun Lu, BoYu Gao, Jinyi Long, and Jian Weng. Hand motion with eyes-free interaction for authentication in virtual reality. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 714–715. IEEE, 2020.

[43] Xiao-Ming Ma, Yan Xing, Jia-Chen Zheng, Xiao-Wei Li, and Qiong-Hua Wang. A real-time interactive rendering method for 360° tabletop integral imaging 3d display. *Journal of the Society for Information Display*, 29(9):679–688, 2021.

[44] Bryan FJ Manly and Jorge A Navarro Alberto. *Multivariate statistical methods: a primer*. Chemical Rubber Company (CRC) Taylor and Francis Group, 2016.

[45] Atsushi Matsubayashi, Yasutoshi Makino, and Hiroyuki Shinoda. Direct finger manipulation of 3d object image with ultrasound haptic feedback. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–11. ACM, 2019.

[46] Tomoya Nakamura and Masahiro Yamaguchi. Rapid calibration of a projection-type holographic light-field display using hierarchically upconverted binary sinusoidal patterns. *Applied Optics*, 56(34):9520–9525, 2017.

[47] Isamu Nakao, Shunsuke Takahashi, Hiroshi Kobayashi, and Masahiro Yamaguchi. A psychometric evaluation of the user interaction system combining an aerial display with gesture recognition. *ITE Transactions on Media Technology and Applications*, 6(3):217–225, 2018.

[48] Kai Nickel and Rainer Stiefelhagen. Visual recognition of pointing gestures for human–robot interaction. *Image and vision computing*, 25(12):1875–1884, 2007.

[49] Soon-gi Park, Jiwoon Yeom, Youngmo Jeong, Ni Chen, Jong-Young Hong, and Byoungho Lee. Recent issues on integral imaging and its applications. *Journal of Information Display*, 15(1):37–46, 2014.

[50] Zhou Ren, Junsong Yuan, Jingjing Meng, and Zhengyou Zhang. Robust part-based hand gesture recognition using kinect sensor. *IEEE transactions on multimedia*, 15(5):1110–1120, 2013.

[51] David E Roberts. History of lenticular and related autostereoscopic methods. *Leap Technologies. Hillsboro*, 16, 2003.

[52] Soma Sakurai, Tomoya Nakamura, and Masahiro Yamaguchi. The use of color in scattered light for 3d touchable holographic light-field display. In *JSAP-OSA Joint Symposia*, page 13a_C301_4. Optical Society of America, 2016.

[53] Iván A. Sánchez Salazar Chavarría, Tomoya Nakamura, and Masahiro Yamaguchi. Interactive optical 3d-touch user interface using a holographic light-field display and color information. *Optics Express*, 28(24):36740–36755, 2020.

[54] Xinzhu Sang, Xin Gao, Xunbo Yu, Shujun Xing, Yuanhang Li, and Yongle Wu. Interactive floating full-parallax digital three-dimensional light-field display based on wavefront recomposing. *Optics express*, 26(7):8883–8889, 2018.

[55] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. Kinect range sensing: Structured-light versus time-of-flight kinect. *Computer vision and image understanding*, 139:1–20, 2015.

[56] Kyoka Shimomura, Alexis Sánchez, Saori Takeyama, and Masahiro Yamaguchi. Integration of holographic light field display and 2d display for 3d-touch user interface. In *International Workshop on Holography and Related Technologies (IWH2021)*, page 10. Optical Society of Japan, 2021.

[57] Ken Sugimori, Hironori Mitake, Hirohito Sato, Kensho Oguri, and Shoichi Hasegawa. Avatar tracking control with generations of physically natural responses on contact to reduce performers' loads. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pages 1–5, 2021.

[58] Phil Surman. Stereoscopic and autostereoscopic displays. In *3D-TV System with Depth-Image-Based Rendering*, pages 375–411. Springer, 2013.

[59] Richard Szliski. *Computer Vision Algorithms and Applications*. Ithaca, Springer, 2011.

[60] Yasuhiro Takaki, Yohei Urano, Shinji Kashiwada, Hiroshi Ando, and Koji Nakamura. Super multi-view windshield display for long-distance image information presentation. *Optics express*, 19(2):704–716, 2011.

[61] Mikito Takenaka, Takashi Kakue, Tomoyoshi Shimobaba, and Tomoyoshi Ito. Interactive holographic display for real-time drawing and erasing of 3d point-cloud images with a fingertip. *IEEE Access*, 9:36766–36774, 2021.

[62] Alan C Traub. Stereoscopic display using rapid varifocal mirror oscillations. *Applied Optics*, 6(6):1085–1087, 1967.

[63] Sam Van der Jeught and Joris JJ Dirckx. Real-time structured light profilometry: a review. *Optics and Lasers in Engineering*, 87:18–31, 2016.

[64] Koki Wakunami, Po-Yuan Hsieh, Ryutaro Oi, Takanori Senoh, Hisayuki Sasaki, Yasuyuki Ichihashi, Makoto Okui, Yi-Pai Huang, and Kenji Yamamoto. Projection-type see-through holographic three-dimensional display. *Nature communications*, 7(1):1–7, 2016.

[65] Fumiaki Watanabe, Tomoya Nakamura, Shiho Torashima, Shunsuke Igarashi, Shinji Kimura, Yuji Aburakawa, and Masahiro Yamaguchi. Dispersion compensation for full-color virtual-imaging systems with a holographic off-axis mirror. In *Practical Holography XXXIV: Displays, Materials, and Applications*, volume 11306, page 1130604. International Society for Optics and Photonics, 2020.

[66] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393, 2013.

[67] Chadwick A Wingrave and Joseph J LaViola. Reflecting on the design and implementation issues of virtual environments. *Presence*, 19(2):179–195, 2010.

[68] Masahiro Yamaguchi and Ryo Higashida. 3d touchable holographic light-field display. *Applied optics*, 55(3):A178–A183, 2016.

[69] Masahiro Yamaguchi, Nagaaki Ohyama, and Toshio Honda. Holographic three-dimensional printer: new method. *Applied Optics*, 31(2):217–222, 1992.

[70] Hirotsugu Yamamoto, Masahiko Yasui, M Sakti Alvissalim, Masashi Takahashi, Yuka Tomiyama, Shiro Suyama, and Masatoshi Ishikawa. Floating display screen formed by airr (aerial imaging by retro-reflection) for interaction in 3d space. In *2014 International Conference on 3D Imaging (IC3D)*, pages 1–5. IEEE, 2014.

[71] Hirotsugu Yamamoto, Masahiko Yasui, M Sakti Alvissalim, Masashi Takahashi, Yuka Tomiyama, Shiro Suyama, and Masatoshi Ishikawa. Floating display screen formed by airr (aerial imaging by retro-reflection) for interaction in 3d space. In *2014 International Conference on 3D Imaging (IC3D)*, pages 1–5. IEEE, 2014.

[72] Masahiko Yasui, Yoshihiro Watanabe, and Masatoshi Ishikawa. Occlusion-robust sensing method by using the light-field of a 3d display system toward interaction with a 3d image. *Applied optics*, 58(5):A209–A227, 2019.

[73] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[74] Pengcheng Zhou, Yan Li, Shuxin Liu, and Yikai Su. Compact design for optical-see-through holographic displays employing holographic optical elements. *Optics Express*, 26(18):22866–22876, 2018.

[75] Zoran Zivkovic and Ferdinand Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.

[76] Aska 3d display–operation principle and structure. https://aska3d.com/en/technology.php.

[77] Ultraleap - through the looking glass with hand tracking. https://www.ultraleap.com/company/news/case-study/looking-glass-hand-tracking-interactions/ (Retrieved July 11, 2022).

[78] 4ukey for android: Unlock your phone without pin, pattern, or password. https://www.sourceht.com/4ukey-for-android-unlock-your-phone-without-pin-pattern-or-password/.

[79] Looking glass factory. https://lookingglassfactory.com/tech.

[80] Point grey official website (accessed may 25, 2022):. https://www.flir.com/products/bumblebee2-firewire/.

[81] Ultraleap official website (accessed may 25,2022). https://www.ultraleap.com/.

[82] Blender. http://www.blender.org/.

[83] Python software foundation, numba official website:. https://pypi.org/project/numba/.

[84] Allvm research project official website:. https://publish.illinois.edu/allvm-project/.

[85] Sony reality display - elf-sr1. https://www.sony.jp/spatial-reality-display/products/ELF-SR1/ (Retrieved June 16, 2022).

[86] Ultraleap - developer website: Introducing the skeletal tracking model. https://developer-archive.leapmotion.com/documentation/objc/devguide/Intro_Skeleton_API.html (Retrieved June 28, 2022).