Surface Gesture & Object Tracking on Tabletop Devices

Yannick Verdié

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Francis Quek, Chair
Roger Ehrich
Yong Cao

30 April 2010
Blacksburg, Virginia

Keywords: Human Computer Interaction, Computer Vision, Object Tracking, Tabletop Surface

Surface Gesture & Object Tracking on Tabletop Devices

Yannick Verdié

(ABSTRACT)

In this thesis, we are interested in the use of tabletop surfaces for interactive manipulations. We focus on the implementation of Image Processing algorithms and techniques in two projects exploiting a horizontal surface: 'Tangram Project' and 'MirrorTrack'.

The 'Tangram Project' studies children's mathematical skills when manipulating geometrical shapes. This project is supported by NFS (NSF 0736151) based on the proposal *'Social Organization, Learning Technologies & Discourse: System Features for Facilitating Mathematical Reasoning in PreK-3 Students'* by M. Evans, F. Quek, R. Ehrich and J. Wilkins. Our contribution is the design and realization of visio-based tracking software that could be used in a classroom. Our implementation offers three modes of interaction making it easier to study the children's behaviors in specific situations and constraints.

The 'MirrorTrack Project' is an idea described in previous research [15, 14] using a horizontal surface with two side-mounted cameras to track fingertips. Our contribution to the 'MirrorTrack Project' is the design and realization of video-based interaction software. 'MirrorTrack Project' provides an improvement to one of the Tangram modes (the Virtual mode) by providing real 3D fingertip location above the surface. Among other functionalities, it provides hovering and touch detection [73].

We conclude by describing the possibility of merging those two systems and by highlighting the benefits of such a fusion. Integrating 'MirrorTrack' with the 'Tangram project' provides even more interaction opportunities for the children.

Acknowledgments

I would like to thank my advisor Dr. Quek for his help. Je remercie Alice sans qui rien de tout cela n'aurait été possible.

This research has been partially supported by NSF grants "Embodied Communication: Vivid Interaction with History and Literature", IIS-0624701, "Interacting with the Embodied Mind", CRI-0551610, and Embodiment Awareness, Mathematics Discourse and the Blind, NSF-IIS- 0451843.

Contents

List of Figures

List of Tables

We are all faced with a series of great opportunities brilliantly disguised as impossible situations.

-Charles R. Swindoll

Chapter 1

Introduction

A picture is worth a thousand words. An interface is worth a thousand pictures.

Ben Shneiderman

1.1 The Goal

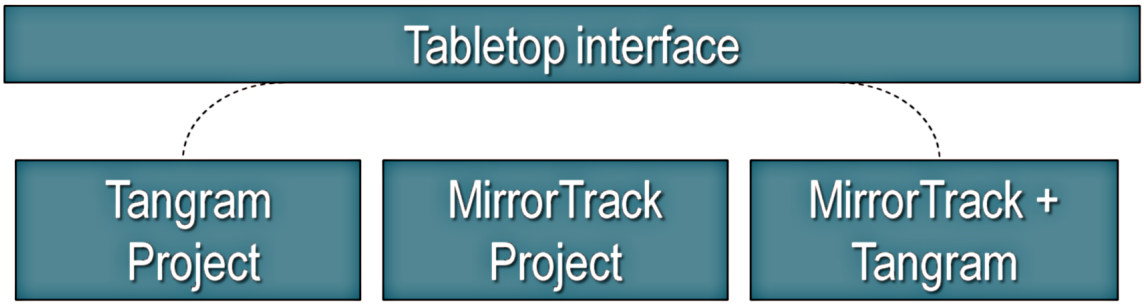


Figure 1.1: One problem, three solutions

This thesis is focused on the challenge of creating a better interface between humans and computers by using gesture recognition and objects tracking rather than the traditional mouse and keyboard.

This area of research has been investigated by the HCI community for decades without any unanimously accepted solutions. We investigated a solution based on a tabletop surface and Image Processing algorithms. Indeed, tabletop surfaces offer endless possibilities to manipulate tangible objects as a Tangible User Interface [35] as well as a means of simultaneous interaction with the computer via the hands. Conferences have been dedicated to exploring the possible applications of tabletop surfaces and many solutions have been proposed [57, 29, 63, 53]. A tabletop surface is a horizontal display where users can gather around and interact with the system by pointing at the surface. Tabletops aim at providing users with multi-touch capabilities and a collaborative workspace. However, technical difficulties lead to the use of dedicated devices to allow for the detection of users' hand and objects on the tabletop surface. For example, researchers have employed devices such as the capacitive and resistive touch detectors used in ATMs, cellphones or [57], back-projectors [17, 71], Infra Red cameras [56], electro-magnetic sensors, and so on. However, all those solutions have their own drawbacks and bring additional costs to the system. In this thesis, we study the possibilities of using vision-based tabletop design with low-cost cameras and a common LCD screen embedded in the table. For more clarity, we can define our problem as an objective function.

> ***Our objective function is to minimize the cost of our design while maximizing interaction possibilities (by offering interactions not possible in other setup) and minimizing the intrinsic drawbacks of a vision-based system.***

1.2 The Setup

To provide a new interface for tabletop surfaces, we decided to investigate two options with a same goal: an improved interaction. One is based on 2D finger and object tracking, the other on 3D finger and fingers'reflexion tracking. Both bear their own advantages and drawbacks, and we showed that these systems can be finally merged together to constitute a more reliable system.

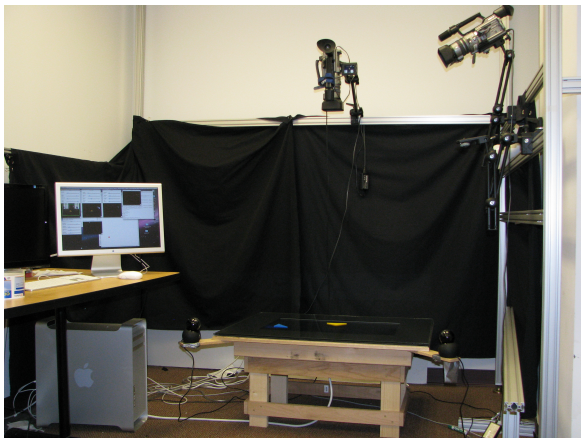


Figure 1.2: Tabletop ‘tangram’ setup, one camera above the screen

The first system, called the ‘Tangram Project’, is focused on tangible interactions and consist of a horizontal surface and by one downward-looking camera (vision-sensor). We decided to build a table around a flat 30” LCD screen and to place a protective glass sheet over it. This protective glass eases the tangible objects’ manipulation on the screen. Video capture can be accomplished with any kind of camera, from basic webcams to more sophisticated cameras. The camera is fixed above the screen in order for the rectangular surface space to fit directly into the camera image, thus, the camera axis normal to the surface. The camera sees tangible objects called ‘tangrams’ [64] and markers placed on the surface as well as the users’ hands. This system will detect the tangrams, the markers and the users’ fingertips during manipulation through its single camera (see Figure 1.2).

Figure 1.3: Tabletop ‘MirrorTrack’ setup, two side-mounted cameras next to the screen

The second system developed in parallel, called 'MirrorTrack', uses the same tabletop surface with the protective glass sheet above it, but with two side-mounted cameras instead of a single overhead camera (see Figure 1.3). Two basic webcams, used as cameras, are placed at a very low grazing angle in order to exploit the reflective behavior of the glass from this viewpoint and to gather additional information (see Section 5.1.3).

Both options have the same goal, facilitation interactions with the computer that are perceived as more natural by users. We developed those two options in parallel to give the opportunity later on to evaluate which one is best received by users. Eventually, we realized that actually merging the two options had a lot of potential.

1.3 The Challenge

Our objective function is made up of three parts, each of them containing their own challenges and constraints.

1.3.1 Minimizing the cost:

To minimize the costs of our system for the two options, we decided to build a table using commonly used devices, such as an LCD screen and a basic webcam. The final product has to be affordable for a large range of users. One can see in Table 1.1 the advantages of our design. The advantage of our design is due to the extensive usage of general Image Processing algorithms rather than dedicated devices for object tracking and detection. Indeed, we decided to develop the software that was not limited by the hardware.

Nonetheless, minimizing the cost by choosing to rely on a software-based implementation raises other challenges. In order to ensure smooth interactivity between the user and the computer, the algorithms used in our system needed to be carefully selected in order to support real time processing of visual and tactile stimuli. In other words, real-time interaction is another consideration that leads us to prefer CPU-inexpensive algorithms and state-of-the-art approaches rather than more

commonly used and CPU-demanding algorithms.

1.3.2 Minimizing intrinsic vision-based drawbacks:

We want to offer possibilities for the users to manipulate tangible objects and to interact with the computer. These two requirements are both related to object detection and tracking. However, object detection and tracking with a vision-based system may suffer from occlusion problems. The results of such a set-up are strongly correlated with the camera position. For example, if the camera is situated at low glancing angle on the side, users tend to cross less often the Field Of View (*FOV*) of the camera.

Unfortunately, vision set-ups carry a major drawback: during manipulations on the tabletop surface, the user hand may come between the overhead camera and the object, creating a partial or complete occlusion of the tangible objects. This drawback can be minimized by (1) judiciously positioning the camera, (2) by using history information, or (3) by using any remaining information available during a partial object's occlusion to estimate the location of the object and reconstruct it accordingly. In our system, we decided to tackle the problem by implementing all three approaches just mentioned for our two options. In the Tangram project, we used an overhead camera and exploited available information during occlusion to estimate and reconstruct the position. We used historical information as well as local information; such as the relative position of the available part of the object compared with the position of the occluded part. These techniques allow the system to handle the occlusion issue as described in Chapter 4. In the MirrorTrack project, we use two side-mounted cameras to minimize the risk of occlusion. Moreover, historical and local information are also used to make the system more reliable (see Section 1.2 for set-up description and Section 5 for more detail on the project).

Table 1.1: Comparative table with commercialized tabletop solutions

Products	**Screen Characteristics**	**Sensor Characteristics**	**Advantages**	**Drawbacks**
Diamond Touch	Top-Down Projector Active area 64 x 48 cm or 86 x 65 cm	arrays of antennas	• Touch resolution ($<$ 1 pixel) • Transparent integration on Windows • Up to four distinct users • Robustness of an Hardware-based solution	• Expensive • Contrast decreases with high lighting • Limited to four users • Only detects touch (2D plane detection) • Shadows during manipulation
Microsoft Surface	Back-projected clear acrylic screen, Volume 22 x 21 x 42 cm.	five IR cameras	• Constant contrast for display • No users limitation (max 52 Touches) • Transparent integration on Windows • Robustness of an Hardware-based solution	• No users distinction • Expensive • Cumbersome • Only detects Touch (2D plane detection)
Smart Table	Back-Projector Active Area 57.2 x 42.9 cm	IR camera	• No users limitation (max 40 Touches) • For children education • Robustness of an Hardware-based solution	• No users distinction • Cumbersome • Only detects Touch (2D plane detection)
Tangram Alone	LCD screen	Top-Down Camera	• Inexpensive • Versatile (flexible) • Readily available • Possibility of Augmented Reality and Virtuality	• fingertips location limited in 2D plane • Software based solution (may be less robust than a Hardware based solution) • Occlusion problem
Tangram with MirrorTrack	LCD screen	Top-Down and 2 sided-mounted Cameras	• Inexpensive • Versatile (flexible) • Readily available • Possibility of Augmented Reality and Virtuality • full 3D hand location (more gesture recognitions possible)	• Software based solution (may be less robust than a Hardware based solution)

1.3.3 Increasing interaction possibilities:

In this section, we discuss some new interaction possibilities offered by our tabletop surface system. We will introduce two new Human-Computer interfaces that our tabletop system can provide: one is based on the users' hand motions and the other is based on objects laying on the tabletop surface.

1. Hand interaction and gesture recognition:

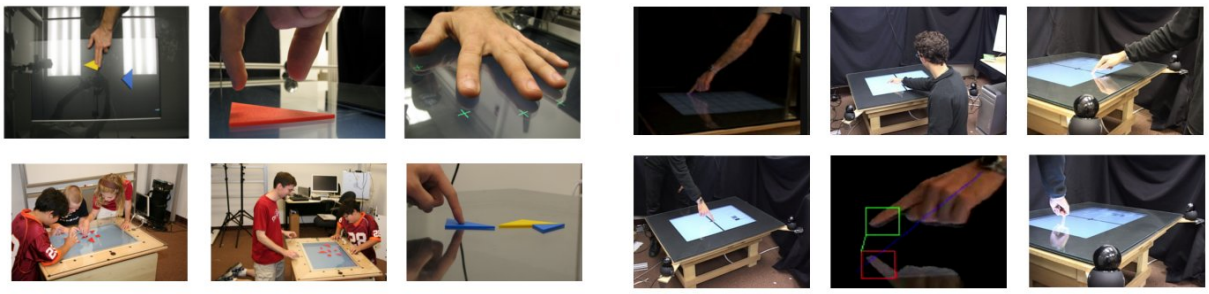
   Tabletop surfaces are usually designed for collaborative interactions by tracking users' fingertips touching the surface. When based on hardware solution, it is generally possible to detect the fingertips only when they touch the table. Even if we are not taking into account that hardware solutions for tabletops are expensive and cumbersome, their capacity to be aware of the user only when a 'touch' occurs make them greatly limited and restricted. For example, gesture recognition and hovering are not possible. Hovering is the action of moving the mouse above a target without selecting it. Nowadays, operating systems such as Windows 7 or MacOS X react to this action by changing the cursor or by highlighting the target. However, this technique is very difficult to implement on an overhead tabletop design because the fingertips' presence above the surface cannot be distinguished from fingertips touching the surface. A. Esenther et al. [22] described techniques to simulate a hovering action on such a tabletop surface. Nonetheless, it is not possible to obtain as intuitive results as during a real mouse manipulation. To solve this problem and make gesture recognition and hovering possible, we decided to use the two side-mounted cameras of MirrorTrack to compute fingertips's 3D location above the screen. This way, the complete 3D position of users' fingertips is available to the system. This approach of using stereo cameras for fingertip detection was also studied previously by Wren et al [79], TouchLight [77] and the Visual Touchpad [49].

2. Object recognition and manipulation:

   Tabletop surfaces are usually very restricted by their technology and cannot changed quickly.

However, a camera-based system is very flexible in that it allows for quick and versatile adaptation to any objects and hands. For example, camera-based systems such as Tangram and MirrorTrack [73, 14, 15] allow for object recognition and tracking on the screen, whereas hardware-based systems, such as Diamond Touch [19], can only detect users' fingertips and do not allow for any physical object tracking. This advantage offers Tangram and MirrorTrack new interaction possibilities between the users' hands and the system (gesture recognition), between objects and the system, or among all three. In our system, objects can be both Tangrams (plastic shapes of interest) and Markers (interaction tools).

1.3.4 Generals difficulties related to camera-based system:



(a) Tangram Project (b) MirrorTrack Project

Figure 1.4: Mosaic of both Projects(a), (b): Notice the shapes diversity and the occlusion challenge during manipulation

The foremost difficulty to build a good object detection and tracking system is driven by various factors such as:

- Objects' projective image varies with respect to the camera position and its relative orientation. In other words, the 2D image of an object changes constantly.

- Background images and reflections can be complex and confuse the object recognition system.

- The color and texture of the object depend on the light condition and the sensors' characteristics. Howerer, techniques are available to minimize the effects of such variations [11, 82, 50].
- The object classes can include very different shapes and colors (i.e each user hand is unique)
- Occlusions occur frequently when only a partial part of the object is detected by the system.

1.4 Outline

Now that we have defined the bases of our problem, we will structure the thesis the following way:

- Chapter 2 gives an overview of the problems and solutions existing in the domain of tabletop surfaces and hand tracking.
- Chapter 3 explains our contribution in this field and introduces the further chapters describing our solutions.
- Chapter 4 describes the methodologies and algorithms studied and selected in the Tangram project.
- Chapter 5 highlights the improvements brought by MirrorTrack compared to Tangram and further describe the problem: "How to detect the best reflection without compromising the results".
- Chapter 6 discusses the opportunities to merge these two projects and offers suggestions for the possible improvements of both systems.
- Chapter 7 briefly describes the technical details of Tangram and MirrorTrack.

Chapter 2

Previous contributions

> No problem can be solved from the same level of consciousness that created it.
>
> Albert Einstein

To solve the problems discussed above in Section 1.3, various approaches have been adopted. Since our two options are based on different designs, we will present previous work related to each design separately. No one has developed a work gathering those two options yet.

2.1 Tangram Tabletop

2.1.1 Tangram Tabletop - The beginning:

Multi-touch user interfaces have garnered a lot of interest from both educational and HCI researchers. Many studies and conferences explored those technologies [57, 29, 63, 53] because manipulations of objects and direct interactions with computers are perceived as more natural ways to communicate with devices. Initially, research such as that conducted by Lawrence D. Cutler et al. [17] or Brygg Ullmer et al. [71] in 1997 has been conducted using rear-projected tabletop dis-

plays as a solution for tabletop surfaces. This technique is cumbersome and is strongly restricted in terms of interaction. Only objects in contact with the surface are detected by the system. Later on, video projected solutions were adopted by Patten, James et al. [54], Stacey D. Scott et al. [63] or Shen Chia et al. [65] for example, but the occlusion problem appeared to be difficult to solve and they do not do depth measurement.

2.1.2 Tangram Tabletop - Reference solutions:

This area is very promising and a lot of solutions are currently available on the market. Diamond Touch [19] uses arrays of antennas inside the tabletop surface. Each user has a receiver typically placed under the user's chair. When a user touches an antenna on the table surface, the circuit between the antenna and the receiver is closed. The system can then successfully detect muti-touch and multi-user touch. However this system is very expensive and technically restricted (i.e hovering is not directly possible).The Lumisight Table [40] uses Lumisty film and mirrors to back-project the different user images on the surface. However, this solution is quite voluminous (as every back-projected solution) and expensive. Microsoft Surface [75] is a 30-inch reflective surface with a back-projector and four Infra-Red cameras. This system has the same drawbacks as Lumisight. The SMARTable [67] uses its SMART's patented DViT technology with a back-projected camera. Finally, the SenseTable [54] uses a combination of three or more sensing elements called Zowie (based on Wacom-tablet system) to detect fingertips and objects laying on the table and a video-projector on top of the table.

It is interesting to note that almost all these solutions are based on a back-projected setting and/or sensing devices. It seems that those solutions are not targeting the home-design system market but rather the research and academic market. However, even if those solutions are expensive and cumbersome, their main advantage is their reliability. Indeed, by creating a system detecting hands and objects relying on hardware, most of the problems discussed above are solved, at the price of reduced flexibility.

However, those solutions based on back-projection are becoming obsolete as flat panel displays become the norm. Finally, those projected-based systems are less attractive because of their size, cost, and robustness (need for calibration & mirrors).

2.1.3 Tangram Tabletop - Other developments:

A lot of other solutions were studied. We cannot provide an exhaustive list, but the most important solutions are listed here.

InteracTable and its i-Land project [68] developed in 1999 uses a touch-sensitive interactive display with bottom-up projection. PitABoard [13] uses a 8-by-8 sensor grid, which creates 15 distinct areas on the screen. InfoTable [58] uses two top-down cameras (one with very high resolution and small FOV) to detect objects markers thanks to a barcode attached to each device laying on the table. PDH Table [65] is an application for a circular tabletop surface, allowing personal data management, annotation, and organization. Illusion Hole [42] uses a mask on top of the tabletop surface with a hole in the middle to simulate stereoscopic images. This gives the illusion of 3D objects displayed on the surface (each user from each side can see a different view of an object).

2.2 MirrorTrack - Hand Tracking:

For several years, researchers have been trying to implement the best solutions to make direct Human-Computer interaction possible. Hand tracking techniques evolved a lot during the last years from one camera solutions to 3D solutions, offering users improved experiences. Indeed, the use of the hand as an input device instead of a mouse is an attractive idea. The huge success of embedded touch devices such as Apple's iPhone demonstrates that this complex area of research still has potential. However, even if a lot of research has already been done, better solutions are still to be found.

2.2.1 MirrorTrack - The choice of the sensor can make the difference:

One solution is to select the good capture device to detect the hand. Indeed, a rotated polarizing filter on a camera can create the illusion of a screen switched off due to the polarization of the LCD screen's light. Contrary to the LCD screen's light, the light reflected by the hand is unpolarized and thus detected by the camera. A. Agarwal et al. [1] used this process to directly disable the content of the screen to obtain the hands' shape. This is very easy to implement and does not require a lot of computer resources. However, it requires the use of an external component: the filter on the camera. This requirement is quite restrictive because it is not easy to find a filter that fits the camera. Moreover, it bears additional cost.

An Infra-Red camera *(IR)* can also be used as described in [56, 60]. Q. Wu et al. [80] uses these techniques combined with a graph cut algorithm. A.D. Wilson [77] and H. Benko et al. [10] combine a back-projected display with *IR* cameras to detect the touch on the screen. S. Izadi et al. [36] install on an LCD back screen a network of *IR* sensors (detector + emitter) detecting the proximity of fingertips. Similar to the polarized filters, adding an external component reduces the computer resource usage but brings external cost and complexity. Not all cameras provide *IR* system or support filter add-ons. Moreover, *IR* cameras are not able to process objects adequately and are only useful for hand interaction. This is one of the main limitations of the SMARTable [67].

2.2.2 MirrorTrack - The way the color is processed can make the difference:

The goal of the color segmentation solutions is to detect the hand based on the specific color of the skin. X. Zhu et al. [87] use a dataset of hand pictures created during a learning period. A hand segmentation by Bayes decision rules is used to define for each pixel whether or not it is a skin pixel.

S. Malik [48] uses this same approach and exploits the dataset for a histogram-based skin classifier described in [37]. A lot of techniques and derivatives aim at classifying a pixel as a skin-pixel or a non-skin pixel based on a non parametric approach ([18], [82] and [5]). The chosen color space for

this process is also important. HSV color representation seems to be the best choice [84] because it avoids the influence of shadow and lighting variation by discarding the V channel. Moreover, while working in this chromatic color space (H,S), Q. Huynh-Thu et al [33] show that the distribution of skin-pixel samples of 253 persons from different ethnicities, e.g. Caucasian, Asian, African, and so on, could be roughly drawn as a Gaussian mixture model built with four Gaussian bells. This knowledge could be exploited to detect skin-color without a learning period. The result is doubtless improved. Unfortunately, those processes based on skin-color have some drawbacks. The background could have similitudes in terms of color, resulting in a non-segmentation of the hand and the background. Another problem is the influence of the external light color on the skin. HSV space can minimize these kinds of issues, but only if the light condition hardly affects the hue and saturation.

2.2.3 MirrorTrack - Segmenting the foreground from the background:

Frame differencing is an Image Processing (*IP*) operation aiming at detecting a new event in an already-known situation. This technique is very popular in the *IP* community because of its simplicity and low CPU-cost for correct results.

Usually, a reference image is defined as the background. S. Malik et al. [48, 49] or A. D. Wilson [78] use a simple technique by grabbing just one reference frame during the beginning of the algorithm assuming that nothing will be in front of the camera at that moment. But if this condition is violated, the correctness of this method will be jeopardized. This simple process is still popular in the case of static background. However, in the case of dynamic background, this simple approach will fail. To tackle this issue, C. Von Hardenberg et al. [74] use a smart reference update runtime that dynamically adjusts the reference frame during processing. P. Kaewtrakulpong et al. [39] model the background as a gaussian mixture (GMM), but this technique requires a lot of computation because each pixel is parametrized by three gaussians.

T. Horprasert et al. [32] use a statistical approach by storing four tuples per pixel, the expected

color value, the standard deviation of the color value, the variation of the brightness distortion, and the variation of the chromatic values making the system more robust. In 2004-2005, Yungnam Kim et al. [41] proposed a new algorithm storing not a reference per frame as shown previously, but a reference per pixel (introduced as a 'box' in their paper). This algorithm is very effective but uses substantial resources.

The main problem with frame differencing is the difficulty of updating the background intelligently and frequently without including the foreground inside. As explained earlier, C. Von Hardenberg et al. [74] and K. Kim et al. [41] found solutions to this problem, but it is still not completely satisfying because each solution brings other issues (difficulty to tailor the code to each case).

2.2.4 MirrorTrack - Atypical techniques:

The previous sections gave an overview of the three main categories of techniques used in computer vision. However, this list does not aim at being exhaustive, but illustrates the main techniques used. Various research works tried different and unique methods to reach the same goal, and some of them are listed bellow.

Martin et al. [51] proposed an appearance-based approach of hand motion recognition. The appearance manifolds for six different hand gestures are constructed and then recognized. In order to do so, hand posture is classified using principal component analysis (PCA) of the distribution of hand images. Lathuiliere et al. [43] suggested a real-time visual hand tracking and posture estimation system with a single camera, which detects color clues on the hand glove. El-Sawah et al. [21] proposed a system for 3D hand tracking and dynamic gesture recognition using a single camera and [45] used a graph cut algorithm.

Finally, Imagawa, K et al. [34] uses a color based segmentation and a Kalman filter for gesture recognition and Wu et al. [81] use a theoretical framework for hand tracking.

Chapter 3

Overview of the projects

> The idea is to try to give all the information to help others to judge the value of your contribution; not just the information that leads to judgment in one particular direction or another
>
> Richard Feynman

3.1 Our Approach

In this chapter, we describe our approaches for both projects (Tangram then MirrorTrack). The 'Tangram project' aims at tracking physical shapes as well as users' fingertips above a surface using an overhead camera. The 'MirrorTrack project' aims at tracking the users' fingertips in space by using stereo cameras.

3.1.1 Tangram - TableTop:

The objective of the Tangram project is to study children's learning about mathematical reasoning via interactions with either geometric shapes laying on a TableTop surface, or their virtual counterparts.

In order to detect a geometric shape on the table, we first run a learning phase when we save the topology and the color of each kind of shape (tangrams). The system can then recognized those shapes on the surface. In order to do so, we use background subtraction as in [48, 49] or [78] followed by two different color segmentations. Indeed, we want to segment the foreground from the background and to cluster it in three parts: the tangrams, the hands, and the noise. Thus, the tangram colors are processed in normalized (r, g) space to make them insensitive to illumination variations, [44] and the hand skin color is processed in HSV in order to make the system robust as explained in [72]. Each tangram's colors are processed independently. This way, two tangrams with different colors can overlap or touch each other without confusing the system. From there, the system splits into two different threads: one for the hand and one for the tangrams. A template matching approach is used to locate the fingertips as in [61] and an optional Kalman filter is applied to the fingertip trajectory to smooth the result. This filter is optional because the result is smoother, but it adds more lag, and some users may prefer to disable it. In the meantime, corner and line detection algorithms are used to detect the tangrams [3]. Based on the shapes learned at the beginning, the system can still reconstruct the correct shapes during partial occlusion.

3.1.2 MirrorTrack:

The objective of MirrorTrack is to improve the interaction opportunities by using the information from two cameras as well as the reflection from the glass table surface.

MirrorTrack uses two side-mounted cameras for triangulation. The first step is to calibrate the cameras to learn their intrinsic and extrinsic parameters. We use a set of 48 points following the Tsai's calibration model [70]. This methods allows us to find the 3D location of a point with the

2D coordinates of this point from both camera images.

As with Tangram project, background segmentation and color segmentation are used followed by another template matching process to detect the fingertips. Then a Kalman filter is applied to smooth and track the fingertips. The main difference with Tangram project is that we employed a pyramidal approach for better accuracy and efficiency and that we detect the fingertip and its reflection. A last step consists in associating a reflection to a corresponding real fingertip and to improve the system by interpreting the meaning of the fingertip and its reflection position relative to one another (like hovering, click, ...). This constitutes a naive gesture recognition. This completely new approach was the subject of three papers [14, 15, 73] from Virginia Tech including one released in the Fall of 2009 at ICMI'09 and included in the Chapter 5 of this thesis .

3.2 Our Contribution

3.2.1 Tangram - TableTop:

TableTop design is very popular and a lot of different solutions were proposed. (see Chapter 2). The solutions adopted previously mainly use hardware support to provide robust information on the touch or object location. However, those solutions add extra costs, making them less attractive. Moreover, and more importantly, those solutions are strongly limited in their initial purposes and cannot evolve to match new requirements (i.e, if the system is designed to track circles, it usually cannot track anything else).

By adopting a software-based solution, we have the possibility to evolve and to be flexible to accept new requirements. Moreover, functionalities such as hovering and gesture recognitions can be implemented on our system. Finally, the final product is very inexpensive and more affordable. However, this solution is far more challenging technically than a hardware-based solution. It requires more computation to equal the hardware's performance. It is also very difficult to control the environment because each modification can impact the whole system's architecture.

Advantage: Flexibility, easy to update/upgrade and add new functionalities. Not restricted by any hardware solution. Inexpensive.
Drawbacks: More challenging - occlusion problem difficult to solve and environment control. Less reliable, less marketable.

3.2.2 MirrorTrack:

MirrorTrack aims at detecting fingertip locations in 3D. Compared to solutions using a single 2D camera, MirrorTrack can accurately estimate the 3D fingertip locations. Most existing solutions using stereo cameras or 3D cameras are based on Inverse Kinematic methods or equivalents to evaluate the position with some clues to reconstruct the full position and hand structure. Those methods are very computationally expensive and require fine tuning to reach the correct results. The method presented in this thesis provides a more accurate technique by using hand reflection as additional information.

Advantage: Increased Accuracy, improved interactivity, complete 3D fingertip information, 'smart' cameras setting minimizing occlusion problem, Inexpensive.
Drawbacks: More challenging, needs camera calibration, less marketable.

Chapter 4

Design of 'Tangram' software

> Most of the fundamental ideas of science are essentially simple, and may, as a rule, be expressed in a language comprehensible to everyone.
>
> Albert Einstein

4.1 Introduction

This section presents the Tangram project, a project aiming at helping children to learn and understand basic geometrical concepts through play. This chapter is focused on the implementation side of the project and describes technical issues and solutions. All the concepts of learning by playing, the other part of this project, are currently developed by the MediaLab center of Virginia Tech: http://www.colab.soe.vt.edu/itemL/

This section introduces a concept of games based on shape recognition and tracking. We use a library of models to detect and match shapes. Our proposed method uses clues (corners and boundaries) to reconstruct a shape even if it is partially occluded. Several teams tried to create

general matching algorithms using shape context [16, 31, 25, 9]. In our case, we define unique characteristics for each shapes processed (the number of corners and the largest line between two corners) simplifying the concept of shape matching. However, this rule is occlusion-sensitive, and we will discuss a solution to overcome possible problems during shape recognition.

4.2 Definitions and Concepts

4.2.1 Presentation of Tangram project:

The current Tangram project was initiated in 2008. Its objective is to study the learning process of preK-3 students using geometrical concepts, especially through the recognition and manipulation of shapes. This project is composed of two collaborating teams together, the first focused on the social learning aspect, and the second working on the technical implementation of an interface based on shape matching.

4.2.2 Functionalities of the Software:

Modes of interaction

The software developed for the Tangram project is structured in three modes (as described in Table 4.1).

In its first mode, tangrams are tracked and their shadow images are displayed on the screen in real time (around 20 fps for an immediate causality as explained in [12]). In order to handle the problem of occlusion when a hand comes between the camera and the tangrams, a database of shape models is created at the first step of the program (learning period) and then used to reconstruct the missing parts during occlusion. However, it is also possible to save this database to skip the learning part later on.

Table 4.1: Short description of the three modes of Tangram project:

The project include three modes:

The reality manipulation, the Virtual manipulation, and the Augmented manipulation mode.

- In the physical manipulation mode, real colored basic geometric plastic shapes called 'tangrams' are laid on the table surface and the children can manipulate them freely (without any constrain from the system). In the meantime, the system recognizes the shape and draws a shadow image of the tangrams below them (Objects tracking and recognition with occlusion treatment).
- In the virtual manipulation mode, the real tangrams and their shadow are disconnected and behave independently. If the real tangrams are removed, their shadows remain displayed on the screen and the children can manipulate them with their fingers. (Hands tracking).
- In the parametric manipulation mode, the children can manipulate the shadow only through Markers (with constraints from the system).In other words, this mode consists in parameter control by manipulating physical proxy. Three Markers are provided, X, Y, and Rotation. When those Markers are placed on the corners of the reflective surface (active area), rotating them allows the children to control not the tangram itself, but the abstract parameter attached to it, such as X location, Y location or angle of Rotation. The children can change the Marker to control successively those three parameters (Object recognition without occlusion).

shadow images remain. The users' fingertips are then tracked in order to create interactions between the user and the virtual tangrams displayed. Those interactions are translation and rotation through multi-touch detection. A physical engine called *Pymunk*, (the python interface to the C physics engine *Chipmunk*) is used to handle the problem of collision between two shadow images and to create more intuitive interactions, closer to reality.

A third mode is possible as an alternative to the second mode. This mode offers shadow parameter control by direct manipulation of a physical proxy. It consists in recognizing real Markers: small tangible objects with a symbol on the top. They are placed by the children in the active area (the corners of the tabletop surface in our case). The children manipulate the Marker to interact with a virtual tangram.

Comparison of Tangram modes

This section will compare the advantages and weaknesses of the three modes, Physical manipulation mode (*Physical environment for learning*), Virtual manipulation mode (*Virtual environment for learning*) and Parametric manipulation mode (*Physical and Virtual environment for learning*). As for the Mode 3: *Parametric manipulation environment*, it combines the advantages of both physical and virtual environments modes as well as additional unique opportunities. The mix between real and virtual is more than just the addition of both environments because it brings a new way to interact with the system by enhancing the reality. Those improvements can also increase the learning reasoning of the user by mixing relaxing constraints with real tangible manipulation.

Tangram architecture

Moreover, there are two layers in the implementation (see Figure 4.1). A low layer mainly relies on Image Processing algorithms to identify, detect, and track objects or fingertips. This layer is described in Section 4.3, Section 4.4 and Section 4.5. A flow diagram of the overall process is available in Figure 4.2. The information thus gathered is sent to the high layer discussed in Section 4.6. This layer displays the result on the tabletop surface screen.

Table 4.2: Mode 1: *Physical environment* for learning (based on [6])

Advantages	**Weaknesses**	**Learning skills stimulated**
direct and intuitive manipulation	passive representations of models and lack of visual *feedback* during manipulation	understanding of the domain and the problem
tangible, tactile *feedback* during interaction (low level feedback)	no feedback in response to decisions made (no high level feedback)	manipulative learning
social interaction (common center of interest between users and "body language" during manipulation of physical objects)	fidelity to the real reality, → Alternative realities difficult to model	collaborative learning through social interaction
constraints defined by the physical objects per se (boundaries of the objects enforced, no overlaps)	collection of data during manipulation harder than with Virtual objects	constraints help the learning process by structuring the reasoning (step by step), can highlight conflicts

Table 4.3: Mode 2: *Virtual environment* for learning (based on [6])

Advantages	**Weaknesses**	**Learning skills stimulated**
well-suited for dynamic models and visualization of behavior	Learning curve (need some training for the user)	learning enhanced by dynamic models
dynamic feedback during manipulation	no tactile feedback (or little)	learning by analyzing decision consequences
flexible virtual models (dynamically change size, color, shape...)	interaction peer-to-computer (difficulty to have peer-to-peer and peer-to-computer simultaneously)	learning in alternate realities
Ease to implement constraints (rules) during manipulation	complex implementation for interaction (i.e physic engine, ...)	learning by relaxing constraints, assessment tools, etc...

It is interesting to have these three modes of interaction together because each of them bear their own advantages in term of mathematical understanding. For example, the physical manipulative helps the understanding of geometric problems solving while being poor in geometrical representation. The virtual manipulative helps the understanding of geometrical representation while being

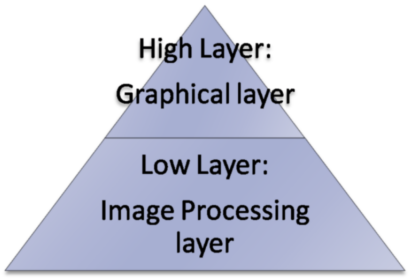


Figure 4.1: Representation of the 2 layers of 'Tangram Project'.

poor for the understanding of geometrical problems solving. Finally, with parametric manipulative, we allow the conceptualisation of abstract parameters such as translation or rotation.

4.2.3 Definition of tangrams:

The tangrams, are colored basic geometric plastic shapes. Those shapes can be used as puzzle and are well-suited for children's Mathematics learning [64]. The shapes used during the development of this software are shown above (see Figure 4.3). It is possible to use other shapes as well, but our current implementation recognizes only convex shapes with three or four corners and straight edges.

There are five distinct shapes in a tangram set, (three triangles, one square, and one parallelogram) and four colors (red, blue, yellow and green). Those criteria (shapes and colors) are stored together in the software.

The learning step allows us to create a tangram library storing all the models of shapes and colors detectable by the software. Even if shapes and colors are stored together in groups of two (each Tangram constitutes a shape and a color), permutations are possible. At least one kind of color and one kind of shape must be present in the library, but shapes and colors are processed independently (i.e. if two shapes are stored in the library, a red square and a green small triangle, four possible tangrams are detectable: red square, red small triangle, green square, green small triangle).

From here on, 'tangram' will refer to the real piece of plastic shape. 'Shadow' will refer to the

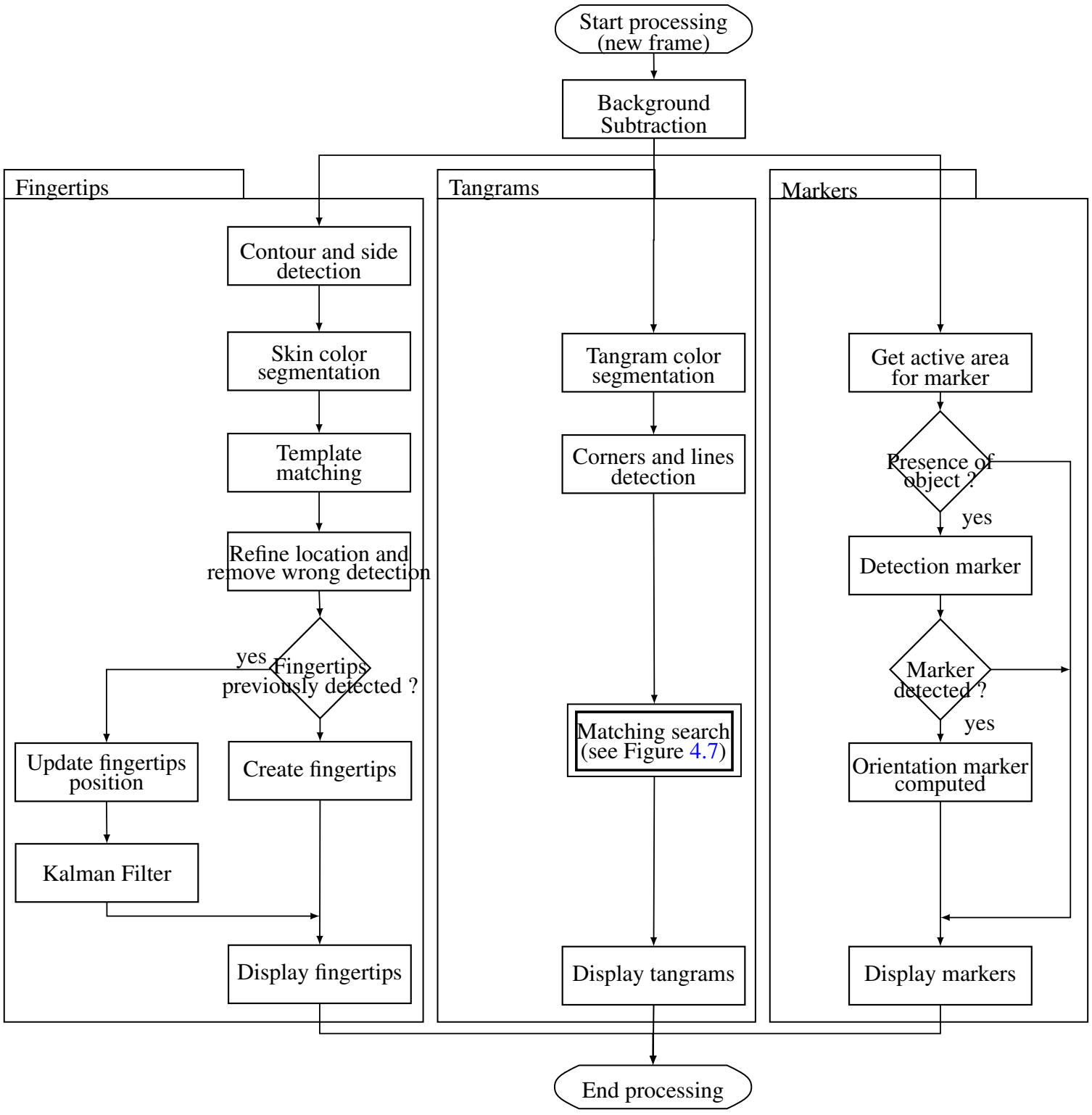


Figure 4.2: Flow diagram of the processes in ‘Tangram Project’.

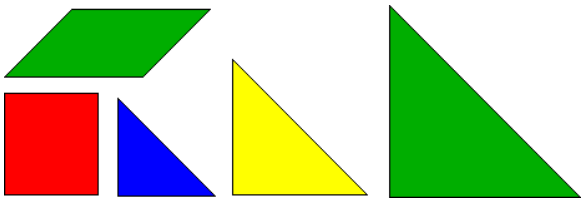


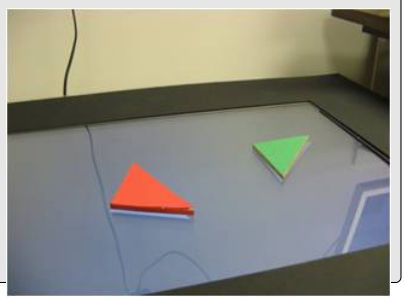
Figure 4.3: Shapes used during the implementation

virtual objects.

4.3 Physical manipulation mode: tracking of the tangrams.

Overview:

This section presents the techniques for objects' tracking and recognition. This allows a physical manipulative mode. The physical mode is important because physical manipulative helps the understanding of geometric problems solving.



This first mode consists of four main sub-steps (see Figure 4.4). Those steps aim at tracking and displaying a shadow for each tangram. In order to do so, the software uses two stacks called 'tangram library' and 'displayed tangram'. The first stack stores the models of tangram, colors and shapes whereas the second stack stores the location of each tangram to display on the current frame. This separation in two stacks allows us to have a full shape reconstruction even during occlusion because the 'tangram library' is never modified during the process, and is only used for reference.

4.3.1 Learning period:

Learning for Real manipulation:

The learning period function is performed at the beginning of the process in order to define a library

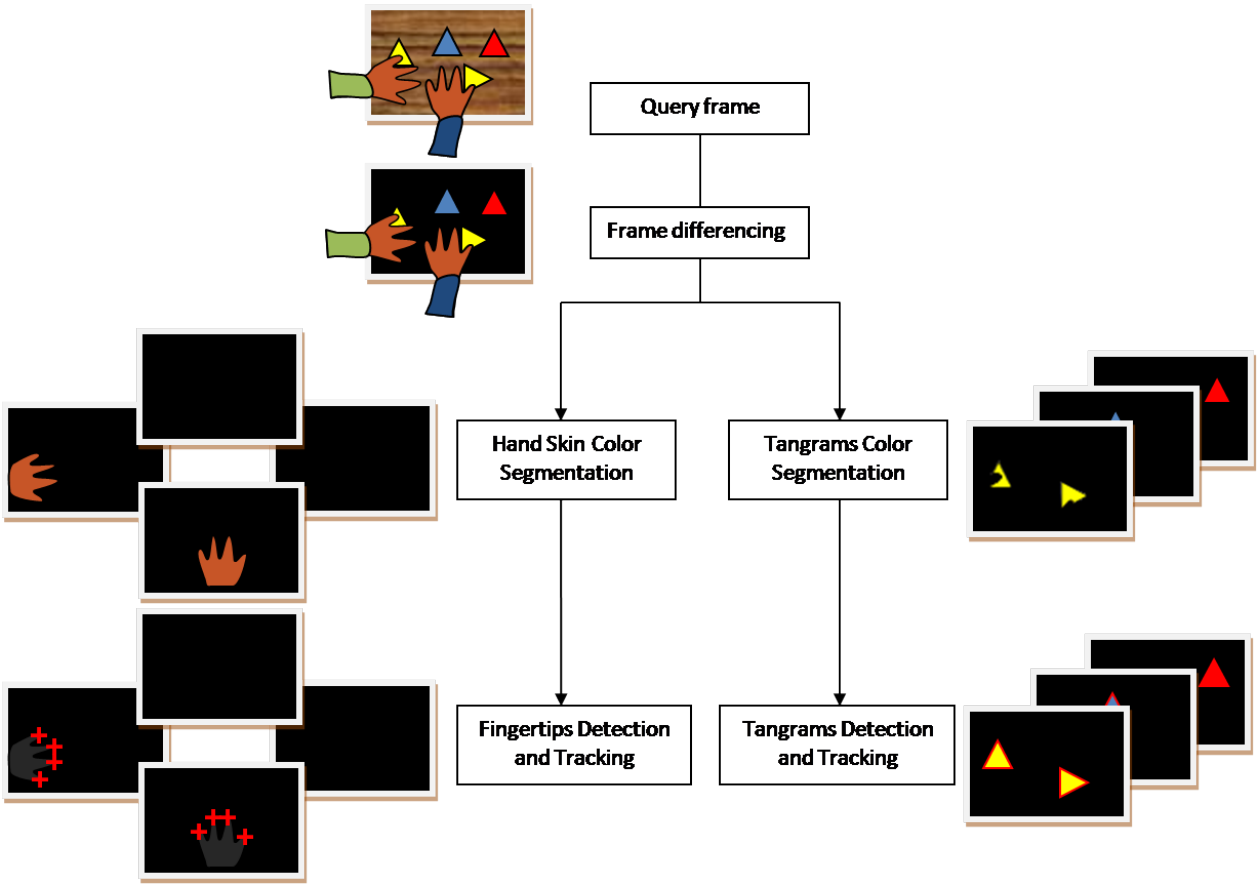


Figure 4.4: Diagrams of the Real and Virtual manipulation modes

of shapes 'tangram library'. This step could be skipped if a library has already been defined and saved in the system. This library will be used to track the tangrams by colors and then will be used to reconstruct the shape displayed, even if the tangrams are partially occluded (the term of partially will be defined later).

To execute the learning process, shapes are placed on the table surface. The learning function can detect and store every new object in front of the camera. In other words, a frame differencing process (see Section 4.3.2) reveals the Tangrams and their contours are detected using a connected-component algorithm introduced by Suziki & al. [69] (see Section 4.3.4). Each contour is processed to look for corners (strong derivatives) proposed by Shi and Tomasi [66]. Those corners are linked to form the new shape and displayed. The average color inside the formed shape is computed and saved as the color of the new shape.

4.3.2 Frame differencing:

Overview:

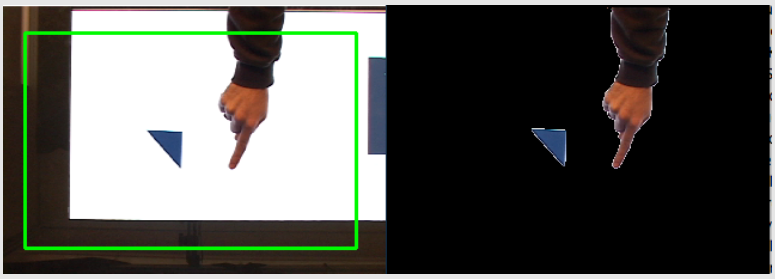


Figure 4.5: Result of frame differencing

Why using this method:

- All the objects in front of the camera are revealed by this process.
- Helps the next step (color segmentation) to correctly segment the hand from the background.

Frame differencing is a process aimed at detecting new events by subtracting a frame of reference from the current frame. This step is composed of two sub-steps. First, a reference is created and then the subtraction with the current frame is processed to detect new events (see Figure 4.13).

In our case, the first frames are grabbed at the launching of the software. We store an average value of each pixel from those frames as the reference. During this process, no object (and more particularly no hands) should be in the field of vision of the camera. The subtraction is processed on each new frame and the result reveals the foreground of our scene. The result of the subtraction may be affected by noises (light variation, vibration of the camera, ...). Therefore, in order to handle those issues, the process considers only 'strong' modifications between the reference and the current frame, and the reference is smoothed with a Gaussian filter. We find that the detection of new events is more effective if the subtraction is made on the three channels and that the color space does not significantly affect the result.

Usually, this process is completed by a final step where the reference is updated 'smartly' [74]. This step is implemented in our solution and improves the result.

4.3.3 Color segmentation:

Overview:

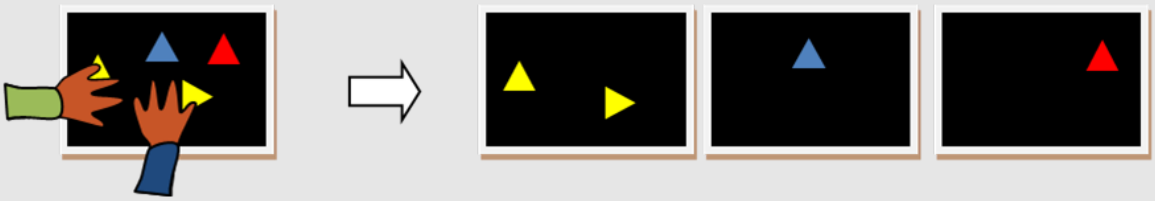


Figure 4.6: Result of color segmentation

Why using this method:

- Clusters the tangrams from other objects detected during the background segmentation
- Eases the next step by grouping the tangrams by color
- Technical details:

  Process in normalized r-g space

  Each tangram color is trained initially and its mean and variance are computed for each channel

The goal of the color segmentation process is to detect the object based on its specific color. We tried different approaches such as Maximum Likelihood Extimation, HSV, RGB, *rg* normalized, and we finally decided to implement a simple 'box' criteria in *rg* normalized. The frame is parsed and for each pixel on each channel, the value is checked to match a criteria closely:

$$\forall (x,y) \in \mathbb{N}^2, I'(x,y) = \begin{cases} \text{I(x,y)} & \text{if } |I_r(x,y) - \text{Ref}_r| > threshold_r \\ & \text{and} \quad \text{if } |I_g(x,y) - \text{Ref}_g| > threshold_g \\ & \text{and} \quad \text{if } |I_n(x,y) - \text{Ref}_n| > threshold_n \\ 0 & \text{otherwise} \end{cases}$$

with I'(x,y) the destination pixel, $I_r(x,y)$ the current frame's pixel on channel r normalized, Ref_r the criteria (reference) for the channel r normalized, and $threshold_r$ the 'tolerance'. The same for the two other channels.

The process is done in *rg* normalized space. We realized that we obtain good results when using saturated color such as those used on tangrams. To solve the problem of shape recognition when the shapes are in contact, we decided to have as many images as tangram's colors detected during the learning period: one image corresponding to one color. This way, when two shapes with different colors touch one another, the system will still continue to detect two distinguished shapes. We will talk later on about the case of two same-color shapes entering in contact.

The resulting frames show only new events (from the frame differencing) that match the colors of the tangram models stored in the 'tangram library'.

4.3.4 Matching search:

Overview:

Why using this method:

- Lines are used to confirm the detected corners location
- Corners are used to compute tangrams' location defined by a translation (x,y) and a rotation Θ
- Even during corners occlusion, lines' intersections will retrieve the corners

This step is the cornerstone of the first mode: *Physical manipulation mode - tracking of the tangrams*. It is very innovative and is explained in detail in this section.

The frames processed at this step result directly from the color segmentation process described previously in Section 4.3.3. Thus, we have as many frames as colors, and each frame is processed independently. From here on, we describe the steps for a single frame.

The algorithm is described step by step in Figure 4.7. The main functions are detailed here:

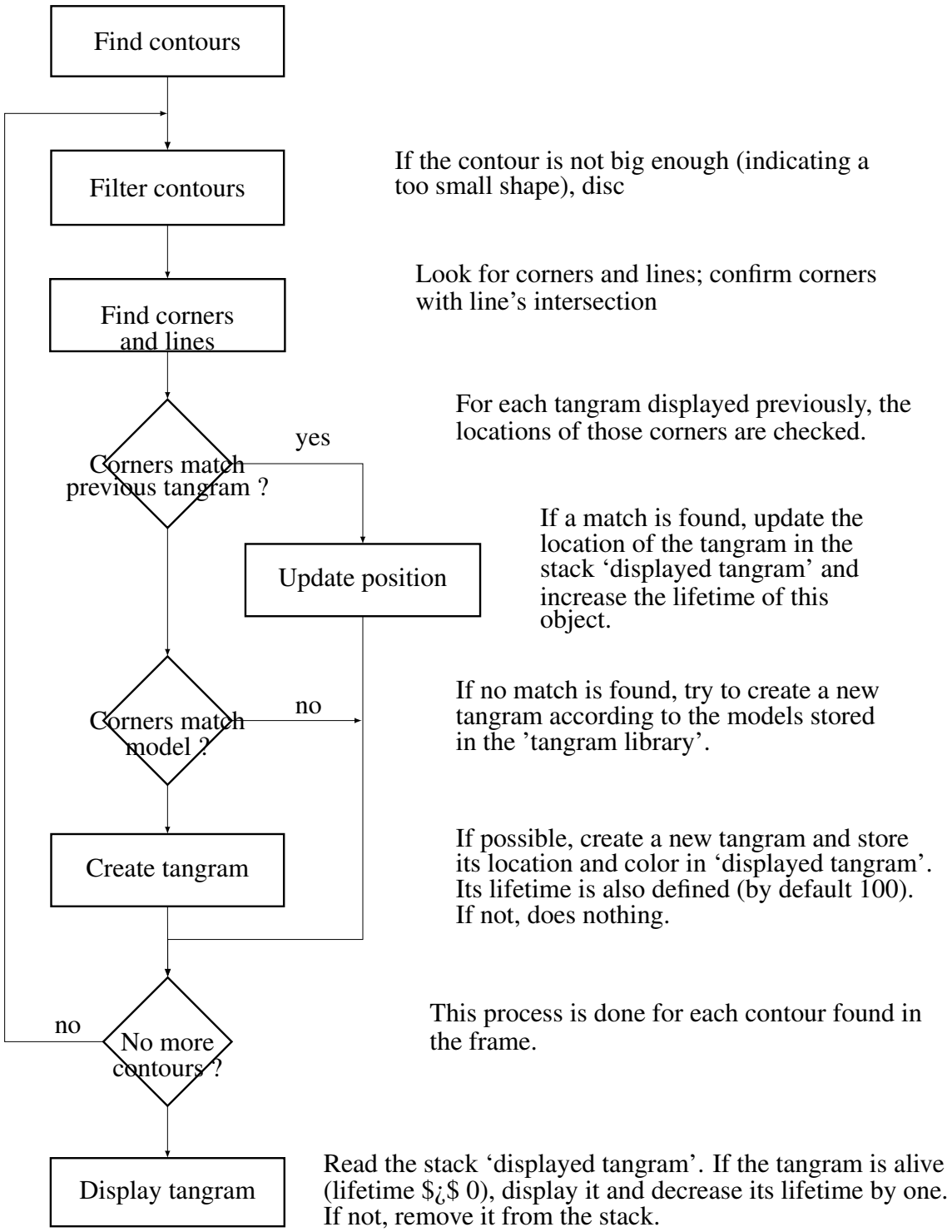


Figure 4.7: Flow diagram of Matching search

Find Contours:

Find Contours processes a binary image to detect the contour of each object. We used the iterative solution proposed by Suzuki et al. [69]. The result is a list of contours, each contour being a list of points. This list of points is then processed separately.

Filter Contour:

Filter Contour takes a contour (list of points) and counts the number of points. Because this list is generated as a succession of 8-neighbor adjacent points, the number of points composing this list is directly proportional to the contour perimeter. This linear relationship between the number of pixels and the length of the contour (in inches for example) allows us to process the size of the contour in pixels without necessarily needing to find a correspondence between distance and pixel. This way, we can discard the contours which are too small according to a predetermined threshold.

Find Corners and Lines:

This step is structured in two methods. (1) First the corners are detected using Formula [**?**] and (2) a hough transform is then performed to detect lines in the image. This second method is implemented because the first method may be affected by noise. That is why we computed the intersection of the lines to confirm whether the corners detected belong to a tangram or are simply noise.

Find Corners and Lines takes a contour (list of points) and draws it in a temporary image. This image is then parsed, looking for strong evidence of corners. This process relies strongly on the algorithm defined by Shi and Tomasi [66] based on the computation of the autocorrelation matrix

described in Formula 4.1:

$$\begin{pmatrix} \sum_{-K\leq i,j\leq K} w_{i,j} I_x^2(x+i,y+j) & \sum_{-K\leq i,j\leq K} w_{i,j} I_x(x+i,y+j) I_y(x+i,y+j) \\ \sum_{-K\leq i,j\leq K} w_{i,j} I_x(x+i,y+j) I_y(x+i,y+j) & \sum_{-K\leq i,j\leq K} w_{i,j} I_y^2(x+i,y+j) \end{pmatrix} \tag{4.1}$$

For each pixel, the autocorrelation matrix of the second derivative image is computed over a small window (size defined by K, see Formula 4.1). Shi and Tomasi showed that a corner is present if the smaller eigenvalue of this matrix is greater than a threshold.

Shi and Tomasi's algorithm is a good start but is not precise enough because it works at the pixel level. However, we need sub-pixel accuracy. Each corners' location found is used as a starting point for an other algorithm refining the location of those corners (see Chapter '10.3. Subpixel Corners' of [38]). This additional algorithm provides more precision, at the pixel level.

The result is a list of points (the corners) for each contour. This list is used to define the contour.

Moreover, using hough transform algorithm for parametric line [20], we detect the lines formed by the edges. This technique processes a binary image and 'votes' in an accumulator (r, θ) to detect all the possible lines going through each pixel processed. The research of a local maximum in the accumulator gives the lines' position in the source image. We decided to employ a Gaussian filter during the voting process to reduce the boundary artifacts appearing between two accumulator's bins that could possibly create line detection errors. The accumulator is filled following this rule:

for all $I(x, y)$ such as $I(x, y) \neq 0$ **do**
  for $\theta = 0$ to π **do**
    Gaussian voting of size K center on $r = xcos(\theta) + ysin(\theta)$
  end for
end for

We note that using the gradient direction of the edge will reduces the number of computation by

'voting' only in a small window around the detected gradient angle (i.e gradient angle $\pm 10^{\circ}$ for a window of 20 degrees) and not for the entire $0 - 180^{\circ}$ range.

Finally, we can now use lines and corners to recognize and match our reference tangram with the shape detected on the screen.

Corners match previous tangram: (registration)

At this step, we want to know whether or not we have to create a new shadow tangram (see Section 4.3.4) or update the previous position (see Section 4.3.4) of an existing shadow. It could also be an error that has to be ignored in case of a noise problem, or an occlusion for example.

To effectively look for the new location, the function computes the estimate of the current location of the corner based on its previous location. The previous locations are stored until t-3 (t-3 means 3 frames before the current frame). In other words, this function estimates the position of the point at t+1 based on its position at t, t-1, t-2 and t-3.

If the location of the corner is known until t-3 included, the algorithm computes the location based on a linear accelerated motion (the velocity is constant) and a linear accelerated speed rotation. An average of the linear accelerated motion is made (see Figure 4.8).

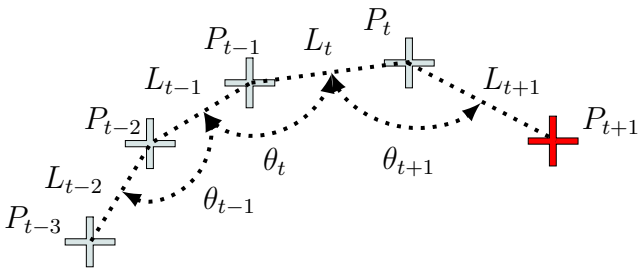


Figure 4.8: Prediction of the next position of each corner

We designed the following formulas to estimate the new position of the corner based on its previous positions.

$$L_{t+1} = \frac{({L_t}^2 \times L_{t-2} + L_t \times {L_{t-1}}^2)}{2L_{t-2}L_{t-1}}, L_{t-2} \neq 0, L_{t-1} \neq 0$$
$$\Theta_{t+1} = \frac{{\Theta_t}^2}{\Theta_{t-1}}$$

The general formula for t-n known location, $n > 2$ (with acceleration average) is:

$$L_{t+1} = \frac{L_t}{n-1} \times \sum_{i=1}^{n-1} \frac{L_{i+1}}{L_i}, L_i \neq 0, n > 1$$
$$\Theta_{t+1} = \frac{\Theta_t}{n-2} \times \sum_{i=1}^{n-2} \frac{\Theta_{i+1}}{\Theta_i}, \Theta_i \neq 0, n > 2$$

Create tangrams:

If none of the tangrams displayed previously have their corners relatively close to the corners found, we create a new shadow tangram defined by some or all of the corners found.

To create this shadow tangram, we need to find a unique physical characteristic for each physical tangram. If we can do that, then we can easily know which kind of tangram is the one laying on the table and forming the cloud of corners detected by the system.

The solution found was to compute the longest line linking two corners. In the case of the tangram set used for this game (and presented previously in Section 4.2.3), the length of this line and the number of corners (three or four) creates a pair of unique combination for each of our shadow tangrams.

The system looks in the stack 'tangram library' for the model matching these criteria. If none of them is found in the library, the function does nothing, and those corners are ignored. In the case of a match, the system tries to create a new shadow tangram. To do so, it makes the extreme points of the longest line match together. All the tangrams' shapes have the middle of the longest line between two corners as a central symmetry point except for the triangle. That is why the triangle (three corners) is the only shape requiring additional processing to localize the shape. It consists of checking if the last point (other than those of the line's extremities) is closely located to a previous

corner belonging to the tangram of interest. If not, we flip the extreme points to solve this problem.

Update position:

In the case when a match with the library is found (enough evidence of a match is present, such as corner correspondence, shape correspondence, and so on), we update the position of the relevant tangram.

To update the position, we seek, for each matching corner, the one that moved the shortest Euclidean distance. This corner becomes *'the reference'*. This method reduces strongly all the noise occurring in the location computed between two successive frames because the probability that all the matching corners were affected by noise in the same frame is small.

To move the shape to its new location, we do a rotation followed by a translation. To do so, a vector for the translation (between the reference's previous and new locations) and an angle for the rotation (between the largest line's previous and new locations) need to be computed.

Translation: The vector is defined by *'the reference'* corner as defined above and its previous location.

Rotation: Three methods have been tried to compute the angle of rotation:

- The first method calculates three angles. They are the angles between each side of the shape and its new position. The smallest angle computed is selected in order once again to minimize the influence of the noise. Figure 4.9 shows this concept.

$$Angle = min(\widehat{\overrightarrow{0'1'}, \overrightarrow{01}}, \widehat{\overrightarrow{1'2'}, \overrightarrow{12}}, \widehat{\overrightarrow{2'0'}, \overrightarrow{20}})$$

Unfortunately, this solution is not stable. When the tangram rotates with a small angle, the error is small. However, when the tangram moves too fast, its computed position has too much error. The shadow is moved far away from the real location. During the next iteration, the location is too poor

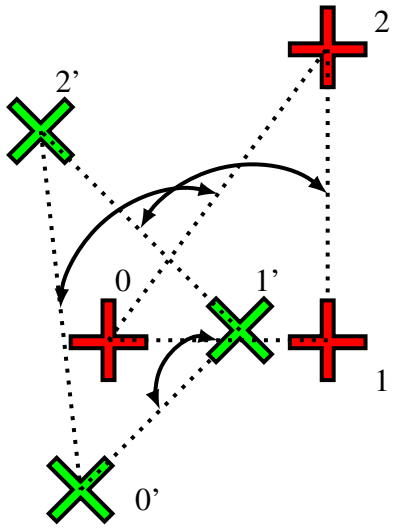


Figure 4.9: The red triangle (on the right) is the previous location. The green one (on the left) is the new one. For each corner, the angle formed between the previous side and the current side is computed. The smallest angle is selected.

for the algorithm to be able to auto-correct the errors occurring in the previous localization. The software is not longer able to locate the tangram.

- The second method searches for the new location. Once the reference corner is found (the corner that moves the least), the tangram first has to be translated before the rotation is computed. Then the shape is rotated -90°, and for each angle between 0° to 180°, the number of pixels inside this shape is computed (in other words, we use an AND operator between the shape pixels and the tangram pixels). This way, the maximum number of matching pixels is found when the shape best fits the image at the angle X. The angle sought is X-90. (see Figure 4.10)

$$\boxed{\begin{aligned} X &= \frac{(\Theta 1 + \theta 2)}{2} \\ Angle &= X - 90 \end{aligned}}$$

Unfortunately, this solution is too computationally expensive to be seriously considered.

- The last approach aims at working even if only one corner has been found. It looks around this corner (10 pixels around it) to define completely the corner by finding two other points.

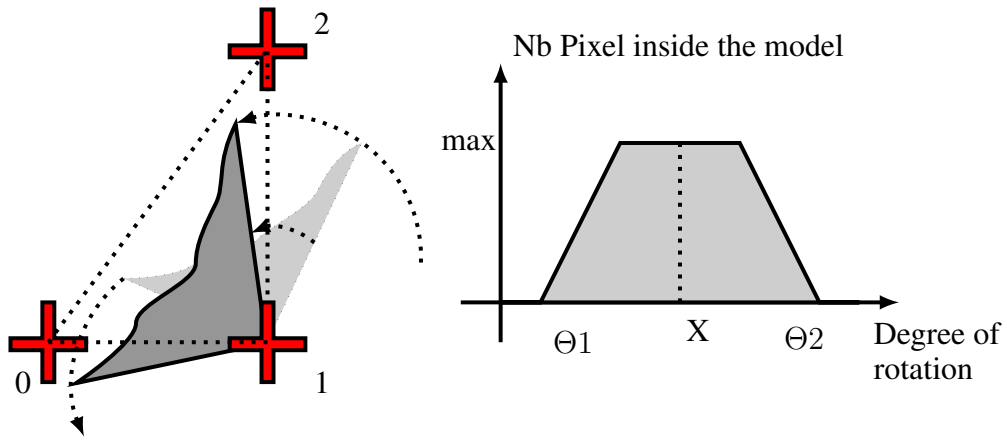


Figure 4.10: The value of the angle where the maximum of pixel is found is a range rather than a unique value. That is why the computation of X, Θ1 and Θ2 is needed.

The angle bisector of this corner is then computed and the angle between this angle bisector and the angle bisector of the previous location is the angle analyzed (see Figure 4.11).

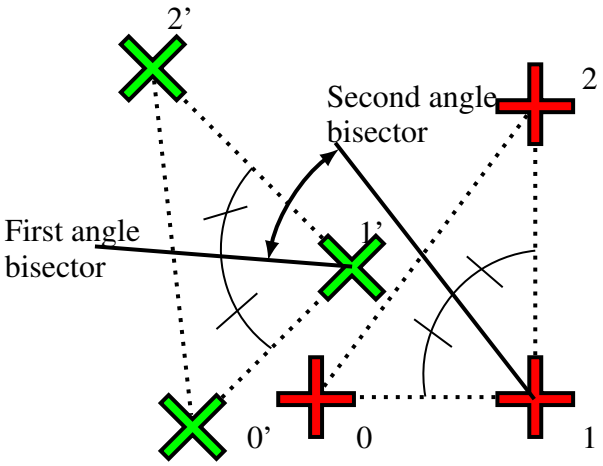


Figure 4.11: The red cross (on the right) is the previous location of the reference. The green one (on the left) is the current one. The square is the window in which the edge is searched. The two points found help to create the first angle bisector with the reference. The second angle bisector is easily computed from the previous location. The angle formed by those two angle bisectors is our angle of rotation.

Unfortunately, this solution does not work very well because the border of the tangram is defined by a jagged pixel staircase. This form creates some inexactitudes.

Finally, we decided to implement the first solution and to limit the drawbacks discussed previously by creating some restrictions:

- We experimentally found that a normal tangram manipulation cannot generate a rotation angle of more than 20° per frame, thus a result higher than this limit is ignored (or thresholded to 20°).
- We do not compute the rotation angle when a large occlusion is detected (such as not enough corners detected, distances between corners correspondence too big, area of the shape too small comparing to the area of the non-occluded shape, and so on).

Once the new location is found (rotation and translation as explained above), the tangram's 'lifetime' is increased to indicate that this tangram was found and is still on the screen. By updating the status of the tangram every time (as being detected or not), we can handle occlusion more easily, by temporary keeping the tangram information. This way, a brief occlusion will not remove the shadow from the screen, and the historical data of the tangram will be available as soon as the tangram is again fully detected (no occlusion).

Detection of multiple tangrams in contact:

If the tangrams have different colors, the segmentation between touching tangrams operates correctly because each tangram belongs to a different group (and thus is stored in a different image).

However, if touching tangrams are of the same color, we need to do a CPU-expensive search to segment the whole shape in various tangrams. In order to do so, we first need to detect when a shape needs to be split into separate tangrams. The number of corners may be a solution, but occlusion (during manipulation) will also temporarily increase the number of corners detected. That is why we decided to compute the area of the shape to decide when a shape needs to be split (Indeed, an occlusion can only decrease the area but never increase it).

Once we know that we have a shape to split, we look around the current shape position for the

previous positions of the tangrams. This will provide information about which tangrams have to be moved and how many. Moreover, the area could also provide the number of tangrams forming the shape, but occlusion will again make this process challenging.

Moreover, we decided to neglect the rotation between the two successive frames during the period when the tangrams come in contact. This way, we can compute the minimum move needed to match the tangram with its previous positions. Iteratively, we can split the shape in as many tangrams as it is constituted of. However, it must be noted that because the rotation was neglected during the last movement, this solution is not guaranteed to find the correct result every time and may be affected by the way the user manipulated the tangram.

Tracking stack:

The stack 'displayed trangram' is now up-to-date. It is parsed and for each tangram inside, its lifetime is checked. If the value is zero, the tangram is removed definitively from the stack. In the other case, it is displayed and its lifetime is decreased by one. As just explained, this variable 'lifetime' allows us to display the tangram even if this one is completely occluded during a small lapse of time (if the variable is initialized to 100 frames and is decreased at each frame by one, the tangram stays on display for approximatively 5 seconds during occlusion). In other words, even if the tangram is not in the field of vision of the camera anymore, it will be displayed for some seconds and only then disappear.

4.4 Virtual manipulation mode: tracking of the fingertips

Overview:

This section presents the techniques for fingertips' tracking; allowing a virtual manipulative mode. This mode is important because virtual manipulative helps the understanding of geometrical representation.

The second mode requires that some or all tangrams be removed while keeping their shadow images on the screen. This time, the user can move the shadows with his or her fingertips as if they were real tangrams. To do so, we need to track the user's fingertips.

This virtual mode uses the data provided by background subtraction already explained in the physical manipulation mode (see Section 4.3.2). Moreover, a learning period is also required to learn the skin color of each user in order to improve the performance of the system (an empirical skin color value is set by default but it improves the system performance to learn each user's skin color).

4.4.1 Learning period:

The learning period is used to learn the skin color of each user in HSV space (this space provided the best result for skin color segmentation [84]). The users' hand is placed inside an active area indicated on the screen, and an average of the skin color is computed and saved in a database of skin colors. This skin color will be used during the skin color segmentation by the system.

4.4.2 Hands's side identification:

Overview:

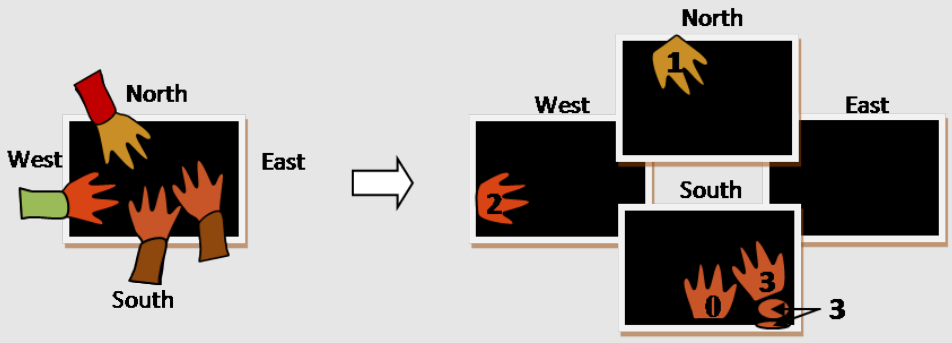


Figure 4.12: Hand side classification

Why using this method:

- Allows multi users detection
- Each side is defined as one user, so a maximum of 4 disctinct users can be detected by the system.

Before the skin is segmented by color, we need to detect from which side the user's hand is coming. This way, we will be able to determine to whom the hand detected belongs. We thus have 4 groups of hands in which the result of the skin color segmentation will be saved, one for each side of the table (let's called them North, South, East and West side as indicated in Figure 4.12). We compute the contour of each object using the algorithm presented in Section 4.3.4, and the pixels of each contour are labeled with the number of contour just processed (so labels go from 0 to nb_contour-1). The contour is checked to determine from which side it is connected and then a skin color segmentation in HSV space is performed for each user's skin color learned by the system.

4.4.3 Skin color segmentation:

Overview:

Figure 4.13: Result of frame differencing

Why using this method:

- Clusters the hands from other objects detected during the background segmentation
- Technical details:

  In HSV space.

  The system is trained on the user's skin color to obtain the mean and variance (for each channel)

  Shift of 0.2 on the H channel. (see [47])

  Discard V value.

  Only the largest blob by contour is processed.

This skin color segmentation is similar to the one presented in Section 4.3.3 but in HSV space. The result (a mask image) is saved in an image representing the side from which the hand enters the screen (North, South, East or West), with as pixels' value the label explained above. This mechanism is illustrated in Figure 4.12. Using labels allows the system to cluster the blobs resulting from the skin color segmentation by contour, and then we sort the contours by size for each cluster independently. This way, we can eliminate noisy blobs in case of bad color segmentation as explained below.

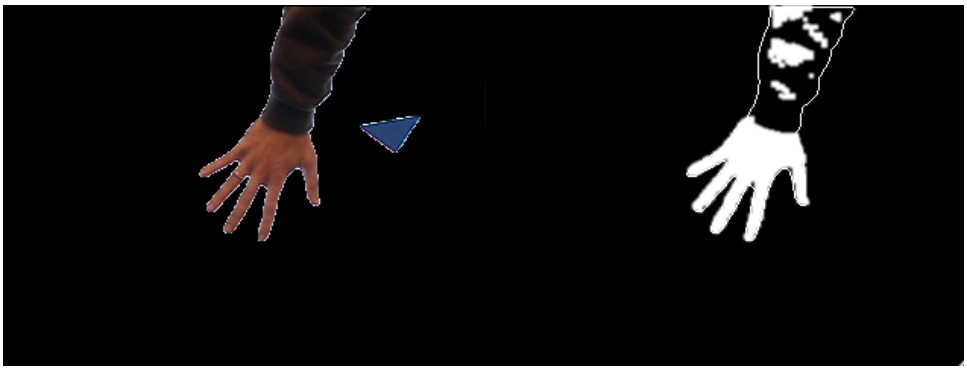


Figure 4.14: Example of multiple blobs due to the clothes' color that are too similar to the skin color.

In case of bad skin color segmentation, multiple blobs for the same contour could appear (see Figure 4.14). In this case, only the largest blob per contour is processed because we noted that the hand generally forms the largest one and the other blobs are noise due to the similarity between the clothes' color and the skin color. This occurs when the clothes are brown for example but processing only the biggest blob (in term of area or number of pixels) solves this problem as illustrated Figures 4.12 and 4.14.

4.4.4 Fingertips detection:

Overview:

Figure 4.15: Fingertips detection

Why using this method:

- Extracts fingertips location from blobs
- Uses a circular template with black boundaries
- This gives better results than a curvature approach

Once the blob of the hand is determined by the previous step, we use template matching with a circular template to determine the fingertips candidates. To correctly operate, the circle should have a size similar to the fingertips and a '0 pixel value' boundary must surround the template to simulate the 'empty space' around each fingertip.

The template matching used is defined as:

$$Result(x_1, y_1) = \frac{\sum_{x_2,y_2}(Template(x_2, y_2) - Image(x_1 + x_2, y_1 + y_2))^2}{\sqrt{\sum_{x_2,y_2} Template(x_2, y_2)^2 \cdot \sum_{x_2,y_2} Image(x_1 + x_2, y_1 + y_2)^2}}$$

with $Result$ the resulting map in which we look for local maxima to locate fingertips, (x_1, y_1) being the location in the Image and (x_2, y_2) the location in the template.

During the search for the local maxima, if a cluster of maximum values is found, we use the barycenter of this cluster as the fingertips' position.

Once the fingertips are detected, we have to prune the result in order to keep only the interesting ones. We keep only the fingertip that is the furthest to the side from where the hand is coming to eliminate the false detection at the wrist.

4.4.5 Fingertips refinement:

Overview:

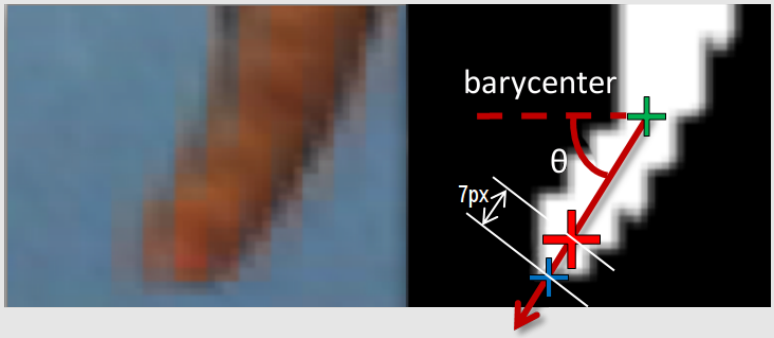


Figure 4.16: Fingertips' refinement

Why using this method:

- Decreases the jitter (x2 accuracy on fingertips position)
- Technical details:

  Computes barycenter and angle Θ

  Iterates on the line until fingers extremity

We decided to add one more step in our process to refine the fingertip's location. Indeed, all the previous steps processed 320x240-resized frames in order to save resources and remain in real time. However, the fingertips location will be mapped to the 2900x1600 table screen with an homography function (see Section 4.6.2). This extreme difference between resolutions makes the

fingertips' location jitter (1 pixel of error before homography results in approximatively 8 pixels of error on the screen table). Thus, we implemented a pyramidal approach in two levels like with MirrorTrack project. A window around the location of the fingertips detected from the original frame (before resizing) is extracted and a skin color segmentation in HSV is used to segment the hand. We then compute the barycenter $(\bar{x}, \bar{y})$ as well as the blob orientation θ as explained in Suzuki et al. [69]:

$$\bar{x} = \frac{M_{10}}{M_{00}}, \bar{y} = \frac{M_{01}}{M_{00}} \text{ with } M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Then we compute each central moment as follows:

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

And finally, the angle θ is obtained as:

$$\Theta = \frac{1}{2} \arctan\left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}\right)$$

Once θ and $\bar{x}, \bar{y}$ are obtained, we use a Canny filter on the shape to detect the edges. At this point, we go through the line passing by the barycenter and forming a θ degree angle with the x axis and we test each pixel on this line. When the pixel value reach an empirical threshold, we are at the finger extremity and stop the process. This location is then converted back in the resized, smaller, frame (in float value) and will be used for the fingertips tracking.

4.4.6 Fingertips tracking:

Overview:

Why using this method:

- Decreases the jitter by smoothing the trajectory
- Provides a pseudo sub-pixel accuracy
- Technical details:

  Computes an estimate value, the uncertainty of the estimate value, and then do a weighted average between the measured value and the estimated value.

Finally, we decided to smooth the trajectory of each fingertip by using a Kalman filter [76]. The Kalman filter will provide to the system an estimated position of a fingertip when the fingertip is not detected.

Our Kalman filter is designed the most common way with an update and a prediction step:

Table 4.4: Kalman Filter cycle

$Predict$:	given $\hat{\mathbf{x}}_{k\|k-1}, \mathbf{F}_k, \mathbf{Q}_k, \tilde{\mathbf{y}}_k, \mathbf{z}_k, \mathbf{H}_k, \mathbf{R}_k$ with:
$\hat{\mathbf{x}}_{k\|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1\|k-1}$	$\hat{\mathbf{x}}_{k\|l}$ (respectively, $\mathbf{P}_{k\|l}$) the a posteriori *state estimate* (respectively, *error covariance matrix*) of time k with observation up to time l.
$\mathbf{P}_{k\|k-1} = \mathbf{F}_k\mathbf{P}_{k-1\|k-1}\mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k$	
$Update$:	$\mathbf{F}_k$ the state transition matrix of time k
$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k\|k-1}$	$\mathbf{Q}_k$ the process noise matrix of time k
$\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k\|k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k$	$\tilde{\mathbf{y}}_k$ the estimate measurement of time k
$\mathbf{K}_k = \mathbf{P}_{k\|k-1}\mathbf{H}_k^{\mathrm{T}}\mathbf{S}_k^{-1}$	$\mathbf{z}_k$ the noisy measurement of time k
$\hat{\mathbf{x}}_{k\|k} = \hat{\mathbf{x}}_{k\|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k$	$\mathbf{H}_k$ the observation model of time k
$\mathbf{P}_{k\|k} = (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k\|k-1}$	$\mathbf{R}_k$ the observation noise of time k

When we do not have new measurement (in case of occlusion), the best we can do is to update the system with its own prediction as follows:

$value \leftarrow Predict$

if newMeasurement **then**

$value \leftarrow Measurement$ and $value$

end if

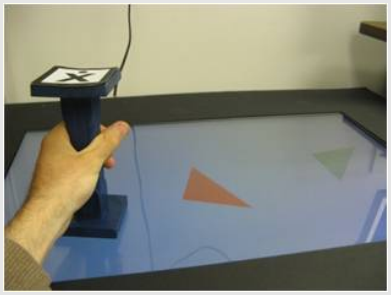
$Update \leftarrow value$

The Kalman filter successfully smooths the trajectory and handles occlusion correctly. However, it also adds a small lag between the measurement and the prediction. Indeed, the more we want to smooth the trajectory, the more lag we have. That is why we keep both measurement and prediction during the processing and we can switch from one to the other at any time depending on the user preference.

4.5 Parametric manipulation mode: tracking of the markers

Overview:

This section presents the techniques for markers tracking. This allows a parametric manipulative mode. This mode is important because parametric manipulative helps the conceptualisation of abstract parameters such as rotation or translation.



The Parametric mode is based on immersive reality. We enable the user to manipulate real markers placed on the surface in order to interact 'indirectly' with a shadow tangram instead of doing direct manipulation with fingertips. This way, children can manipulate parameters themselves.

A marker is a tangible object with a specific character or symbol on its top depending on its action on the virtual space. To avoid occlusion problems met during the real manipulation mode and described in Section 4.3, our Markers have a vertical height to allow the user to grab them without masking (partially or completely) the symbol on the top (see Figure 4.17). Therefore, a

Figure 4.17: Grip for the marker.

simple shape recognition with rotation, translation and scale invariance will identify and localize the Markers laying on the table.

Figure 4.18: Illustration of the two possible solutions for the markers designed and interaction possibilities (two or three markers) Note that we designed two solutions for the rotation marker.

First, we want the markers to be used as a ‘joystick’ to manipulate the shadow on the surface. We thus have to control three Degrees of Freedom *(DOF)* (Translation X, Translation Y, Rotation Z) with those markers. Two possible solutions can be proposed: (Illustrated on Figure 4.18)

- Solution 1: We use two Markers, one controlling the Translation, the other controlling the Rotation. To separate the Translation in two separate cases (X and Y) with one Marker, we could identify the initial orientation of the Marker to differentiate the X from the Y Translation (see Figure 4.18).

- Solution 2: We use three Markers, one for each *DOF*.

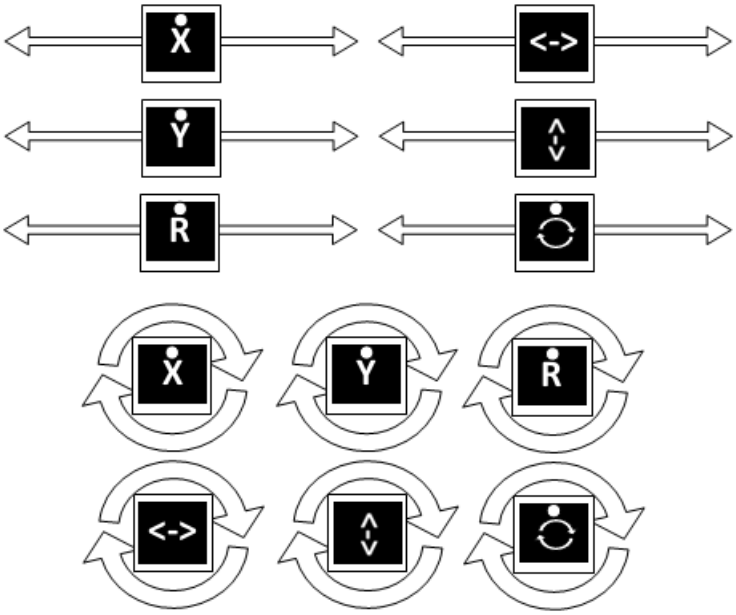


Figure 4.19: Illustration of the two possible solutions for markers manipulation (Translation or Rotation) for each possible marker.

Secondly, we need to lock a marker manipulation to a specific shadow interaction. Two designs for users' markers manipulation are proposed: (Illustrated on Figure 4.19)

- Solution 1: The markers are put down on the table. Their initial location is identified as a reference for the system. The marker can be translated on a virtual line parallel to the side from which the hand comes from to manipulate it. The projected position of the marker on this line determines the new value for the DOF represented by the marker.

- Solution 2: The markers are put down on the table. Their initial location is identified as the reference for the system. The Marker can be rotated around its center of mass, and the rotated value computed determines the new value for the DOF represented by the marker.

Finally, we need to define a metric correspondence between the value (rotation or translation) measured on the marker and the shadow tangram's parameter affected. Once again, two solutions are proposed.

- Solution 1 *Absolute value*: When a marker is manipulated, the value measured (rotation or translation) is mapped to a scale between 0 to 100. This new scale represents a percentage of the shadow tangram's parameter (for example, if the marker affects the X *DOF* parameter, a value of 36 will move the shadow tangram of 36% of the width of the screen, a marker Y with 19 will move the shadow of 19% of the height, and so on...).

- Solution 2 *Relative value*: When a marker is manipulated, the value measured (rotation or translation) is rounded on a scale from -5 to 5, 0 being the reference position. For each frame, the value of this scale is added to the shadow parameter (for example, if the marker affects the X *DOF* parameter, a value of +2 will move the shadow tangram to its current position+2, a marker Y with -5 will move the shadow to its current position-5, and so on...).

Now that we presented the behavior of the system, we will introduce the technical solution implemented to identify and localize the Markers.

4.5.1 Learning period:

In order to detect the markers, we first need to 'learn' them. Thus, we used the solution proposed by Suzuki et al. [69] to detect contours and then we compute seven translation, scale, and rotation invariant moments ($I_1, I_2, I_3, I_4, I_5, I_6, I_7$). We already used this approach in Section 4.4.5 to refine the fingertips' location.

First we compute the barycenter($\bar{x}, \bar{y}$) as follows:
$\bar{x} = \frac{M_{10}}{M_{00}}, \bar{y} = \frac{M_{01}}{M_{00}}$ with $M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$

Then we compute each central moment as follows:
$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$

We need scale invariant moments thus we normalize the central moments as follows:
$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}}$

Now we can compute our seven moments as:

$I_1 = \eta_{20} + \eta_{02}$

$I_2 = (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2$

$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$

$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$

$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$

$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$

$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$.

Those seven moments are saved in a database and will be compared with the study symbol's moments to see the similarity between them and the stored moments, to determine whether or not they are the same symbols.

4.5.2 Active area:

Overview:

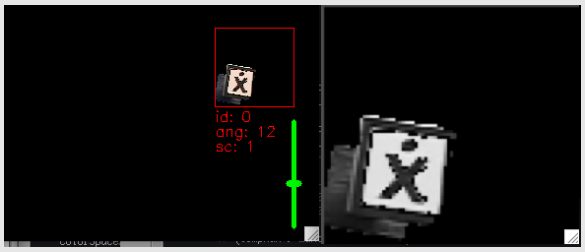


Figure 4.20: Active area process

Why using this method:

- Defining a specific area where to position the marker allows:

  More control for the children

  Less computation (only this area is checked every time)

Currently, we have implemented a three marker solution, computing their rotations, and their relative value is used by the system. Moreover, we decided to create an 'active area' in a corner where the markers have to be placed. A marker will be processed only in this area. This brings more control and requires less computation because only this area will be checked to detect a marker during the processing. In addition, we implemented a different type of marker allowing the users to control the options of the tangram system (i.e., selection of the mode, ...).

4.5.3 Marker detection:

Overview:

Figure 4.21: Example of marker

Why using this method:

- Each marker has a different meaning (rotation, translation, mode selection, ...) therefore a recognition is needed
- The parameter mapped to the marker is proportional to the markers orientation

Every time we need to identify a symbol, we compute the seven moments and compute the sum of the difference between them and each model in the database as follows:

$$Diff(C_1, C_2) = \sum_{i=1}^{7} |I_i^{C_1} - I_i^{C_2}|$$

with C_1 the unknown contour, C_2 the model, $I_i^{C_1}$ the i^{th} moment of contour C_1, and $I_i^{C_2}$ the i^{th} moment of contour C_2 as defined above.

This solution provides good results, but it must be noted that some shapes' moments are too close to one another to distinguish them easily. This is why we tested three shapes to be sure they are correctly identified (see Figure 4.18).

4.6 OpenGL and Physics Engine

The top layer of this project aims at displaying the data the low layer provided. Indeed, this top layer consists of an OpenGL interface to draw the shadow tangrams, and a physics engine called Pymunk (the python interface to the C physics engine Chipmunk). This layer was developed in Python for simplicity. We will not extend this part because its not the principal topic of this thesis and does not contribute in our specific field. However, the technique used for communication between the low layer and top layer were the subject to a publication at ICMI'08 [52]. Chreston Miller, a PhD student at Virginia Tech is in charge of this top layer.

4.6.1 Game and design:

This top layer includes a game design to help study the children's mathematical reasoning. This game or puzzle asks the user to manipulate physical or virtual tangrams to specific positions by using the system. Manipulation constraints can be applied (i.e a child can only move an object along the X coordinate whereas another child controls the two other *DOF*, Y and Rotation) as well as specific rules (i.e a child can only manipulate green objects whereas another one can manipulate red ones).

4.6.2 Mapping between the camera image and the Tabletop Surface:

This section describes the algorithm used to map the camera frame to the tabletop surface. This 'bridge' between the low layer and top level was designed in the low layer part and later on integrated to the top layer for more control.

Indeed, the low layer provides tangram locations that must be mapped into the tabletop coordinate plane to make the shadow position itself exactly below its real counterpart.

We need a plane homography (see [86] and [2] for more detail) that maps the plane 'camera image' to the plane 'tabletop surface'. The algorithm is illustred Table 4.5.

Table 4.5: Linear Homography function

Lets have 4 correspondences $p_i^{\pi_1} \longleftrightarrow p_i^{\pi_2}$ with $p_i^{\pi_j} = (x_i^{\pi_j}, y_i^{\pi_j}, w_i^{\pi_j})$ the homogeneous coordinate of the i^{th} 2D point $p_i = (x_i, y_i)$ in the plane image π_j.

We look at defining the 3x3 Homography matrix Hsuch as:

$\forall i \in [1, ..., 4], \exists H = [h^{1T} h^{2T} h^{3T}]$ such as $p_i^{\pi_2} = H p_i^{\pi_1}$

We can express this condition in terms of vector products as:

$p_i^{\pi_2} \times H(p_i^{\pi_1}) = 0$

$$\begin{bmatrix} 0 & -w_i^{\pi_2}(p_i^{\pi_2})^T & -y_i^{\pi_2}(p_i^{\pi_2})^T \\ w_i^{\pi_2}(p_i^{\pi_2})^T & 0 & x_i^{\pi_2}(p_i^{\pi_2})^T \\ -y_i^{\pi_1}(p_i^{\pi_2})^T & x_i^{\pi_2}(p_i^{\pi_2})^T & 0 \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

We have 3 equations for each couple of points, but only two of them are linearly independent. Thus, for each couple of points, we obtained 2 equations. The homography matrix is constituted by 9 unknowns but is defined up to a scale so the number of *DOF* is 8. This is why 4 correspondences, will generate 8 equations for 8 unknowns.

However, the constraint $p_i^{\pi_2} = H p_i^{\pi_1}$ is not always respected because of noise during measurement. This is why more points and a minimal least-square estimation is usually preferred (see [2] for very clear description of this method).

By solving those equations, we obtained the homography matrix mapping points from one plane to another. This method was implemented for the high level layer and allows us to display correctly the shadow image.

4.7 Conclusion

We presented a full implementation of a tabletop design strongly relying on Image Processing algorithms. Three modes are provided, a Physical manipulation mode where the users manipulate tangible objects and the system tracked them, a Virtual manipulation mode where the users manip-

ulate virtual objects and the system track fingertips, and an Parametric manipulation mode where the users are limited to the use of a tangible ‘go-between’ device to interact with the virtual objects.

Those three modes study children’s mathematical reasoning by designing games and puzzles. As we showed, our low layer is flexible and allows multiple manipulations and interactions, which would be difficult to implement with a hardware-based solution.

However, the Virtual manipulation mode is still limited by the usage of a single camera above the surface. It limits the fingertips’ location in a plane (no depth information). To tackle this problem, we implemented another approach called MirrorTrack using two side-mounted cameras to provide 3D fingertips location.

Chapter 5

Design of 'MirrorTrack' software

> On the other hand, you have different fingers.
>
> Steven Wright

This chapter is constituted by two papers that have been respectively published or submitted to major conferences: ICMI'09 and ICPR'10.

The first paper *"MirrorTrack - Tracking with Reflection - Comparison with Top-Down Approach"* has been accepted at ICMI'09 held at Boston in September 2009 [73]. The idea was studied previously by HCI students who published two papers [15, 14]. However, this new paper presents an innovative approach of the idea using a completely new implementation and technique.

In this paper, we compared our implementation of MirrorTrack to a simple top-down approach for fingertips tracking and showed improved results in 3D fingertips location. It has a lot of potential because of its new interaction possibilities (such as Gesture recognition above the table, hovering, ...).

The second paper *"Study of Camera Position Above a Reflective Surface - Area of Confidence"* has been submitted at ICPR'10. The idea is to mathematically (as opposed to empirically) deter-

mine the position range where the two cameras used in MirrorTrack provide the best result. We quantify a good result as the best reflection ratio (see definition of this term in section 5.2.2) while minimizing the triangulation error and respecting *FOV* constraints (see section 5.2.5).

We showed that a common position range could be defined for each screen whereas the LCD reflection behavior could be approximated by tuning a BRDF function.

5.1 ICMI-MLMI'09

In this paper, we implemented from scratch a new version of MirrorTrack based on the idea of [15, 14]. This new implementation based on a pyramidal approach is compared to a top-down approach and showed similar accuracy in term of fingertips location and better click as well as hovering detection. The flow diagram is illustrated Figure 5.3.

The following sections are based on the paper published at ICMI-MLMI'09 (long version):

> Y. Verdie, B. Fang, and F. Quek. Mirrortrack: tracking with reflection - comparison with top-down approach. In ICMI-MLMI 09: Proceedings of the 2009 international conference on Multimodal interfaces, pages 347350, New York, NY, USA, 2009. ACM.

Tabletop hand tracking techniques have evolved much during the last few years from single to multiple cameras, offering users an improved interactive experience. MirrorTrack is one of such techniques. This paper demonstrates the comparison of accuracy between MirrorTrack and top-down approach, which is generally used for table top tasks. In this paper, we focus on the comparison of distance errors in finger trajectory, and clicking errors by manual monitoring.

5.1.1 Introduction:

Tabletop or surface interaction has garnered significant interest as a display and interaction configuration. The horizontal surface morphology of such systems are well-suited for touch and

multi-touch interaction [77, 10, 28, 19]. Touch interaction requires detection and tracking of surface touch behavior of the human. The technologies for such detection may be generally divided into video-based and non-video-based.We consider video-based solutions more promising because their are less expensive to implement whereas non-camera solutions need more expensive technologies. This paper compares two video-based touch detection and tracking approaches differentiated by the placement of the camera(s), and the concomitant video processing approaches. Our approaches are motivated by the observation that the typical back-projected screen technologies with rear-positioned cameras are incompatible with the obvious display technologies involving flat panels like LCDs and plasma displays that can greatly reduce the cost and increase broader adoption of tabletop interaction systems.

The approaches we discuss in this paper are: 1. A top-down camera placement model where the camera is placed above the screen with the optical axis perpendicular to the display surface [49, 44, 59], and 2. A 'MirrorTrack' approach where multiple cameras are placed close to the display surface with their optical axes making sharply acute angles with the surface plane [15, 14].

5.1.2 Top-down approach:

An obvious camera placement approach consists in having a camera above the display surface with its optical axis perpendicular to the display surface. In this configuration, the rectangular surface space maps directly into the camera image along with advantages of minimal perspective distortion and a maximization of the useful resolution of the camera. Top-down has advantage of near orthonormal projection with even pixel distribution similar to back projection camera system. Top-down approach was developed in accordance with previous contributions in this domain such as [49, 44, 59].

Hand segmentation

Hand segmentation aims at extracting the hands region from the frame. We employ a two-step process to realize this. One of the critical challenges to detection of the hand over the glossy display surface is to minimize the effect of the reflection and shadow of the hand in the image. Employing a HSV color space, we notice that H and S are very sensitive to the color content of the hand that is preserved in the reflection. To minimize this effect, at the initial stage of processing, we employ an image differencing approach in V space alone. This approach has advantages to using the commonly-used RGB space because the normalized (r, g) are precisely sensitive to the reflected color [44]. This processing step effectively segments the silhouette of the hand from the background.

Once the silhouette is obtained, we reverse the color space process and employ H and S because this is now most sensitive to the skin color. This will remove any shadow boundary that were inadvertently included in the segmented silhouette. This also helps to remove other moving objects (e.g., screen objects) that are not close to the color of the hand [72].

Fingertips location

Various techniques such as *k-curvature* , *convex hull*, *circle Hough transform*, *shape filtering* or *template matching* [49, 44] have been used to find fingertips in computer vision. Our experiments with the tabletop shows that curvature based approaches can suffer from significant jitter because of the derivative effect in digital images. This effect is less pronounced with the latter two techniques cited.

We choose to use a morphological template matching approach using a circle as a structuring element, because it is an efficient region processing algorithm. Moreover, it gives good results for scale invariant targets such a hand over a table top (we observe that in this projection, finger projections may be approximated as cylinders, and the fingertip as a half-circle). Given the camera configuration, with the standoff-distance far exceeds the depth volume in which the finger resides.

Under this condition, the size of the finger in the image is approximately constant.

Tracking and smoothing

The tracking of each fingertip is done by a polynomial curve fitting over five previous fingertips location. In this way, we also smooth the trajectory of each fingertip to continue to attenuate the jitter.

5.1.3 MirrorTrack:

In the MirrorTrack approach, multiple cameras are placed at low azimuth angles with respect to the surface plane. The rationale for this is that given the specular nature of the display surface, it approximates a perfect mirror from the camera viewpoint in this configuration. Hence both the image of a hand or finger hovering close to the display surface, and its clean reflection can be detected in the video. This has two other key advantages. First, it eliminates much of the effect of the image in the video display. Second, it eliminates the specular reflections of typical overhead lighting common in most office, home, and school environments. Furthermore, given the perspective effects at the low azimuth, it is relatively easy to determine if a stereo ray intersection takes place over the surface or behind it [14, 15].

The MirrorTrack algorithm is composed of three main steps: Stereo Calibration, 2D Processing and 3D Processing.

Stereo calibration

We use Tsai's calibration model [70] with a set of 48 control points to calibrate the cameras [14]. The measured accuracy of our calibration algorithm on a 640×480 picture is around 0.2in which suffices for our purpose.

2D Processing - Pyramidal approach

Overview:

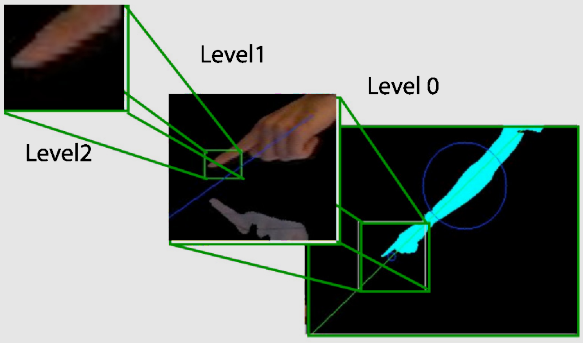


Figure 5.1: Pyramidal approach of MirrorTrack algorithm.

Why using this method:

- Improves the performance
- Allows real time processing even with two cameras
- Technical details:

  Each level in the pyramid defines a new region of interest for the next level. Each level has a resolution two times better than the previous level

We implement MirrorTrack as a sequential pyramidal process in three levels as shown in Figure 5.1.

Level 0 At this level, we resize our native image by an order of 2. A first rough estimate of our hand location can thus be found. We are using the hand segmentation introduced in section 5.1.2 that can segment the hand without its shadow and reflection. Then we approximate the blob as an ellipse to compute its spatial moment as described in [83]. The center of our *region of interest* (ROI) is finally defined as the intersection between the line generated from the major axis of the ellipse and the horizontal line passing through the lowest point of the blob.

Level 1 We take a higher resolution sub-image centered around the ROI computed in Level 0. We empirically resize the sub-image to 160×160 to include the whole hand.

In opposition to the hand segmentation algorithm discussed in section 5.1.2, we want to keep the hand and its reflection for MirrorTrack, so we process the background segmentation in Hue channel. Indeed, the hue value of the hand and its reflection are very similar and it is easy to classify the hand, its reflection, the background and hand shadows in HSV space.

Since we now have the hand and its reflection, we can apply template matching to locate the fingertips. We want to clearly detect when the hand touches its reflection to be able to classify this action correctly. That is why the template used here is an ellipse with horizontal major axis, because this specific ellipse matches the shape formed by the intersection of the hand and its reflection. We then threshold our probability map in order to reveal the fingertips location that are used for Level 2.

Level 2 This step uses the candidate coordinates found in the previous level to re-calculate a new ROI, and reset the coordinate values by looking at fingertips location with an edge operator.

Classification

Overview:

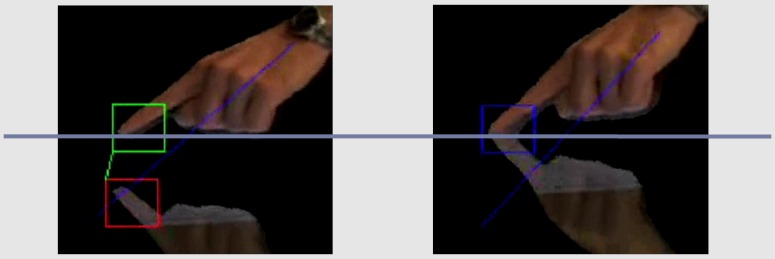


Figure 5.2: Example of classification.

Why using this method:

- Eases the 3D computation by eliminating some pairing cases.
- Technical details:

  In the hands region of interest, splits the area in the middle.

  Classifications:

  [Green]: 2D candidate above the line is a real fingertips

  [Red]: 2D candidate bellow the line is a fingertips reflection

  [Blue]: 2D candidate close to the line (and one blob) is a click

Classification aims at determining the interactive action (*hover*, *click*, *move*) using the location of the fingertips with the image configuration we detected.

- Fingertip: The fingertip above the surface is detected in the upper half image.
- Reflection: The fingertip 'below' the surface is detected in the lower half image.
- Click: The fingertip and its reflection is detected to be merged together.

As we discuss in section 5.1.4, this classification increases the accuracy of the system because more information is available in defining whether or not the finger touches the surface.

3D Processing

Triangulation We use the calibration information and the data from the 2D processing to compute 3D locations of the fingertips. Since we have three types of fingertip candidates, if we can combine them with the same classification (see section 5.1.3), we compute the 3D locations with the same type; otherwise, we just calculate all the possible 3D positions without considering the classification. This helps us to reduce the computational cost.

Tracking and smoothness At this point, we have a set of all the possible 3D locations, and we prune it by looking at the closest candidate with the estimate position of our target. The estimation employs a polynomial curve fitting of 5 previous points to track the target in 3D space. Meanwhile, the estimation helps to smooth the trajectory of the target.

5.1.4 Experimental results:

We will show that MirrorTracks accuracy is very close to the accuracy of a top-down tabletop single camera approach while providing hovering and touching information emending the main weakness of a top-down setup. We compare the top-down approach and MirrorTrack approach for distance and click accuracy. Moreover, we also compare the MirrorTrack approach with three setups: 1. We set the low-azimuth to have best reflection. 2. We set the azimuth a little higher to have better depth resolution, but less reflection. 3. We set a high azimuth to have best depth resolution, but the worst reflection.

Description of the setup

We tested MirrorTrack with three different angles to study the influence of the angle over the accuracy and the capacity for this algorithm to take advantage of the reflection when available.

We ran three experiments by varying the height of the camera K to 25in, 30in and 35in, and the

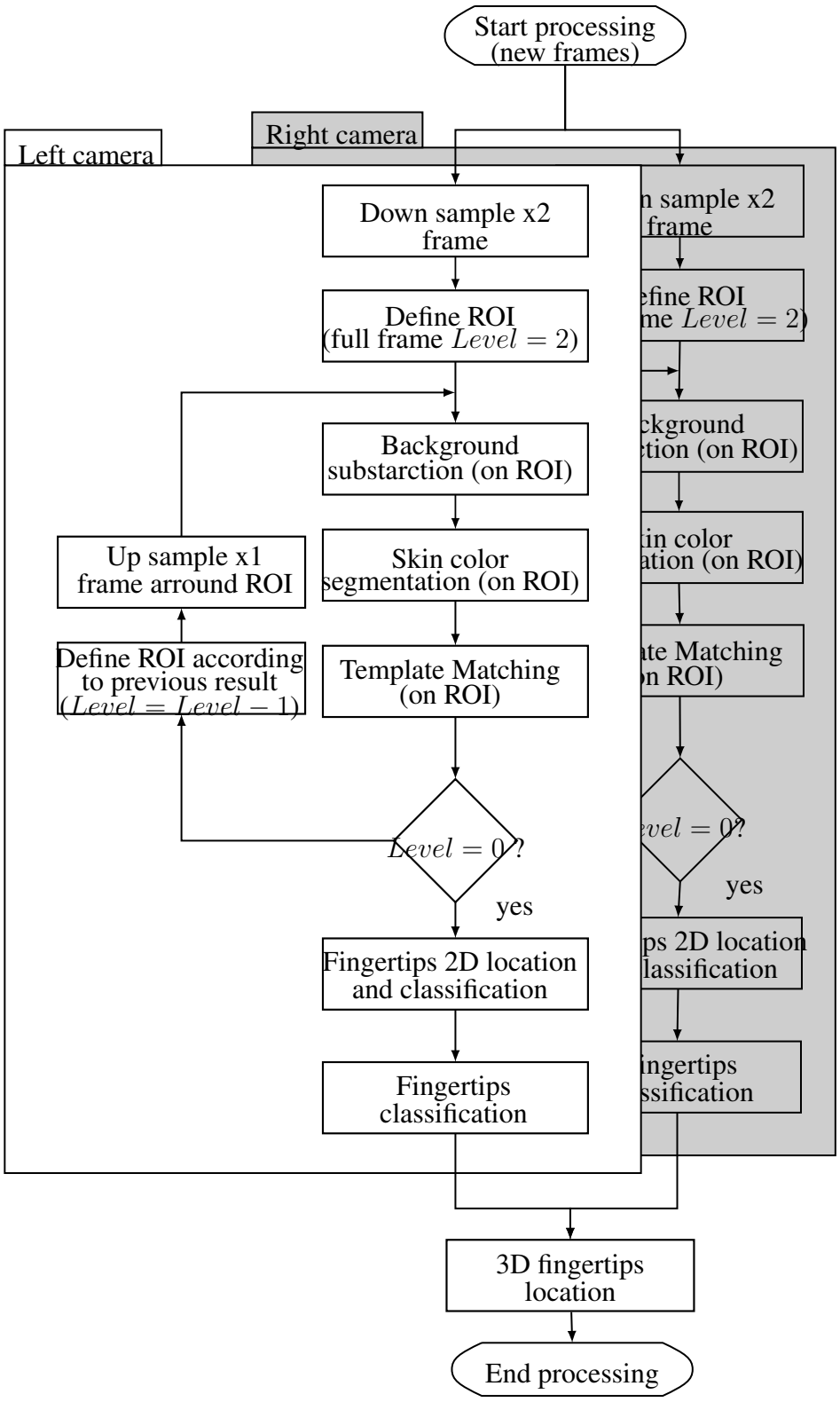


Figure 5.3: Flow diagram of the processes in 'MirrorTrack Project'.

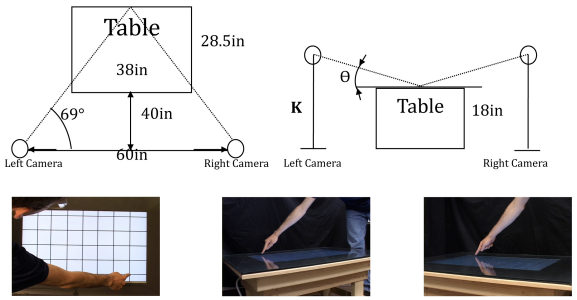


Figure 5.4: Camera setup for the MirrorTrack algorithm. Upper left: top down view of the setup, Upper right : frontal view of the setup with K varying from 25in to 35in. Lower from left to right: top-down camera, MirrorTrack left camera and MirrorTrack right camera.

setting is shown in Fig. 5.4. The corresponding angles (θ) are $13°$, $22°$ and $30°$.

Results

In our experiment, we display a grid in the screen embedded in the table. We ask the user to move his finger across all the grid intersections on the display, and capture the motion for top-down and MirrorTrack cameras at the same time. Then, we use the tracking result from top-down camera as ground truth, and calculate the offset errors of the tracking results. Then, we label the real click by observing the original video clips, and use the labeled result as ground truth to compare the 'clicking' correctness for both top-down and MirrorTrack approaches.

Distance error Before we can compare the distance error between top-down and MirrorTrack approaches, we need to map the both tracking trajectories into the same coordinates. As the resolution of each camera is 640×480, we use the same resolution for mapping. After mapping them into the same coordinate, we calculate the Euclidian distance between them. The following figures show the results of distance error by different settings (Fig. 5.5 shows the distance error when cameras are set as $22°$, Fig. 5.6 shows the distance error when cameras are set as $13°$).

Click error In order to evaluate the capacity of MirrorTrack to differentiate hovering to clicking, we compare top-down and MirrorTrack to a ground truth manually selected from video. Fig. 5.7

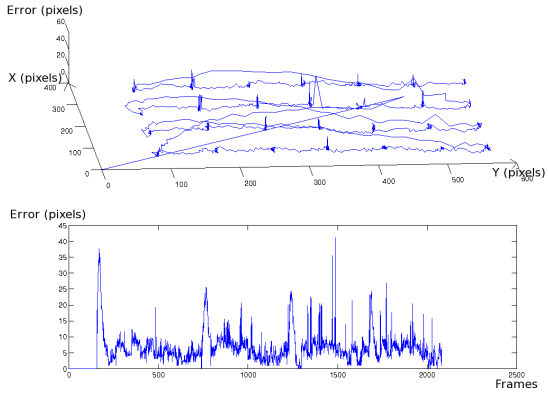


Figure 5.5: Distance error when camera angle is equal to $22°$. The upper figure shows the error by mapping coordinates. The lower one shows the error by frame.

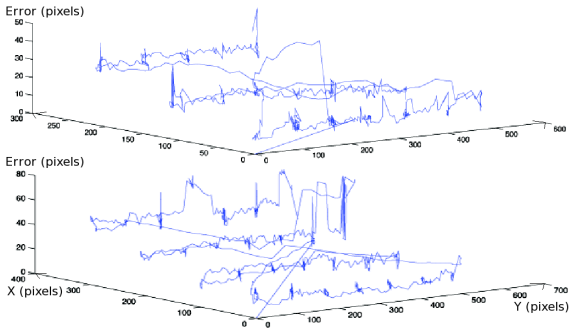


Figure 5.6: Left and right figures: Distance error when camera angle is equal to respectively $13°$ and $30°$.

shows the click detection for the whole captured video, and Fig. 5.8 shows one of the click for insight detail.

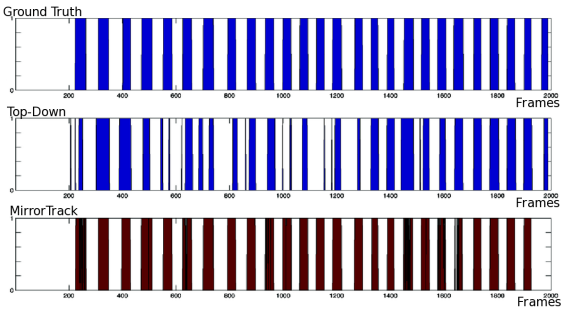


Figure 5.7: Click error when camera angle is equal to $22°$. The y coordinate is equal to 1 when clicking happens, and 0 otherwise.

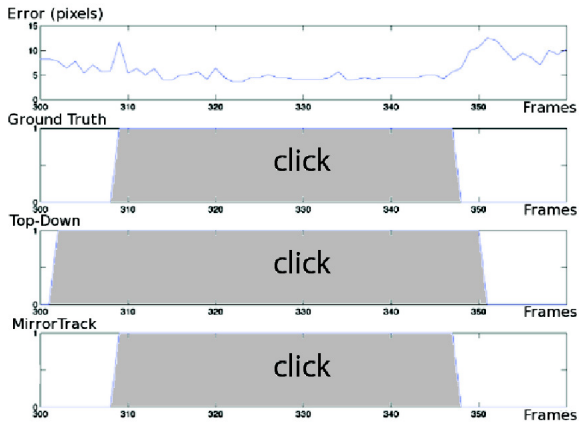


Figure 5.8: Click error when camera angle is equal to $22°$. The y coordinate (except the 1st one) is equal to 1 when clicking happens, and 0 otherwise.

5.1.5 Discussion:

As shown in Fig. 5.5, the upper figure shows the error (z-axis) by mapping corresponding top-town and MirrorTracking tracking results into the same coordinates (640×480) when cameras' angles are set to be $22°$. As we can observe, peaks appears around where clicking happens. Fig. 5.8 shows a randomly selected click event. From the top two figures, we can observe that the peaks actually happened before and after clicking action. This is due to the polynomial curve fitting we used to smooth the trajectory and estimation. Indeed, based on the history data, the trajectory before the 'click' is predicted as continuing to move down whereas the trajectory predicted when the 'click' is released is static. These artefacts generate those peaks, but do not affect general effectiveness of the system. Moreover, we can observe from Fig. 5.5 that, the peaks last around 6 frames, which also prove our hypothesis. We also notice that the error is not correlated with the finger location over the table.

From Fig. 5.6, we demonstrate that the peaks happen no matter what angle of camera setting is used. However, we notice that the setting of camera affect the overall distance error. When cameras are set to $13°$, we increase the accuracy of detecting finger reflection, but lose the resolution for x-axis, thus, we lose accuracy on that axis on the whole. When cameras are set to $30°$, we increase

the resolution for x-axis, but loss the accuracy of detecting finger reflection. Therefore, the general distance error is worse.

We use Fig. 5.7 to show the 'click error' when cameras are set to $22°$. The top figure shows the 'real click' which is selected manually. Fig. 5.8 shows one of these clicks. From these figures, we can observe that MirrorTrack has a more accurate result for click than top-down approach, and it is close to the 'real' clicking. Even though top-down results may be improved by tunning the predetermined rules to detect a 'click', this approach will still be limited per se because it does not detect the 'real' click. In opposition, MirrorTrack detects 'clicking' by detecting the finger and its reflection to produce more accurate results, which are independent from any predetermined rules.

Finally, the accuracy of our prototype can still be improved for some reasons: we use template matching in 3D space by approximating the movement of the hand to be in a plane. While this assumption is appropriate for the top-down approach, it introduces errors in MirrorTrack implementation because of perspective effects. It could be relevant to use the convex-hull of the hand and its abduction angle of the fingers to detect fingertips because this method is scale invariant. Another solution could be using shape filtering and update the criteria depending on the size of the blob for scale invariance. Moreover, the step described in section 5.1.3 was not fully implemented during the experiment and could significantly improve the fingertips location to allow us estimating better results.

5.1.6 Conclusion:

We presented a novel technique for hand devices interaction that allows new action by using hand's reflection. We have shown that the accuracy of our results with MirrorTrack is similar to a top-down setup while the touch action is detected and not modeled by predetermined rules. We also studied the influence of the angle on the system to find a trade-off between capacity to detect reflection and accuracy during the triangulation.

Using MirrorTrack on flat LCD screen for day-to-day usage seems to be a very promising and

attractive solution to enhance interaction with devices. It supports hover mode that has been identified as desirable. Future work will be conducted on the usability of the system with various users.

5.2 ICPR2010

In this paper, we answered the question of trade-off between reflection quality and triangulation constraints discussed in the paper presented earlier. We also remarked that the reflection screen behavior could be approximated by a BRDF curve.

The following sections are based on the paper submitted at ICPR2010:

> Y. Verdie, J. Anurodh, and F. Quek. Study of Camera Position Above a Reflective Surface - Area of Confidence. In ICPR2010

Tabletop interactive surface has garnered much interest in HCI. MirrorTrack is a technique for tabletop surfaces that uses two cameras to overcome the limits of the topdown approach. It relies on capturing the reflection of user's hand to detect interaction. This paper conducts an objective study of the factors affecting the quality of reflection in order to optimize tracking. The goal is to maximize the reflection quality while minimizing the error during triangulation. We also discuss other constraints which ensure that the reflection and the finger are always within the FOV. We define an 'Area of confidence' for the camera locations that respects these constraints.

5.2.1 Introduction:

Tabletop interaction is a growing area of research as it offers user an improved interactive experience [10]. These systems rely on touch detection for user interaction. It has been shown that a 'hovering mode' can improve interactivity of pen or touch-based interaction [26]. Our MirrorTrack approach can distinguish touch from hovering by capturing both the user's finger and its reflection

at a low viewing azimuth such that the screen approximates a reflective surface. However, lowering the cameras decreases their *Field Of View* (*FOV*), and the cameras lose important information of the display surface. Moreover, depending upon the required *FOV*, there is a limitation to how low the cameras can be placed to capture a reflection that can be correctly processed. In Section 5.2.2 we shall describe our experimental setup. We discuss the models of light scattering in relation to MirrorTrack in Section 5.2.3, and the corresponding factors that affect reflection quality in Section 5.2.4. In Section 5.2.5 we detail the constraints relating to triangulation error, and the relationship between camera placement and the elevation of the hovering finger. In Section 5.2.6, we discuss the key goal of our paper, which is to derive a summary *'Area of Confidence' (AC)* measure that takes our findings concerning the reflection models and operational constraints into consideration.

5.2.2 Experimental Setup:

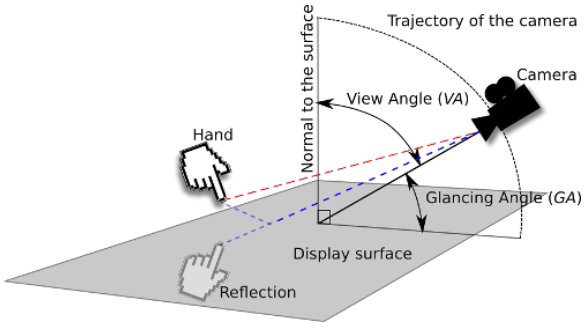


Figure 5.9: Experimental Setup

Figure 5.9 shows the setup for our experiments. A 3CCD camera captures images of the user's hand and its reflection on the screen at *Glancing Angles* (*GA*) 0-90 degrees. We refer to the complement of this angle as *Viewing Angle* (*VA*). The data were collected under multiple sources of light to emulate day-to-day laboratory conditions. We assume that the incident light is white and unpolarized.

For generality, we tested our MirrorTrack approach on four different monitors: a 30" Apple Cinema

HD with matte surface (A), a glossy 21.5" iMac (B), a Samsung SSO3 TV (C), and a 20" Apple Cinema Display with glossy surface (D). In addition, we tested each with a protective glass sheet (designated pg, such that A-pg means monitor A with glass). This gives us a total of 8 material surfaces. We quantify the quality of the reflection by measuring the ratio of the average pixel value of the target reflection to the average pixel value of the target. We will refer to this ratio as the *Reflection Ratio* (*RR*). We prefer a high *RR* to simplify processing of the reflection. We also define a Reflection Ratio Constraint as the minimum *RR* required to correctly process the reflection.

5.2.3 Models for surface light scattering:

The behavior of a beam of light incident on a smooth surface can be estimated using the Fresnel equation [62]. Surfaces such as LCD displays appear visibly smooth but have micro-roughness and other impurities. Thus, the light scattered from these surfaces cannot completely be predicted using the Fresnel model. The Bidirectional Reflectance Distribution Function (BRDF) can describe light scattering on a rough surface [23]. We compare the data collected with these models for two reasons. First, we expect the display to approximate a smooth reflective surface especially at low *GA*s, so it will be interesting to compare the collected data against the Fresnel model. Second, assuming that the display has no impurities, only micro-roughness of the screen should contribute to the diffuse scattering. Thus, a Micro-roughness BRDF model should give a better approximation of the light scattering from the display surface [24], especially as the *GA* increases. Understanding the representational power of these models to the real MirrorTrack scenario will allow them to serve as theoretical guides for future design, adapting to different monitors. Indeed, by varying models' parameters, we approximate the curves obtained during data collection and avoid the need to collect data every time another monitor is used.

Figure 5.10 shows a comparison between the *RR* curve as predicted by the Fresnel and the BRDF models against data collected during our experiments for our 4 basic material surfaces. BRDF allows us to tune the surface characteristics to give a better approximation of each surface. We use a micro-roughness BRDF model as defined in [24] and tune the refractive index as well as the

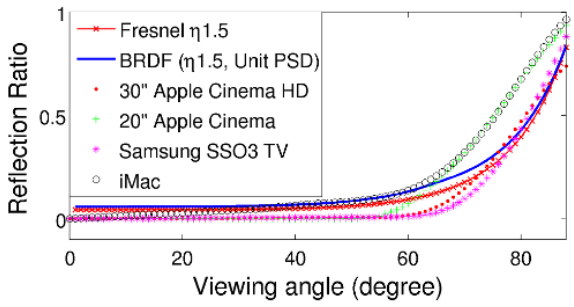


Figure 5.10: Fresnel and BRDF models

surface *Power Spectral Density* function (*PSD*). This model better describes the real reflectance of typical displays as it accounts for the diffusion on the surface.

BRDF and the surface's PSD S(f) (ABC model) are given by:

$$\text{BRDF} = \frac{dI_s}{I_i(d\Omega \cdot cos(\theta_s)}, S(f) = \frac{A}{(1+(Bf)^2)^{\frac{C}{2}}}$$

For a micro-roughness surface (see [24]), we have:

$$\text{BRDF}_{topo} = \frac{16\pi^2}{\lambda^4} \cdot cos(\theta_i)cos(\theta_s) \cdot \mathbf{S}(f) \times \|q_{ij}^{topo} \cdot \hat{e}\|^2$$

The parameters describing the refractive index with micro-roughness is embedded in the q_{ij}^{topo} matrix (see eq. 27 in [24]), and $\mathbf{S}(f)$ models the *PSD*.

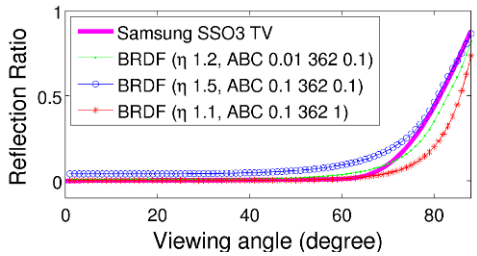


Figure 5.11: BRDF with varying refractive indices and PSDs

In Figure 5.11 we show a comparison of display C against BRDF models with different refractive indices and *PSD*s, showing that an (index, *PSD*) pair of (1.2, ABC(0.01 362 0.1)) approximates

the empirical surface.

5.2.4 Optimizing Reflection:

We investigate two main factors that affect the reflection obtained: 1. The camera, and 2. The Display properties. We use *RR* to measure the quality of reflection. The goal is to optimize the *RR* by optimizing individual factors.

Sensor characteristics (aperture)

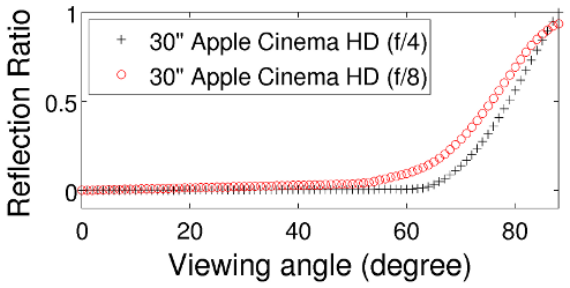


Figure 5.12: RR curves for f/4 and f/8

To determine if the sensor characteristics can be tuned to obtain a better reflection, we regulate the amount of light coming into the camera by varying the aperture. We choose two apertures f/4 and f/8 for data collection from various screen models. Figure 5.12 shows the *RR* curves for screen A with varying aperture. The reflections for f/8 are clearly better than for f/4 (higher *RR*). This is to be expected, since the aperture has to stay open longer for the smaller aperture, leading to better noise tolerance through the averaging effect. The trade-off, obviously is that longer exposures also result in more motion blur if the hand moves quickly.

Display properties

In this section we investigate if the Display properties can be used to optimize the *RR*. We consider three variables: 1. Screen model, 2. Background color of the Display, and 3. Glass addition on top

of the screen.

To test our approach across ***Screen Models***, we compare surfaces A, B, C, and D.

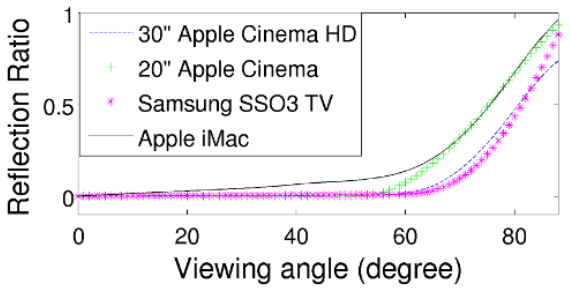


Figure 5.13: Results obtained for all screens

Figure 5.13 shows that the iMac monitor (B) and the glossy Apple monitor (D) produced better reflection results than the other two display surfaces. This confirms our intuition that glossy surface is closer in its behavior to smooth surface (mirror) and as such produces better reflection quality. We conclude that using glossy surface seems to enhances the quality of reflection.

To study the effect of ***background color of the display*** on the *RR*, we selected surface A and used three different background colors - blue, red and, green.

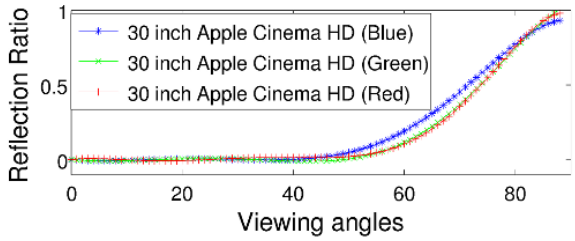


Figure 5.14: RR for different backgrounds

Figure 5.14 shows that the *RR* curves obtained for the different backgrounds are not significantly different. We conclude that background color has no significant affect on the reflection quality.

To determine if adding a ***reflective surface on top of the screen*** can improve the *RR* we used six surfaces A-pg, A, B-pg, B, C-pg, and C.

As expected, adding a smooth surface on top of the screen improves the reflection quality and thus

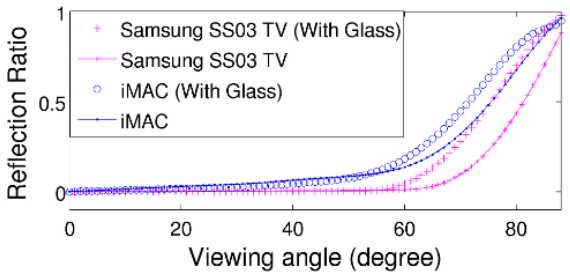


Figure 5.15: Glass addition on top of a screen

can enhance the system performance.

In a nutshell, we demonstrated that using small aperture, adding a glass on top of the display, and using glossy monitors improve the *RR*.

5.2.5 Triangulation and hovering constraints:

There are two other factors that constrain the *VA* for camera placement. These constraints arise from the need to minimize triangulation error and to ensure the finger and its reflection to stay within the *FOV*.

We study the effect of camera positions on ***triangulation error*** using the *mid-point method* [30]. The error for each *VA* is calculated as the distance between the target and the mid-point of the two projected lines.

The maximum error occurs when the two projected lines are parallel to each other. Moreover, triangulation error increases with decrease in the angle between the cameras. Figure 5.16 shows that the triangulation error increases with decreasing *VA*.

Previously only the target position was taken into account to determine the camera placement. However, we need to consider both the target position and its reflection on the display; this adds further constraints for camera placement. It is imperative to capture the reflection of the target along with the target position; we will refer to this constraint as the *Hovering constraint* (*HC*).

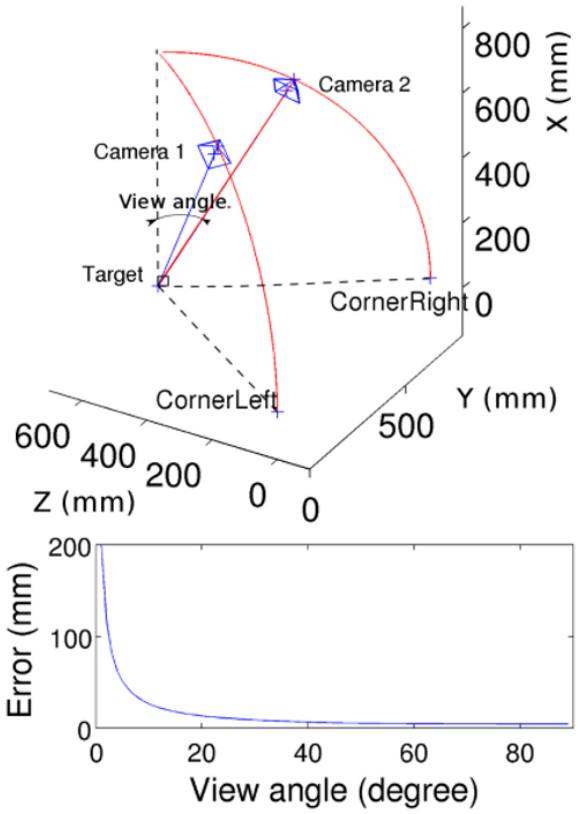


Figure 5.16: Top: Mid-point Triangulation, Bottom: Maximum error by varying θ.

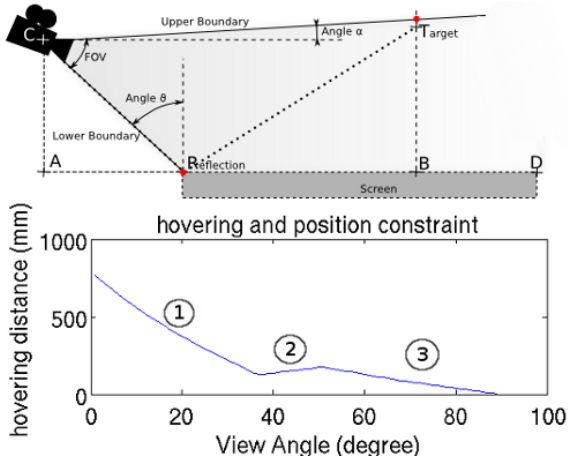


Figure 5.17: Hovering constraints curve

We define a *Lower Boundary* (LB) for the FOV:

$$\text{LB} = Max(\|\overrightarrow{RC}\|, Const.) \text{ with} \tag{5.1}$$

$$\|\overrightarrow{RC}\| = \frac{tan(\alpha) \cdot \|\overrightarrow{RD}\|}{cos(\theta) - tan(\alpha) \cdot sin(\theta)}, \alpha = \text{FOV} + \theta - \frac{\pi}{2}$$

Hence, we can express the FOV *Upper Boundary* (UB):

$$\text{UB} = \text{LB} \cdot sin(\theta) \cdot tan(\alpha) \cdot (\text{LB} \cdot cos(\theta) + \|\overrightarrow{RB}\|) \tag{5.2}$$

From equation 5.1 and 5.2, we have

$$\boxed{hovering(\theta) = Min(\text{LB} \cdot tan(\theta)^{-1}, \text{UB})} \tag{5.3}$$

Equation 5.3 provides us an upper boundary for the *HC* respecting the following rule: the reflection, the screen, and the finger are always within the camera's FOV. Figure 5.17 shows a curve with these constraints. ① is because of the screen constraint, ② due to finger constraint and ③ due to the reflection constraint. This curve represents the maximum finger elevation with respect to *VA*.

5.2.6 Area of Confidence (*AC*) - Discussion:

The two previous sections define three ranges (for *VA*) based on the *RR* constraint, Triangulation constraint, and the *HC*. The first two constraints favor higher *VA*s. The third constraint ensures that the reflection is on the display surface and favors lower *VA*s. By using the intersection of these ranges, we can obtain a range producing a reflection that can be correctly processed and yet has tolerable triangulation error with capacity to detect reflection at reasonable hovering distance. Figure 5.18 shows the *AC* for surface B-pg obtained by imposing the *RR* constraint of 0.2 and a *HC* of 2" (this hovering distance was found reasonable).

The range defined by the *RR* is included within the range defined by the Triangulation constraint. Therefore, we use the intersection of *RR* range and the Hovering range solely to determine the *AC*. In figure 5.18, the first range on the left is defined by the *HC*; the designer decides when the system should start detecting the reflection. The second range on the right is defined by the *RR*

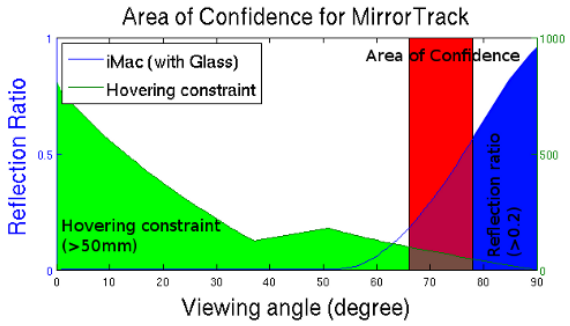


Figure 5.18: Area of confidence (*AC*)

constraint. By imposing these constraints, an *AC* of 61-78 degree was found for surface B-pg. The *AC* determined for the other surfaces are as follows: 72-78 degrees for A-pg, 68-78 degrees for C-pg, and 67-78 degrees for D-pg.

For all the screens used in our experiments, the *VA* preferred for camera placement overlap. In the future, more screens could be studied to see if we can generalize the results and define a common range for all kinds of screens. It might also be interesting to study if different types of glasses can be used to increase the *AC*.

5.2.7 Conclusion:

Through our research work, we were successfully able to determine the trade-off discussed in [73] between the need to detect a good reflection while keeping the reflection on the display surface. We also showed that BRDF model can be tuned to approximate display surfaces which precludes the need to collect data every time in order to determine the *AC*.

5.2.8 Acknowledgment:

The authors would like to thank Dr. T. Kubota (Susquehanna University) and Dr. T. A. Germer (National Institute of Standards and Technology) for their suggestions.

This research was supported by NSF grants ‘Interacting with the Embodied Mind,’ CRI-055610, and ‘Social Org, Learning Tech & discourse System: System Features for Facilitating Mathematical Reasoning In Pre K-3 Students.’ IIS-0736151.

Chapter 6

Merging MirrorTrack and Tangram

> Eventually, all things merge into one, and a river runs through it.
>
> Norman Maclean

Tangram and MirrorTrack both comport innovation in their own domain. Tangram is very flexible, not limited to a single design and can operate in real, virtual and augmented space. MirrorTrack provides a 3D hand location and gesture recognition and improves the location result by using a new technique relying on the location as well as the reflection of the target. However, they also have their own drawbacks. Tangram has a weakness in Virtual mode for manipulation via fingertips. MirrorTrack is strictly limited to hand tracking and cannot track objects. Therefore, merging them has a lot of potential.

6.1 The Goal

In this section, we propose the idea to merge Tangram and MirrorTrack together in order to create a unique and better system with the main advantages of both projects and minimizing or removing their respective drawbacks.

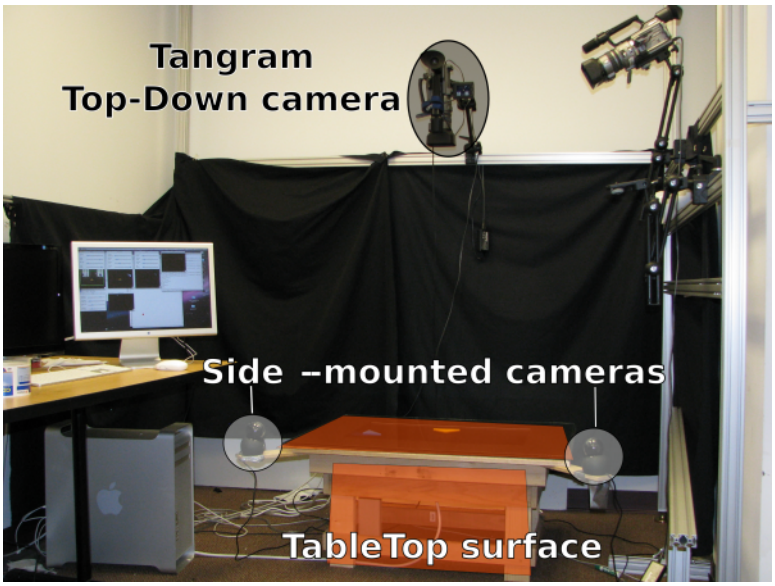


Figure 6.1: Tangram (Top-Down camera) and MirrorTrack (two sided mounted cameras) simultaneously

Indeed, we could use an overhead camera above the tabletop surface for the Real and Augmented manipulation mode as defined in the Tangram project whereas the Virtual manipulation mode could be provided by two sided mounted cameras and MirrorTrack.

Finally, this new setup design will be a tabletop surface with 3 cameras employing both MirrorTrack and Tangram together (see Figure 6.1). Some processing (like the background subtraction) are common to both systems and it would be judicious to completely merge the systems together (in opposition to run them in parallel) in order to be more CPU-efficient.

As a final solution, Tangram & MirrorTrack is a versatile and powerful system providing an interface for Augmented Reality and Augmented Virtuality modes.

The Augmented Reality mode *(AR)* is when Virtual objects are integrated into a real environment. Various research were conducted in this area [8]. For examples of *AR* in our design, we could decide that our system displays a GUI around the Marker laying on the surface during manipulation. This GUI will indicate the name of the marker as well as the real rotation degrees measured.

Moreover, a virtual line could be drawn on the surface between the Marker and the virtual entity currently manipulated.

The Augmented Virtuality mode *(AV)* is when Real objects are integrated into a virtual environment (as gauges, human inputs, ...) Various research were conducted in this area [55, 4]. For examples of *AV* in our design, each real tangible tangram laying on the surface has its virtual counterpart in the Virtual space. This space is made to 'simulate' a real world (mainly relying on a physical engine) but the behavior of this virtual entity can also be ruled by constraints such as velocity constraints, inertia enhancement (give an illusion of rough or slippery surface), inter-object magnetism, and so on.

This bipolarity $AR \Leftrightarrow AV$, provides an infinite possibility of interactions and creates powerful assessment tools for learning.

6.2 The Possible Future Improvements

Both systems (MirrorTrack and Tangram) are in their early stages. A first prototype for both has being released and we still need to conduct a user study. Moreover, we could list some possible improvements as a to-do list:

Tangrams:

- User detection [Difficult]: Segment the skin color independently. Because each skin color can be saved, we could try to segment the skin color independently (like what was done for tangram's colors) in order to give an I.D. to each user and thus to know who is using the system. (Note that this improvement is also possible on the MirrorTrack Project that should replace the fingertip detection of Tangram project after merging the two systems).
- Shapes segmentation [Medium]: Improve the segmentation when two or more shapes touch together. We could try to do the same as [7].

- Shapes location [Medium]: Improve the shape location by using a pyramidal approach as we did for MirrorTrack. This approach will strongly decrease the jitter. (for now 1 pixel of error on the frame creates around 6 pixels of error on the screen).

- Performance [Easy]: Increase the performance by implementing the high layer in C++. Indeed, the current implementation in Python is up to 60 time slower than a compiled code and the resources saved by switching to C++ may allow us to process the image in a higher resolution for example (and thus decreasing the jitter).

MirrorTrack:

- Multi-Touch multi-users system [Medium]: Test the behavior of the system with multi-touch and multi-users and improve it if necessary.

- Better triangulation [Difficult]: Use another technique for triangulation more convenient and potentially more interesting than the Tsai triangulation technique currently used. For example, Bouguet stereo calibration, epipolar lines and RANSAC could be used to improve the results [85].

Chapter 7

Technical Details

Science may set limits to knowledge, but should not set limits to imagination.

Bertrand Russell

7.1 Tangram: Technical summary

Technical overview:

- LCD flat 30” screen:
  Screen resolution - 2560x1600 (variable)
- 1 camera:
  Touch resolution - 320x240 for objects
  640x480 for fingertips
- Pro:
  Objects and fingertips’ detection.
  Colors and shapes’ recognition.
  Detection on and above the surface.
  High screen resolution and manageable-size setting.

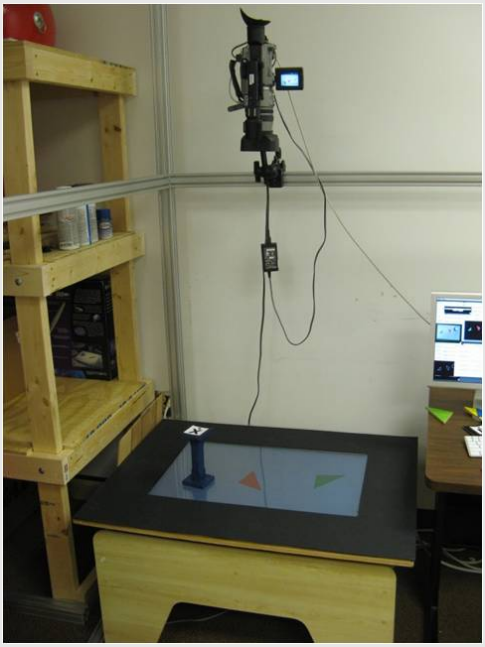
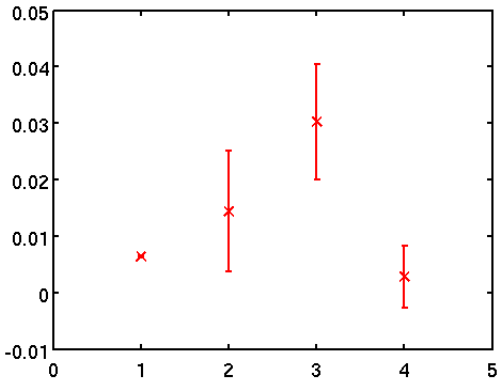


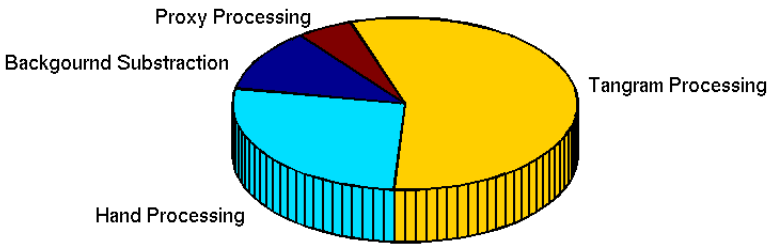
Figure 7.1: Setting for Tangram

- Cons:
  Less accurate, Less robust because it is limited by the camera resolution to meet the real-time requirement.

We measured the performance (time needed) for each major step of the system. The Figure (a) indicates the mean and standart deviation *(std)* for the different steps of processing used in the Tangram Project. The Figure (b) give the repartition in percent over the time to process one frame. For these measurments, we processed two objects with differents colors, two hands from the same side, and one marker.



(a) Means and std



(b) Means

Figure 7.2: ErrorBox (a) and pie graph(b): Mean and variance time for main Tangram's processes

On the Figure (a), the x axis is labeled as follow:

- The number 1 is the background substraction step that has a low std because the number of hands or tangrams does not affect the time of processing.
- The number 2 is the hands' processing step. The time of processing here depends of the number of hands on display which explains the high variance. Moreover, we note that about 45% of the time during this process is spent on the fingertips' refinement and kalman step.
- The number 3 is the tangrams' processing step. This step is the most computational expensive step and the time is corralated to the number of tangrams and the colors on display which explained why the variance is high. All the precesses inside this step have similar weight.
- The number 4 is the markers' processing step. This step is active only when a marker is inside the active area.

7.2 MirrorTrack: Technical summary

Technical overview:

Technical detail:

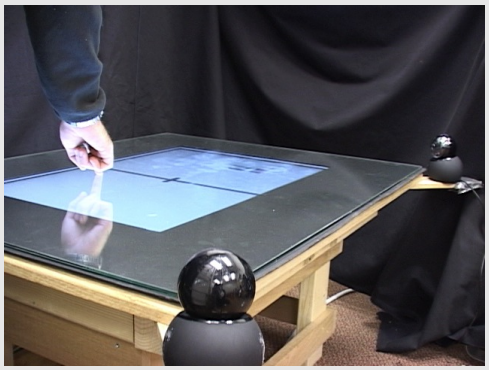


Figure 7.3: Setting for MirrorTrack

- LCD flat 30 screen:

  Screen resolution - 2560x1600 (variable)

- 2 cameras:

  Touch resolution - 5mm accuracy 5ppi

- Pro:

  3D fingertips location.

  Real touch detection, multi-touch, detects also above the surface

  Similar accuracy than top-down approach but better in click detection.

  Less affected by occlusion (because of the setting).

  Manageable-size setting (LCD screen and 2 cameras)

- Cons:

  No object detection yet.

  Compared to an hardware solution, MirrorTrack is

  Less robust.

  Accuracy inferior but sufficient (0.5 cm).

  Use more CPU resources.

Chapter 8

Conclusion

A conclusion is simply the place where someone got tired of thinking.

Arthur Block

8.1 Concluding Thoughts

We showed in this thesis the possibility to design fully software-based systems for *Surface Gesture & Object Tracking on Tabletop Devices*. Those two systems, Tangram and MirrorTrack, contribute in their own field of research to reduce the boundary between the reality and the virtual in Human-Computer interaction. Moreover, they offer an opportunity to improve the understanding of children reasoning.

Tangram Project offers three modes of interactions; Physical, Virtual, and Parametric mode. MirrorTrack project is an improvement of the Tangram's Virtual mode that is why merging them together brings a complete and more powerful new system. These two projects are each of them contributing in the design of new tabletop surface systems.

8.1.1 Toughts on Tangram:

Tangram is a Tangible User Interface using a top-down camera to detect, recognize, and track real plastic shapes (tangram) laying on the surface. Moreover, Tangram detects users' hands above the surface and tracks fingertips. It offers three modes of interaction: A Real manipulation mode where users manipulate physical tangrams and the system tracks them, a Virtual manipulation mode where users manipulate virtual tangrams displayed on the screen with the system tracking users' fingertips location, and finally, an Augmented manipulation mode where users manipulate real Markers to control the virtual tangrams.

Those three modes provide an adequate environment to study children's mathematical reasoning because of their flexibility. Indeed, the possibilities to control different parameters and set constraints and limits during the manipulation ease the understanding of early-age-reasoning.

To conclude, Tangram is a Software-based approach rather than a Hardware-based system, making the system extremely flexible, inexpensive to implement, with infinite possibilities to design new rules and assessment tools.

8.1.2 Toughts on MirrorTrack:

MirrorTrack is a Hand Tracking System using the reflection provided at low glancing angle by the reflective surface to detect, recognize, and track fingertips' location. Compared to a Top-Down system as Tangram, this approach provides a full 3D location of all the user's fingertips using stereo cameras with a similar result in term of location and better result in term of click detection (because the 3D location is known at each time). A study of the best position range to set up the cameras was conducted taking into account the reflection ratio, the error during triangulation, and respecting other constraints.

To conclude, MirrorTrack is a Software-based approach rather than a Hardware-based system, inexpensive to implement, with new interactive possibilities and gesture detection.

8.1.3 Toughts on the merging of Tangram & MirrorTrack:

Both systems together compensate one another for their weaknesses to form a unique and powerful new system for Object and Hand tracking above a surface. This improved system uses three cameras and allows a multitude of new possibilities and interactions.

8.1.4 New approaches:

There is a number of important areas for future research in this project. Now that the two first prototypes have been created (Tangram and MirrorTrack), we need to merge them together. Moreover, we should start to conduct users studies to quantify the improvement made in mathematical reasoning by the users. Both systems are in their early stages and their robustness 'on the field' should be tested more in details (users may possibly try to 'break' the system).

Another possible way to investigate could be to reduce the number of cameras to two by eliminating the top-down camera. Indeed, this camera was dedicated to objects' tracking because in this position, the camera image plane was parallel to the tabletop surface (no perspective effect). However, it is possible to eliminate this camera and track the objects with the two side-mounted cameras (minimizing the occlusion problem). Indeed, the two sided cameras can track objects and fingertips at the same time. However, this solution would bring another problem of perspective effect on objects' shapes but research were done for algorithms working in this case (such as *'Perspective-transformation-invariant generalized hough transform for perspective planar shape detection and matching'* [46] or *'Planar 3D Object Detection by Using the Generalized Hough Transform'* [27])

Bibliography

[1] A. Agarwal, S. Izadi, M. Chandraker, and A. Blake. High precision multi-touch sensing on surfaces using overhead cameras. *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP '07. Second Annual IEEE International Workshop on*, pages 197–200, Oct. 2007.

[2] A. Agarwal, C. V. Jawahar, and P. J. Narayanan. A survey of planar homography estimation techniques. Technical report, 2005.

[3] A. Aguado, M. Montiel, and M. Nixon. Arbitrary shape hough transform by invariant geometric features. In *International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2661–2664, 1997.

[4] G. Ahuja, G. Kogut, E. B. Pacis, B. Sights, D. Fellars, and H. R. Everett. Layered augmented virtuality. In *RA '07: Proceedings of the 13th IASTED International Conference on Robotics and Applications*, pages 258–263, Anaheim, CA, USA, 2007. ACTA Press.

[5] A. Argyros and M. Lourakis. Vision-based interpretation of hand gestures for remote control of a computer mouse. In *CVHCI06*, pages 40–51, 2006.

[6] E. Arias, H. Eden, and G. Fisher. Enhancing communication, facilitating shared understanding, and creating better artifacts by integrating physical and computational media for design. In *DIS '97: Proceedings of the 2nd conference on Designing interactive systems*, pages 1–12, New York, NY, USA, 1997. ACM.

[7] S. E. Ashley and R. Green. A method for identifying irregular lattices of hexagonal tiles in real-time. In *IVCNZ'07*, 2007.

[8] R. Azuma. A survey of augmented reality, 1995.

[9] S. Belongie and J. Malik. Matching with shape contexts. *Content-based Access of Image and Video Libraries, 2000. Proceedings. IEEE Workshop on*, pages 20–26, 2000.

[10] H. Benko, A. D. Wilson, and P. Baudisch. Precise selection techniques for multi-touch screens. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 1263–1272, New York, NY, USA, 2006. ACM.

[11] D. Borghesani, C. Grana, and R. Cucchiara. Color features comparison for retrieval in personal photo collections. In *ACS'08: Proceedings of the 8th conference on Applied computer scince*, pages 265–268, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[12] S. K. Card, A. Newell, and T. P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2000.

[13] H. E. Center and H. Eden. Getting in on the (inter)action: Exploring affordances for collaborative learning in a context of informed participation. In *In G. Stahl (Ed.) Proceedings of the Computer Supported Collaborative Learning (CSCL '2002) Conference*, pages 399–407, 2002.

[14] P.-K. Chung, B. Fang, R. W. Ehrich, and F. Quek. Mirrortrack. volume 0, pages 1–5, Los Alamitos, CA, USA, 2008. IEEE Computer Society.

[15] P.-K. Chung, B. Fang, and F. Quek. Mirrortrack - a vision based multi-touch system for glossy display surfaces. pages 571–576, 29 2008-Aug. 1 2008.

[16] G. S. Cox and G. D. Jager. A survey of point pattern matching techniques and a new approach to point pattern recognition. In *Proc. South African Symposium on Communications and Signal Processing*, pages 243–248, 1993.

[17] L. D. Cutler, B. Frhlich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 107–114, 1997.

[18] J. Deng and H. Tsui. An hmm-based approach for gesture segmentation and recognition. *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, 3:679–682 vol.3, 2000.

[19] P. Dietz and D. Leigh. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM.

[20] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, 1972.

[21] A. El-Sawah, C. Joslin, N. D. Georganas, and E. M. Petriu. A framework for 3d hand tracking and gesture recognition using elements of genetic programming. In *CRV '07: Proceedings of the Fourth Canadian Conference on Computer and Robot Vision*, pages 495–502, Washington, DC, USA, 2007. IEEE Computer Society.

[22] A. Esenther and K. Ryall. Fluid dtmouse: better mouse support for touch-based interactions. In *AVI '06: Proceedings of the working conference on Advanced visual interfaces*, pages 112–115, New York, NY, USA, 2006. ACM.

[23] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, May 2003.

[24] T. A. Germer. Angular dependence and polarization of out-of-plane optical scattering from particulate contamination, subsurface defects, and surface microroughness. *Appl. Opt.*, 36(33):8798–8805, 1997.

[25] P. Griffin and C. Alexopoulos. Point pattern matching using centroid bounding. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(5):1274–1276, Sep/Oct 1989.

[26] T. Grossman, K. Hinckley, P. Baudisch, M. Agrawala, and R. Balakrishnan. Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 861–870, New York, NY, USA, 2006. ACM.

[27] N. Guil, N. Guil, J. R. Cozar, J. R. Cozar, E. Zapata, and E. L. Zapata. Planar 3d object detection by using the generalized hough transform, 1999.

[28] J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM.

[29] J. Y. Han. Multi-touch interaction wall. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Emerging technologies*, page 25, New York, NY, USA, 2006. ACM.

[30] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146 – 157, 1997.

[31] T. Heap and D. Hogg. Wormholes in shape space: Tracking through discontinuous changes in shape. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 344, Washington, DC, USA, 1998. IEEE Computer Society.

[32] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. pages 1–19, 1999.

[33] Q. Huynh-Thu, M. Meguro, and M. Kaneko. Skin-color extraction in images with complex background and varying illumination. *Applications of Computer Vision, 2002. (WACV 2002). Proceedings. Sixth IEEE Workshop on*, pages 280–285, 2002.

[34] K. Imagawa, S. Lu, and S. Igi. Color-based hands tracking system for sign language recognition. *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on*, pages 462–467, Apr 1998.

[35] H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241, New York, NY, USA, 1997. ACM.

[36] S. Izadi, S. Hodges, A. Butler, A. Rrustemi, and B. Buxton. Thinsight: integrated optical multi-touch sensing through thin form-factor displays. In *EDT '07: Proceedings of the 2007 workshop on Emerging displays technologies*, page 6, New York, NY, USA, 2007. ACM.

[37] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *International Journal of Computer Vision*, pages 274–280, 1999.

[38] D. G. R. B. A. Kaehler. *Learning OpenCV, 1st Edition*. O'Reilly Media, Inc., 2008.

[39] P. Kaewtrakulpong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection, 2001.

[40] Y. Kakehi, M. Iida, T. Naemura, Y. Shirai, M. Matsushita, and T. Ohguro. Lumisight table: An interactive view-dependent tabletop display. *IEEE Computer Graphics and Applications*, 25:48–53, 2005.

[41] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis. Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, June 2005.

[42] Y. Kitamura, T. Konishi, S. Yamamoto, and F. Kishino. Interactive stereoscopic display for three or more users. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 231–240, New York, NY, USA, 2001. ACM.

[43] F. Lathuilière. Visual tracking of hand posture with occlusion handling. In *ICPR '00: Proceedings of the International Conference on Pattern Recognition*, page 7141, Washington, DC, USA, 2000. IEEE Computer Society.

[44] J. Letessier and F. Bérard. Visual tracking of bare fingers for interactive surfaces. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 119–122, New York, NY USA, 2004. ACM.

[45] L. Li, W. Huang, I. Y. H. Gu, and Q. Tian. Foreground object detection from videos containing complex background. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 2–10, New York, NY, USA, 2003. ACM.

[46] R.-C. Lo and W.-H. Tsai. Perspective-transformation-invariant generalized hough transform for perspective planar shape detection and matching. *Pattern Recognition*, 30(3):383 – 396, 1997.

[47] Y. x. Lu, Y. Liu, Y. Guo, L. t. Kong, X. q. Chang, and X. y. Shan. Features of human skin in hsv color space and new recognition parameter. *OPTOELECTRONICS LETTERS*, VOL 3:312–314, 2007.

[48] S. Malik. Real-time hand tracking and finger tracking for interaction. csc2503f project report. Technical report, Department of Computer Science, University of Toronto, 2003.

[49] S. Malik and J. Laszlo. Visual touchpad: A two-handed gestural input device. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 289–296, New York, NY, USA, 2004. ACM.

[50] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26:530–549, 2004.

[51] J. Martin and J. L. Crowley. An appearance-based approach to gesture-recognition. In *In Proceedings, ICIAP*, pages 340–347, 1997.

[52] C. Miller, A. Robinson, R. Wang, P. Chung, and F. Quek. Interaction techniques for the analysis of complex data on high-resolution displays. In *IMCI '08: Proceedings of the 10th international conference on Multimodal interfaces*, pages 21–28, New York, NY, USA, 2008. ACM.

[53] S. J. V. Nichols. New interfaces at the touch of a fingertip. *Computer*, 40:12–15, 2007.

[54] J. Patten, H. Ishii, J. Hines, and G. Pangaro. Sensetable: a wireless object tracking platform for tangible user interfaces. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 253–260, New York, NY, USA, 2001. ACM.

[55] H. Regenbrecht, T. Lum, P. Kohler, C. Ott, M. Wagner, W. Wilke, and E. Mueller. Using augmented virtuality for remote collaboration. *Presence: Teleoper. Virtual Environ.*, 13(3):338–354, 2004.

[56] J. M. Rehg and T. Kanade. Digiteyes: vision-based human hand tracking. Technical report, 1993.

[57] J. Rekimoto. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *CHI*, pages 113–120, 2002.

[58] J. Rekimoto and M. Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 378–385, New York, NY, USA, 1999. ACM.

[59] A. Sanghi, H. Arora, K. Gupta, and V. B. Vats. A fingertip detection and tracking system as a virtual mouse, a signature input device and an application selector. In *Devices, Circuits and Systems, 2008. ICCDCS 2008. 7th International Caribbean Conference on*, pages 1–4, 2008.

[60] Y. Sato, Y. Kobayashi, and H. Koike. Fast tracking of hands and fingertips in infrared images for augmented desk interface. *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 462–467, 2000.

[61] Y. Sato, Y. Kobayashi, and H. Koike. Fast tracking of hands and fingertips in infrared images for augmented desk interface. In *In Proc. of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 462–467, 2000.

[62] T. Scharf. Polarized Light in Liquid crystals and polymers. pages 40–47. SPIE Press 2002, Bellingham, 2006.

[63] S. D. Scott, R. L. Mandryk, and K. M. Inkpen. Understanding children's interactions in synchronous shared environments. In *IN PROCEEDINGS OF COMPUTER SUPPORTED COOPERATIVE LEARNING*, pages 333–341. ACM Press, 2002.

[64] K. Sedighian and M. Klawe. Super tangrams: a child-centered approach to designing a computer supported mathematics learning environment. In *ICLS '96: Proceedings of the 1996 international conference on Learning sciences*, pages 490–495. International Society of the Learning Sciences, 1996.

[65] C. Shen, N. B. Lesh, F. Vernier, C. Forlines, and J. Frost. Sharing and building digital group histories. In *CSCW '02: Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 324–333, New York, NY, USA, 2002. ACM.

[66] J. Shi and C. Tomasi. Good features to track. pages 593–600, 1994.

[67] P. Steurer and M. B. Srivastava. System design of smart table. In *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, page 473, Washington, DC, USA, 2003. IEEE Computer Society.

[68] N. A. Streitz, J. Geiler, T. Holmer, S. Konomi, C. Mller-tomfelde, W. Reischl, P. Rexroth, P. Seitz, and R. Steinmetz. i-land: An interactive landscape for creativity and innovation. pages 120–127. ACM Press, 1999.

[69] K. Suzuki, I. Horiba, and N. Sugie. Linear-time connected-component labeling based on sequential local operations. *Computer Vision and Image Understanding*, 89(1):1–23, 2003.

[70] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. pages 221–244, 1992.

[71] B. Ullmer and H. Ishii. The metadesk: Models and prototypes for tangible user interfaces. pages 223–232. ACM Press, 1997.

[72] V. V. Vassili, V. Sazonov, and A. Andreeva. A survey on pixel-based skin color detection techniques. In *in Proc. Graphicon-2003*, pages 85–92, 2003.

[73] Y. Verdie, B. Fang, and F. Quek. Mirrortrack: tracking with reflection - comparison with top-down approach. In *ICMI-MLMI '09: Proceedings of the 2009 international conference on Multimodal interfaces*, pages 347–350, New York, NY, USA, 2009. ACM.

[74] C. von Hardenberg and F. Bérard. Bare-hand human-computer interaction. In *PUI '01: Proceedings of the 2001 workshop on Perceptive user interfaces*, pages 1–8, New York, NY, USA, 2001. ACM.

[75] J. Wall. Demo i microsoft surface and the single view platform. *Collaborative Technologies and Systems, International Symposium on*, 0:xxxi–xxxii, 2009.

[76] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[77] A. D. Wilson. Touchlight: an imaging touch screen and display for gesture-based interaction. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces*, pages 69–76, New York, NY, USA, 2004. ACM.

[78] A. D. Wilson. Robust computer vision-based detection of pinching for one and two-handed gesture input. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 255–258, New York, NY, USA, 2006. ACM.

[79] C. R. Wren, C. R. Wren, Y. A. Ivanov, and Y. A. Ivanov. Volumetric operations with surface margins. In *in Computer Vision and Pattern Recognition: Technical Sketches*, 2002.

[80] Q. Wu, P. Boulanger, and W. Bischof. Robust real-time bi-layer video segmentation using infrared video. *Computer and Robot Vision, 2008. CRV '08. Canadian Conference on*, pages 87–94, May 2008.

[81] S. Wu, L. Hong, and F. Quek. Image based hand tracking via interacting multiple model and probabilistic data association (imm-pda) algorithm. *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW '03. Conference on*, 5:57–57, June 2003.

[82] Y. Xiong, B. Fang, and F. Quek. Extraction of hand gestures with adaptive skin color models and its applications to meeting analysis. *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on*, pages 647–651, Dec. 2006.

[83] E. Yoruk, E. Konukoglu, B. Sankur, and J. Darbon. Shape-based hand recognition. *Image Processing, IEEE Transactions on*, 15(7):1803–1815, July 2006.

[84] B. Zarit, B. Super, and F. Quek. Comparison of five color models in skin pixel classification. *Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on*, pages 58–63, 1999.

[85] Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(11):1330–1334, 2000.

[86] Z. Zhang and T. Kanade. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27:161–195, 1998.

[87] X. Zhu, J. Yang, and A. Waibel. Segmenting hands of arbitrary color. *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, pages 446–453, 2000.