UNIVERSITA' DEGLI STUDI DI TRENTO

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea

Magistrale

Final Thesis

TITLE

A framework for inexpensive and unobtrusive tracking of everyday objects and single-touch detection

Relatore/1st Reader:

Prof. Dr.-Ing. Giuseppe Riccardi
Università degli Studi di Trento

Laureando/Graduant:

Ignacio Martín Avellino Martinez

Relatore/2nd Reader

Prof. Dr. Jan Borchers
Rheinisch-Westfälische Technische Hochschule Aachen

Anno accademico 2012 - 2013

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Trento, March 2013
Ignacio Martin Avellino Martinez

Contents

List of Figures

List of Tables

Abstract

Technology development has surrounded people's daily life with devices, which, for people's convenience, include most of the time a dedicated controller that allows operating them remotely. A problem arises when such controllers are temporarily unavailable or out of a person's reach. *Instant User Interfaces (IUIs)* allow people to casually reappropriate an everyday object at their reach and use it as an input device. In order for a system to interpret such interaction, a system capable of detecting object movement in 3D space as well as touches on its surface is needed.

This work aims at the implementation and evaluation of an object tracking and single-touch sensing system, which can operate without intruding into the user's space. By using a 3D digital model of a real world object (e.g., obtained through an off-the-shelf 3D scanner) it is possible to: (a) automatically and reliably recognize and track objects; (b) identify user actions performed on such objects such as movement and finger touches; and (c) assign semantic value to such actions.

A tracking evaluation performed using data obtained by a state-of-the-art tracking system showed that changes in position were tracked successfully with small errors. Performance on orientation estimation error was found to be higher, although still acceptable. An evaluation on touch accuracy showed how the system performs when the size of touch enabled areas changes in an object.

The presented system enables further research on *Instant User Interfaces* by allowing investigators to understand interaction with everyday objects without interrupting users.

Acknowledgements

Firstly and most importantly, I would like to state how deeply thankful I am to my family for their constant support, overflowing words of joy and for always going an extra mile in order to make my studies abroad my best experience so far. Nothing would be the same without their love.

To my thesis advisor, Dipl.-Inform. Dipl.-Wirt.Inform. Max Möllers for invaluable guidance at all times during the course of this work, especially for his patience and constant availability. To him, I am very grateful for unconditionally sharing his time and knowledge with me.

To both my examiners, Prof. Dr.-Ing Giuseppe Riccardi and Prof. Dr. Jan Borchers for trusting in my ability to perform this work and for later evaluating it.

To my first examiner Prof. Dr.-Ing Giuseppe Riccardi for granting me with the chance of carry out this work while in Aachen. Thank you for your trust and remote guidance.

To my second examiner Prof. Dr. Jan Borchers and the Media Computing Group for providing an incredible infrastructure, abundant resources and a supporting work environment, which allowed me to develop my work to the fullest.

To all the participants of the user studies carried out during the evaluation of this work for their time and patience.

Lastly, special thanks to Chatchavan Wacharamanotham, M.Sc. for sharing his extensive knowledge on designing studies and analyzing results, as well as enlightening paths and constantly generating new questions.

Thank you,

Ignacio Avellino

Conventions

Throughout this thesis the following conventions are used.

Text conventions

Definitions of technical terms or short excursus are set off in colored boxes.

DEFINITION:
Definitions are presented in a colored box such as this one

Definition:
Definition

Parameters of the presented system are set off in colored boxes as below.

PARAMETER:
Parameters are presented in a colored box such as this one

Parameter:
Parameter

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The whole thesis is written in American English.

Chapter 1

Introduction

As a result of scientific development, electronic devices that enrich and ease everyday tasks have become ubiquitous in people's daily lives. Along with them, dedicated controllers have become a common sight: remote controllers, light switches, dimmers, mouse and keyboard, game console controllers, among many others, all designed to control devices. As people become accustomed to having controllers at hand, the question of how useful such devices are when no controller is in sight arises.

Devices have dedicated controllers. How useful are they when their controller is not available?

The use of everyday objects as substitutes for dedicated input devices has been studied in depth by Corsten [2012] in his Master Thesis: *Co-Optjects: Instant User Interfaces through Everyday Objects*. Such substitution occurs when a dedicated controller is not within a person's reach, it consists on choosing an everyday object and performing an action with it in order to control a specific device. This action is called object appropriation.

Instant User Interfaces allow everyday objects to be used as device controllers.

When people interact with everyday objects in order to use them as input devices, they can create ad-hoc gestures and map them into actions (Figure 1.1). It is assumed that the person interacts with the object using its hands, therefore, three basic aspects are included in such interaction—movement of the object in space for gestures that include rotation or translation of such object; finger touches on the surface for gestures that involve contact with the object in

Interaction with everyday objects includes: movement, rotation, touches, and deformation.

Figure 1.1: *Example of object appropriation—an everyday object being used as an input device.*

specific parts; and finally deformations of the object such as bending it or flexing mobile parts.

There is a need to identify how people use everyday objects.

In order to assign semantic meaning to the interaction between people and everyday objects, it is essential to determine and understand how people are using the object and in which situation context. For this end, a system capable of tracking objects in 3D and sensing touches on its surface is needed.

1.1 Motivation

Lack of automatic and reliable object recognition for studying interaction in real-time.

The main limitation of existing systems for studying user interaction is not having an automatic and reliable object recognition and tracking software that allows a system to be informed of how a person interacts with an object. This limitation comprehends:

1. Automatic and reliable object recognition from a 3D model, which represents the shape of the real world object.

2. Real-time tracking of objects in space, which would allow a system to be aware of changes in the 3D position and orientation of the recognized object.

3. 3D input recognition, where manipulation of the object in terms of finger touches on its surface is recognized.

A system which overcomes such limitations would breach the existing technological gap for investigating the use of everyday objects as input devices. This gap involves being able to obtain data without disturbing user interaction or having to extensively augment the manipulated objects, it does not concern precision on measurements. These disturbances in user interaction usually arise from the use of markers or when the user is limited to hold objects in specific ways. In other words, allowing natural interaction while it is being sensed and interpreted by a system.

It is essential that studying user interaction does not disrupts the interaction itself.

The needed level of precision in this system is not extremely granular. Tracking the position of objects does not need to be millimeter precise, considering that actions performed by users on objects are not usually millimeter precise themselves. Take as an example using a cup to control the volume of a TV set, where a change in rotation is mapped to increasing or decreasing the volume level of such device. As a person rotates the cup, a certain amount of rotated degrees are mapped into a volume-change step. It is very hard for a person to rotate an object a precise amount of degrees without using a measuring tool such as a semi circle protractor. Fine rotation movements with a precise amount of degrees are not usually produced by users. On the contrary, a user normally would iteratively rotate the object until the volume reaches the desired level, even if it involves overshooting and correcting its produced rotation.

Extreme precision is not a requirement, people do not perform extremely precise movements themselves when manipulating objects.

1.2 Contribution

An affordable and unobtrusive object-tracking and single-touch sensing system.

The contribution of this work is an affordable and unobtrusive object-tracking and single-touch sensing system. Users of this system will be able to perform studies on the interaction between people and everyday objects without interrupting such interaction. The main concern is the ability to track objects and sense touches without intruding user interaction and using inexpensive components, while providing a reasonable precision.

1.3 Thesis Overview

This work is organized as follows:

- In Chapter 2, related work is presented, including: Instant User Interfaces, along with the concepts that build this paradigm of everyday-object interaction; the technologies used in the evaluation of the mentioned system; the technologies and theory that compose the presented tracking and touch sensing system; and similar existing systems, including their deficiencies.

- In Chapter 3, the description of the system is found. Raging from its concept to its implementation, all the necessary details to understand how the system works are detailed.

- In Chapter 4, the evaluation regarding the tracking capabilities of the system is presented. Data obtained on tracking objects position and orientation are compared to a state-of-the-art tracking system.

- In Chapter 5, the reader can find the evaluation corresponding to the single-touch sensing capabilities of the system. Through a user study, the accuracy of the system is presented as available touch zones on the object change in size and amount.

- In Chapter 6, a summary of the presented work is included, applications for studying user interaction with everyday objects through the use of this system are mentioned, and the work is closed with recommendations as well as potential/suggested future work.

Chapter 2

Related work

In this section, concepts on interaction with everyday objects, as well as the pieces and bits used to build the object-tracking and single touch-sensing system, are covered. Existing systems for investigating interaction with everyday objects are also presented.

2.1 Object Appropriation

As people grow up surrounded by devices, they learn how to use their remote controllers with time. Whenever a new controlling device is presented to a person, the individual is challenged with the task of learning how to use it. To reduce the impact of learning, physical controllers tend to have similar form, functions and affordances.

People need to learn how to use new controllers.

PHYSICAL AFFORDANCES:
"[...] the term affordances refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used [...]" [Norman, 2002]

Definition: *Physical Affordances*

Simultaneously, people learn as well how to use objects that are part of their environment; pens, mugs, scissors and cutlery among many others, are all objects which can be com-

People learn how to use everyday objects since a young age. An object can be substituted for another one with similar affordances.

monly found in people's everyday life. Once the knowledge of how to use one of these objects is acquired, the user is able to use virtually all objects of this class, due to the physical affordances between them remaining mostly the same. Having objects with similar physical affordances allows people to appropriate an object and use it as if it was another. Enabling the studying of how people appropriate objects is the main goal of this work.

Definition: *Object Appropriation*

OBJECT APPROPRIATION:
The process of using a designed tool or object in a different context and for a different purpose than intended [Corsten, 2012]

Examples of object appropriation can be found in Section 2.6 together with systems which enable the study of such interaction.

2.2 Instant User Interfaces (IUIs)

Using everyday objects as input devices.

The recently coined term *Instant User Interfaces* refers to using everyday objects to control remote devices. The formal definition of the term can be found below.

Definition: *Instant User Interface.*

INSTANT USER INTERFACE.:
An Instant User Interface is a user interface that enables the user to select an object within reach in order to control a remote technical system. Based on existing knowledge about physical affordances of the object and the features of the target system, the user establishes ad hoc mappings from artifact to system function without requiring explicit prior setup. [Corsten, 2012]

Everyday objects examples.

In a first diary study by Corsten [2012], it was discovered that the ten most frequently found objects in everyday life are: tables, boxes/containers, paper, bottles, drinking vessels, pens, underlays, cables, books and laptops/computers.

In a second study, it was investigated how people solved different tasks using the identified everyday objects. The study focused on how people controlled a TV set by performing actions such as turning it on or off, switching channels and changing the volume. In the Wizard of Oz study, the main identified gestures were touching, pushing, planar movements and stationary turning and pointing.

There have been studies on how people solve tasks using everyday objects.

New ways of interaction were explored throughout the aforementioned study, using a prototype that detected changes in the position of objects through the use of markers, and detected touches on surfaces through the use of a Kinect depth camera. The setup required objects to be prepared and users had to be mindful of keeping the markers visible to the camera. The contribution of this work aims at providing an improved inexpensive and unobtrusive tracking and single-touch sensing system, such that further studies surrounding everyday object interaction can be carried out.

Previous systems used markers, requiring users to be aware of the presence of the system.

Interacting with everyday objects has the advantage of having a tangible object at hand, and not a virtual one. Virtual objects, such as a button bound to the rules of a digital UI, have no weight, texture or any other tangible property that provides physical feedback. *Tangible User Interfaces* explore the implications of having physical objects as user interfaces.

IUIs provide haptic feedback.

2.2.1 Tangible User Interfaces (TUIs)

TANGIBLE USER INTERFACES (TUIS):
TUIs show concrete ways to move beyond the current dominant model of GUI bound to computers with a flat rectangular display, windows, a mouse, and a keyboard [...] by coupling digital information to everyday physical objects and environments—Ishii and Ullmer [1997]

Definition: *Tangible User Interfaces (TUIs)*

IUIs apply the concept of TUIs to everyday objects. Graspable objects become controllers capable of sending digital information to a device, through the decoupling of the intangible digital representation and the controlling devices

Users interact with physical tangible controllers that send information to the controlled device.

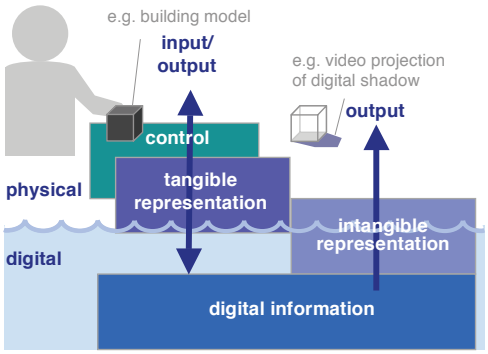

Figure 2.1: *TUIs: detachment between the digital and physical world. From [Ishii, 2008]*

as seen in Figure 2.1. Users interact with tangible controllers, which in turn send digital information, through a channel, to the digital stratum. This layer responds to the user's actions through intangible objects in a GUI.

New trends aim towards deformable and materials that make technology even more invisible.

Current trends on TUIs aim towards deformable materials. This has been labeled *Radical Atoms* by Ishii et al. [2012] at the MIT Lab. Materials that can change form and appearance dynamically can be rearranged in front of the user in the same way digital pixels are. Representation and control become even more decoupled and people become unaware of the underlaying digital tear. By interacting with and receiving feedback from a physical device, computation becomes invisible. Among many benefits, it is important to highlight both continuous haptic feedback and null inconsistencies between tangibles and the underlying digital model. This trend could make computing truly ubiquitous and melt computational devices into everyday objects.

2.2.2 Ubiquitous Computing

Definition: *Ubiquitous Computing (UbiComp)*

UBIQUITOUS COMPUTING (UBICOMP):
Technologies that weave into the fabric of everyday life until they are indistinguishable from it—[Weiser, 1991]

Ubiquitous Computing plays a major role when it comes to IUIs. By controlling virtual devices through the use of physical everyday objects, technology is seamlessly integrated into the device.

Technology disappears into the fabric of everyday life.

As mentioned, IUIs allow temporary use of an everyday object as a device controller when a dedicated controller is not available, making device controllers ubiquitous. Controllers become now more available due to the fact that common objects are turned into replacement controllers.

IUIs aim at making controllers more ubiquitous

2.3 3D Cameras

Depth cameras, also named RGB-D cameras on account of their ability to perceive the world in full color and depth, have become with time commercially available and thus affordable.

Provide information of the world in color and depth.

These cameras are sensing devices that capture RGB images along with per pixel depth information. There are two main technologies for sensing depth information—*active stereo* and *time-of-flight*.

There are two main technologies.

2.3.1 Active Stereo Cameras

These cameras allow independent panning and tilting of two RGB cameras, which allows determining the depth of each pixel in a similar way a human eye does. Although it suffers from problems, it was a widely used method before time-of-flight cameras took the lead. The two separate cameras lead to difficulties with camera calibration and ambiguities on correspondence that must be solved computationally. Lastly, this method exploits parallax along one axis and thus cannot estimate the depth on contours parallel to it. Solutions have been proposed as mentioned by Adelson and Wang [1992] with the disadvantage of involving a third camera or adding computation cycles to the depth image preprocessing. The main advantage of this technique is that it does not require special illumination conditions.

Independent cameras provide two images that, when composed, provide a single 3D image. This approach is robust to light conditions, but suffers from many problems.

2.3.2 Time-of-Flight Cameras

Rely on infrared light phase-delay to compute distances. Suffer from depth distortion, multipath error and object boundary ambiguity.

Time-of-flight cameras can be used to estimate 3D structures directly without using traditional computer-vision algorithms. Distances to objects can be computed by measuring the phase-delay of reflected infrared (IR) light [Hansard et al., 2012]. Both systematic and non-systematic error must be dealt with. Systematic errors come from infrared demodulation error, integration time error, amplitude ambiguity and temperature error. Non-systematic error arises because of the used sensing device, among which is common to find: depth distortion, produced by IR saturation; multipath error, produced by superimposition of several IR sources; and object boundary ambiguity, produced by depth ambiguity between foreground and background on the pixels near boundaries.

Affordable and require low calibration.

These cameras provide an out-of-the-box solution with virtually no manual calibration needed. Commercially available solutions can be found for these type of cameras, varying in size, price, and precision, such as [pri], [dep], [fot], and [dim]. A Kinect depth camera, which uses a Primesense 3D sensor, can be acquired for about USD150 in the US, providing a low-cost and ready-to-use time-of-flight camera. The camera captures both a 640 by 480 pixel registered image and their corresponding depth points at thirty frames per second.

How a Kinect depth camera works.

The Kinect depth camera consists of three basic components: an infrared light emitter, an infrared light receiver, and an RGB camera. The Kinect emits a beam of infrared light as a constant pattern of speckles [Freedman et al., 2012]. While functioning, the camera detects this pattern and correlates it to a reference pattern. The latter is generated in a calibration process, where the image of the pattern at a known distance is stored. When speckles on the recognized pattern are shifted from the baseline reference pattern, it is possible to compute the distance to the pixel where the shift is produced. This allows for computing the absolute distance from the real world point to the sensor. An example image is shown in Figure 2.2

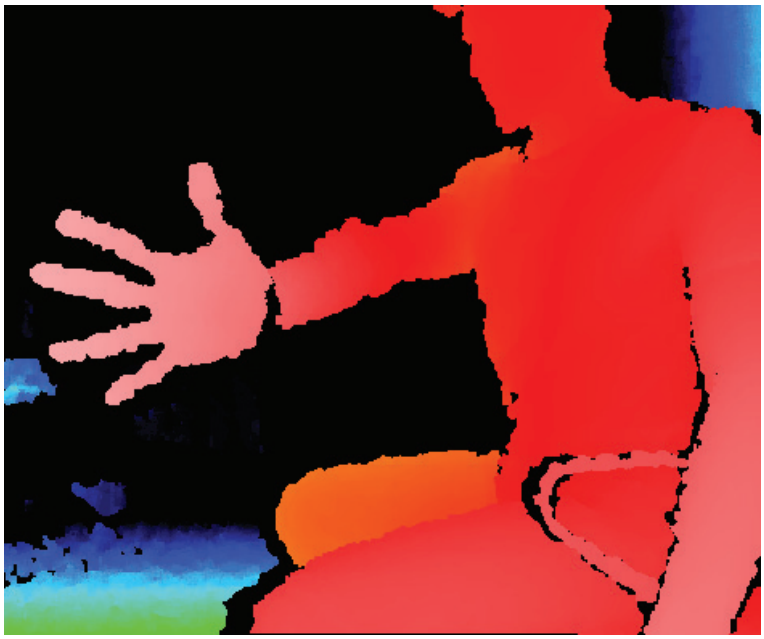

Figure 2.2: *Colored depth image of a person obtained using a Kinect. Distances correspond to a color spectrum raging from light red to dark blue.*

On the downside, this approach produces accuracy to be reduced with increasing distances. According to Khoshelham [2011], "the random error of depth measurements increases quadratically with increasing distance from the sensor and reaches four centimeters at the maximum range of five meters"; as well he concludes that "the depth resolution also decreases quadratically with increasing distance from the sensor". The point spacing in the depth direction, along the optical axis of the sensor, is as large as seven centimeters at the maximum range of five meters". These measurements were obtained by determining alignment residuals between a point cloud[1] generated by the Kinect depth camera and a state-of-the-art laser scanner, used as ground truth data. Three methods were used: point–point distances using Iterative Closest Point (ICP, see Section 2.5.1); point–plane distances, without accounting for differences due to scaling; and point–plane, accounting for differences due to scaling. A brief summary is presented on Table 2.1.

The largest problem of the Kinect is its quadratic loss of precision as depth increases linearly.

[1] A point cloud is a set of vertices in a three-dimensional coordinate system

The reader should note that there were no significant errors introduced by scaling differences using the Kinect depth camera.

Method	Alignment residuals (cm)				
	Min	Mean	Med	Std	Max
point–point distances	0.1	1.2	0.9	0.9	4.4
point–plane distances with scale	0.0	1.1	0.8	0.9	7.1
point–plane distances without scale	0.0	1.1	0.9	0.9	7.0

Table 2.1: *Alignment error between a point cloud obtained with a Kinect depth camera and the point cloud of the same scene obtained through a High-End Laser Scanner. Data from [Khoshelham, 2011].*

The Kinect works in a range of $[0.8 - 3.5]$ meters depth.

Another disadvantage of the Kinect depth camera is its practical limit of around 0.8 to 3.5 meter distance from the sensor to an object. This limitation is not a major one in the context of this work, due to the fact that tracked everyday objects can be placed within this threshold distance without interrupting user interaction. Given its low cost and virtually null manual calibration, it provides an affordable channel sensing solution.

2.4 VICON Tracking System

State-of-the-art 3D object tracking system. It relies on infrared light reflection and 2D image processing.

The VICON motion tracking system[2] is a complete system built for tracking objects in 3D. It is basically composed by: a number of cameras capable of emitting and detecting infrared light, infrared reflective markers, and processing hardware. A Bonita VICON camera and a close-up of an infrared reflective marker are shown in Figure 2.3. The markers are attached to objects intended to be tracked; emitted infrared light reflects on the markers and it is detected by the cameras. The processing hardware infers the 3D coordinates of the markers from a set of 2D images provided by the infrared cameras, located at different angles.

[2] http://www.vicon.com/

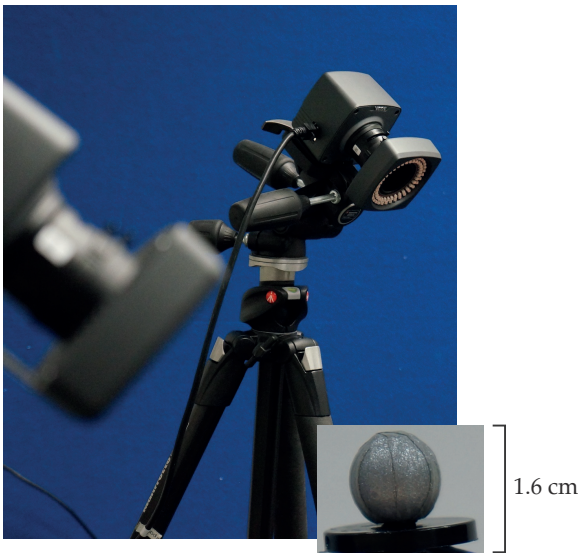

Figure 2.3: *A VICON Bonita tracking system camera on the background and an infrared reflective marker on the foreground.*

Studies performed by Windolf et al. [2008], revealed that the average accuracy of the system for a favorable combination of parameters is $63 \pm 5\mu m$ (mean $\pm$ SD) with a precision of $15\mu m$. For markers with a size of fifteen millimeters, the highest registered accuracy was of $76 \pm 3\mu m$ whereas the lowest registered accuracy was of $129 \pm 35\mu$ with a precision averaging from 15μ to 21μ. Accuracy and precision, even in an unfavorable setup, are sufficient enough for studying interaction between humans and everyday objects. It is important to note that accuracy describes the deviation of the measured value from the true one, whereas precision states the repeatability of measurements taken under identical circumstances.

Accuracy measured in μm.

2.5 Foundation

In this section, the foundation concepts that compose the presented system are found. Object tracking and object alignment are defined and contextualized to the current work. An existing implementation of each one of them, later used in the system itself, is presented.

Object tracking and alignment are presented.

2.5.1 Object Tracking

Object tracking is narrowed down to the context of this work

Object tracking is a widely studied topic in the field of computer vision, narrowed down in this section to the context of understanding interaction between users and everyday objects, based on different definitions.

- From the book *Fundamentals of object tracking* by Challa et al. [2011], object tracking "refers to the problem of using sensor measurements to determine the location, path and characteristics of objects of interest. [...] the typical object tracking problem is essentially a state estimation problem where the object states to be estimated from noisy corrupted and false measurements are kinematic states such as position, velocity and acceleration".

- In their *Object Tracking Survey*, Yilmaz et al. [2006] define object tracking in 2D as "the problem of estimating the trajectory of an object in the image plane as it moves around a scene. Additionally, depending on the tracking domain, a tracker can also provide object-centric information, such as orientation, area, or shape of an object". This definition can be extended to the problem of tracking in 3D by considering the space instead of the image plane for position and orientation.

Three different definitions of object tracking

- Lepetit and Fua [2005], in their *Survey on Monocular Model-Based 3D Tracking of Rigid Objects*, propose that 3D tracking "aims at continuously recovering all six degrees of freedom (6DoF) that define the camera position and orientation relative to the scene, or, equivalent, the 3D displacement of an object relative to the camera".

All definitions commonly state that the location of an object at a given moment of time must be known.

All three presented definitions state that the location of an object at a given moment of time is essential to be known. In addition, its orientation may as well be required although not mandatory for tracking. In the specific context of tracking objects in order to investigate the interaction between everyday objects and people, gestures arise from

changes in the location and orientation of an object, thus knowing an object orientation is nevertheless considered to be essential. In the context of this work, object tracking is defined as below.

OBJECT TRACKING:
The estimation of the location and orientation displacement of an object within a 3D space sensed in two moments of time using sensor measurements.

Definition: *Object Tracking*

Object tracking is a problem that has clear practical applications in Human–Computer Interaction (HCI). As stated by Yilmaz et al. [2006]: "There are three key steps in video analysis: detection of interesting moving objects, tracking of such objects from frame to frame, and analysis of object tracks to recognize their behavior. Therefore, the use of object tracking is pertinent in the tasks of: [...] human–computer interaction, that is, gesture recognition, eye gaze tracking for data input to computers, etc". In a similar way, Lepetit and Fua [2005] mentions that "object tracking has applications in Man–Machine interfaces for tangible user interfaces, through for example, continuously updating the position of a hand-held object which then serves as a 3D pointer".

Object tracking is a problem that concerns HCI.

The architecture of a typical object tracking system is exposed by Challa et al. [2011]. As it can be observed in Figure 2.4, data is acquired from a channel by a sensor, transformed into a specific format, and interpreted by an algorithm in order to track an object. The work of this thesis has its focus on an algorithm capable of interpreting the captured data, and thus track a specific object.

Architecture of a typical object tracking system. This work focuses on the algorithm that interprets data and tracks objects.

Some of the prior problems to tracking that arise, which are not inherent to proper object tracking, include: the representation of the tracked object; the first step to tracking, which is detection of the object in the scene; and the problem of capturing data and transforming it.

Prior problems to tracking include among others initial object pose estimation.

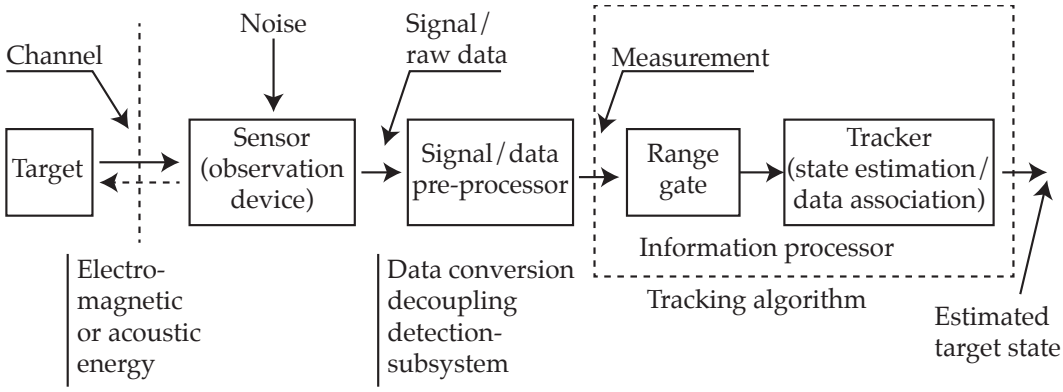

Figure 2.4: *Architecture of a typical object tracking system. From [Challa et al., 2011]*

Object Tracking With Depth Data

Most of the traditional object tracking methods either rely on 2D image data or on tracking hardware attached to the tracked object [Yilmaz et al., 2006].

Many tracking methods require the use of fiducials or electronic components. These foreign objects alter user–everyday object interaction.

Hardware-based tracking requires augmenting the tracked object with a track enabled entity such as fiducials or an electronic device, essentially a Radio Frequency Identification tag (RFID) or a magnetic component. These techniques have two main disadvantages—they are intrusive and they require objects to be prepared before their use. Such preparation comes from the necessity of having the fiducial markers visible at all times with respect to a camera and having to prepare the object with the markers or hardware. These required preparations contravene with the basic requirements of an unobtrusive object tracking system.

Vision-based tracking techniques require either several cameras, or are computationally intensive. Tracking accuracy with 6DoF is very low.

In the case of image-based methods, one or several camera video feeds of a moving 3D object are used to recompose a 3D scene. Even if not intrusive to the movement of the object, these vision-based tracking techniques require complex computer vision algorithms. Usually they rely on naturally present features, such as edges, corners, or texture. Examples of these techniques, which rely on global features, are: edge-based methods, optical flow-based methods, and template matching. Methods that use localized features include: interest point-based methods, n-image methods and filter-based methods [Lepetit and Fua, 2005].

All of these have low accuracy for tracking with 6DoF, since they receive input from a conventional 2D camera. The emergence of depth cameras sheds light to new 3D object tracking methods, which can take advantage of depth-information.

Development of tracking algorithms which take advantage of depth data has taken drive in the past five years with the increasing availability of depth cameras, being the main focus on tracking peoples' hands and bodies. Hand tracking with depth data has been explored by Frati and Prattichizzo [2011] to enhance haptic on wearables; by Raheja et al. [2011] to track fingertips and palms for gesture input; and by Oikonomidis et al. [2011] to detect the hand as an articulated model. Body tracking has been explored by Bleiweiss et al. [2010] with the purpose of using bodies as avatars in games; by Xia et al. [2011] to detect body gestures; and by Shotton et al. [2011] to accurately track body joints in real-time. Two of these studies, one focused on hand tracking and the other in body tracking, are elaborated below.

Current tracking algorithms with 3D input data aim at hand or body tracking

Hand tracking using input data from a Kinect is explored by Oikonomidis et al. [2011], where, by using a model template, tracking of articulated objects is achieved. The method provides accurate continuous 3D tracking and it does not rely on markers. Even if the hand points are segmented from the rest of the scene through skin color recognition, the algorithm relies mainly on depth information, thus changes in light conditions do not affect tracking heavily. Hand pose determination is based on the use of twenty-seven predetermined hand model parameters, used for minimizing the discrepancy between hand hypotheses and actual observations. These parameters correspond to different parts of the hand such as the palm, finger body, fingertips and finger joints. After several hand poses are determined as hypotheses, they are evaluated by fitting the model into the scene points and analyzing for outliers using a formula based on a distance threshold. A Particle Swarm Optimization (PSO) is then used to select one hand pose from the filtered hypotheses. The result of the algorithm, as seen in Figure 2.5, has a low error, reported to be bound to the range $[5 - 25]$ millimeters.

An example of hand tracking using the Kinect.

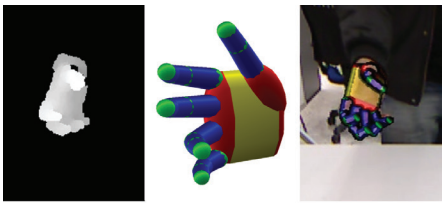

Figure 2.5: *Image modeling using a Kinect. The raw data (left) is modeled into a hand (middle), which follows the real movement of the person's hand (right). From Oikonomidis et al. [2011].*

An example of body tracking using the Kinect.

Xia et al. [2011] focus on human body tracking by using both 2D edge detection and 3D shape detection. In a first stage, a 2-step head detection process takes place: firstly a 2D chamfer distance matching algorithm scans the RGB obtained image from the Kinect in order to find bodies. Candidate regions are examined using a 3D head model and a final estimation is performed. Once the region where the body belongs to is found, the body contour is segmented from the rest of the data through a region growing algorithm, which takes advantage of the fact that depth values throughout the detected person's body are continuous. Using a starting point based on the detected head, the depth similarities with each neighbor pixel are computed and classified as belonging to the body or not, according to a distance threshold. This is repeated iteratively taking into account the points classified as belonging to the body. The algorithm output has a precision, recall and accuracy above 96%, providing an accurate, low-error solution for body contour detection and tracking.

Arbitrary object tracking using two Kinects. The method is based on template matching.

There has as well work been done in tracking arbitrary objects, and not only body parts, using depth data. Liu et al. [2012] provide a 3D object training method using multiple Kinect sensors for tracking. Several poses of an object are registered using the depth camera and by using the Dominant Orientation Templates (DOT) method, the position and orientation of the object can be determined in real-time. As a proof of concept, a Lego building instruction system was built, where the system scanned a Lego piece and instructed the user where to place it in order to assemble a Lego model.

Figure 2.6: *A Lego piece being scanned by two Kinects. From [Liu et al., 2012]*

The previously mentioned object tracking methods make use of depth information. Nevertheless, they do not treat the input data as a 3D scene, where each point is represented in a defined 3D space. A literature review did not yield work which treats depth information in this way with the purpose of object tracking, but only for scene modeling such as in[Henry et al., 2010], *KinectFusion* [Izadi et al., 2011] and [Huang et al., 2011]. Also, there is no modeling of the tracked objects.

Reviewed method do not treat input as a 3D scene, and they do not have a model of the tracked object.

Treating the input data from a depth camera as a set of points in a defined 3D space, gives room for new object tracking methods. Having a 3D model of the desired object to be tracked and a 3D representation of a scene, the object could be fitted into the scene by analyzing both sets of points.

Having 3D information about a scene and an object sheds light to new tracking methods.

Iterative Closest Point (ICP)

ICP is a method for aligning two 3D models in a defined space—the *template* and *model* 3D models. The output of the algorithm is a rigid transformation that applied to the template, it transforms it such that it aligns the model, while minimizing an error metric. In the case of a single model that moves in space with time, if ICP is provided with the oriented model in two different moments in time, it yields the transformation produced by such movement. In other words, given the 3D information of an object in two different positions, say, from two consecutive frames retrieved by a depth camera, ICP determines the transformation that

ICP aligns two point clouds while minimizing an error metric. Aligning two consecutive point clouds in time of an object corresponds to determining its movement.

was applied to such object while it was moved.

Originally proposed to align different views of one object, and create its model.

This method was initially introduced by Yang and Medioni [1992] and Besl and McKay [1992] as a method for aligning two different views of the same 3D object, with the purpose of reconstructing a complete 3D model of an object. In this work, ICP is used to determine the transformation produced by the movement of an object in two moments of time, and thus track it. Before proceeding with the breakdown of the algorithm, a basic mathematical ground is laid concerning finding a correspondence point set alignment between two point sets; in other words, finding the rigid transformation, expressed as a rotation and a translation, between two point sets which minimizes an error metric.

Finding a Correspondence Point Set Alignment Between Two Point Sets

Mathematical background on finding a rigid transformation between two point sets.

Besl and McKay [1992] takes a Singular Value Decomposition approach based on the cross-covariance matrix of two point distributions for this purpose. Firstly, the reader should note that a rotation can be expressed in several ways, being one in terms of quaternions as $\vec{q_R} = [q_0\ q_1\ q_2\ q_3]^t$ where $q_0 \geq 0$, and $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. A 3 by 3 rotation matrix can be obtained from the quaternion representation of the rotation as Equation 2.1 depicts.

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 + q_2^2 - q_1^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 + q_3^2 - q_1^2 - q_2^2 \end{bmatrix} \tag{2.1}$$

Let $\vec{q_T} = [q_4\ q_5\ q_6]^t$ be a translation vector; the complete alignment state vector $\vec{q}$, composed by a rotation and a translation is denoted $\vec{q} = [\vec{q_R}|\vec{q_T}]^t$.

Let P and X be two point sets, such that $P = \{\vec{p}_i\}$ and $X = \{\vec{x}_i\}$ with $N_P = N_X$, being N_A the number of elements in the set A and having $\vec{p}_i$ correspond to $\vec{q}_i, \forall i \in [0..N_P]$. The reader should note that a vector $\vec{a}_1$ is used to conveniently represent the point (x_1, y_1, z_1). The mean square objective

function to be minimized by ICP is:

$$f(\vec{q}) = \frac{1}{N_P}\sum_{i=1}^{N_P} ||\vec{x}_i - R(\vec{q}_R)\vec{p}_i - \vec{q}_T||^2 \tag{2.2}$$

, where R can be found in Equation 2.1.

Let the center of mass of P be $\vec{\mu}_p$ and of X be $\vec{\mu}_x$ where,

$$\vec{\mu}_A = \frac{1}{N_A}\sum_{i=1}^{N_A} \vec{a}_i \tag{2.3}$$

The cross-covariance matrix $\sum_{px}$ of the sets P and X is given by Equation 2.4.

$$\sum_{px} = \frac{1}{N_P}\sum_{i=1}^{N_P}[(\vec{p}_i - \vec{\mu}_P)(\vec{x}_i - \vec{\mu}_x)^t] = \frac{1}{N_P}\sum_{i=1}^{N_P}[\vec{p}_i\vec{x}_i^t] - \vec{\mu_P}\vec{\mu_x}^t \tag{2.4}$$

The cyclic components of the anti-symmetric matrix $A_{ij} = (\sum_{Px} - \sum_{Px}^T)_{ij}$ are used to form the column vector $\Delta = [A_{23}A_{31}A_{12}]^t$. An anti-symmetric matrix A satisfies: $A = -A^t$. This vector is then used to form the symmetric 4 by 4 matrix $Q(\sum_{Px})$ from Equation 2.5.

$$Q\left(\sum\nolimits_{Px}\right) = \begin{bmatrix} tr(\sum_{Px}) & \Delta^T \\ \Delta & \sum_{Px} + \sum_{Px}^T - tr(\sum_{Px})I_3 \end{bmatrix} \tag{2.5}$$

The maximum eigenvalue of the matrix in Equation 2.5, given as the unit vector $\vec{q}_R = [q_0 \;\; q_1 \;\; q_2 \;\; q_3]$, corresponds to the optimal rotation.

The optimal translation is given by the vector: $\vec{q}_T = \vec{\mu_x} - R(\vec{q}_R)\vec{\mu_P}$.

Given d_{ms} as the mean square point matching error, the least squares quaternion explained above is denoted as Equation 2.6.

$$(\vec{q}, d_{ms}) = \mathcal{Q}(P, X) \tag{2.6}$$

With this small mathematical background presented, the steps performed by ICP can be elaborated, as done below.

- Let P be the first set of points with N_P points $\{\vec{p}_i\}$, and analogously X be the model with N_X supporting primitives such as points, lines or faces $\{\vec{x}_i\}$.
- Let $P_0 = P$, $\vec{q}_0 = [1, 0, 0, 0, 0, 0, 0]^t$ and $k = 0$. The algorithm iterates increasing the value of k, and sets the optimal rigid transformation in $\vec{q}_K$.

1. Compute the point correspondence between each point of the first set to a supporting primitive of the second set, by finding the supporting primitive with the smallest distance—it is assumed that closest points among each set correspond. This yields a new set Y such that $Y_k = C\{P_k, X\}$. C is a function that given a point, finds the nearest supporting primitive on another set. This step can have many variations, the point to point correspondence function corresponds to the point–point variation of ICP; another possible correspondence method is to find the smallest distance from a point to a plane, called point–plane, when faces are available in the model and not only points.
2. Compute the alignment transformation between Y and M, as: $(\vec{q}_k, d_k) = \mathcal{Q}(P_0, Y_K)$ —see Equation 2.6
3. Update the model by applying the computed alignment: $P_{K+1} = \vec{q}_K(P_0)$.
4. Iterate until the change in the mean square error falls below a predefined threshold $\tau > 0$. This is computed with the equation $d_K - d_{K+1} < \tau$.

ICP always converges, but sometimes to a local minimum, yielding an incorrect alignment.

It is important to note that ICP always converges. The convergence theorem of ICP can be found in [Besl and McKay, 1992]. Still, there is the chance the algorithm might converge to a local minimum, thus leading to an incorrect alignment. When the two sets of points are closely aligned, the risk of a wrong alignment is low since it is assumed that closest points between sets correspond.

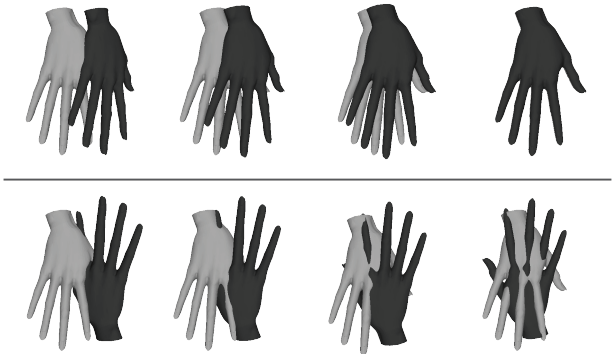

Figure 2.7: *Four frames of the alignment process of two point clouds using ICP. When the point clouds are closely aligned (top) a correct transformation is computed, but when their starting positions is significantly different, and incorrect transformation is yielded (bottom).*

Many variants of ICP have been proposed. They can be classified according to: (a) the selection of points in both meshes, (b) how points in both meshes are matched, (c) weighting of the corresponding pairs of points, (d) rejecting certain pairs based on looking at each pair individually or considering the entire set of pairs, (e) assigning an error metric based on the point pairs, and (f) minimizing the error metric. For more detail on each classification the reader might refer to [Rusinkiewicz and Levoy, 2001].

Variants of ICP.

2.5.2 Object Alignment

In the context of this work, object alignment is not part of the main tackled problem, although it is necessary to accomplish it correctly in order to track objects later. For this reason, a method that registers partially occluded objects in noisy scenes was chosen. In this section, such method for object alignment is elaborated.

Not part of the main problem, but necessary for tracking.

As mentioned in the definition of object tracking, the position and orientation displacement of an object is computed in two moments of time. Throughout several frames, an object can be tracked by iteratively considering two consecutive frames. As in all iterative problems, the initial case

is a special one; the need to identify the initial location and orientation of an object in a defined 3D space is called object alignment.

Definition: *Object Alignment*

OBJECT ALIGNMENT:
To determine at a moment of time the position and orientation of a object in 3D, through the use of sensor measurements.

Frame by frame detection is not used for tracking for two reasons.

It is possible to use a object alignment method for tracking, where the pose of an object is estimated in each frame of a sensor data feed, although this approach is not recommended. This practice is called *Tracking by Detection* as introduced by Lepetit and Fua [2005]. There are two reasons for not performing object detection in each frame instead of tracking.

Not near real-time performance.

1. Object pose estimation is usually not fast enough to work in real time. Using this method only once for initial pose estimation is acceptable, but afterwards a method closer to real-time is generally needed. For instance, the provided implementation of *Efficient RANSAC*, described flowingly, was tested on a GTX580 NVidia graphics card using a GPU accelerated version through the use of CUDA, and in average it took 5.6 seconds to perform pose estimation with 250.000 scene points and 29.000 model points.

Not relying on previous frames information leads to two different estimations in two similar and consecutive frames.

2. It is acknowledge that issues arise from performing detection in every frame ([Lepetit and Fua, 2005]). Due to the fact that information from detection in the past frames is not considered, a detection method might yield two very different positions in two similar and consecutive frames. This holds especially for stochastic methods, where small differences in the position or orientation of the object might have a large impact on the result. The effect is having jumps in the tracked path and orientation.

An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes [Papazov and Burschka, 2011]

The problem of determining the pose of a 3D object from data provided as range data is called surface alignment.

Papazov and Burschka [2011] present an algorithm for determining the pose of a 3D object in noisy and occluded scene, from data provided as non-segmented range data. This problem is also known as *surface alignment*. By providing this method with a digital representation of a model, in the form of a set of points and normals, the algorithm can determine the pose of several instances of this model in a scene.

The algorithm has an offline and online phase

Given a set $M = \{M_1 ... M_q\}$ of models and a scene S, the algorithm yields a set $\{(M_{k1}, T_1), ..., (M_{k_r}, T_r)\}$ of pairs composed by a model, such that it belongs to the set M, and a rigid transformation, which places the model in the estimated position and orientation in the scene. The algorithm is divided in two phases— *online* and *offline*—described in general terms below.

Offline Phase: Model Preprocessing

A 3D hash table of point descriptors is built.

Firstly a 3D hash table of point descriptors is built. The descriptor of oriented pair of points is computed as a quartet composed by: (a) the positive distance between the pair of points, (b) the angle between their normals, (c) the angle determined by the normal of the first point and the vector formed by both points oriented towards the second one, and (d) the angle determined by the normal of the second point and the vector formed by both points oriented towards the first one—see Equation 2.7. This descriptor serves as the key to store the pair of points in the hash table.

$$f(u,v) = \begin{pmatrix} f_1(u,v) \\ f_2(u,v) \\ f_3(u,v) \\ f_4(u,v) \end{pmatrix} = \begin{pmatrix} ||p_u - p_v|| \\ \angle(n_u, n_v) \\ \angle(n_u, p_v - p_u) \\ \angle(n_v, p_u - p_v) \end{pmatrix} \tag{2.7}$$

Online Phase: Recognition

Several steps are done in this phase, being the most important:

Enumeration of the steps of the algorithm

1. Computation of the octree of the scene.
2. Analysis of sampling strategy, in order to derive the number of iterations needed to recognize model instances with a certain success probability.
3. Uniform point pair sampling in the octree of the scene.
4. Normal estimation of point pairs.
5. Descriptor computation of point pairs, as done to model points in the offline phase.
6. Retrieval of oriented model points in the hash table, built in the offline phase, using as key the computed descriptors.
7. Hypothesis generation through retrieving the model stored in the hash table
8. Computation of the rigid transform that aligns both pair of points.
9. Removing false positives by using a non-maximum suppression.

For a more detailed explanation of the algorithm, the reader may refer to [Papazov and Burschka, 2011]

The method is efficient due to efficient structures of the scene (octree) and the point table (hash). Also because it does not work an all the scene points.

The method is efficient for several reasons: it works on an octree of the scene; it computes a finite, and relatively small, number of iterations; and it has a small number of point pairs in the hash table. The later property is due to the fact that only point pairs with a certain distance are considered, which reduces this process from $O(m2)$ to $O(m)$—being m the number of points of the model. The time of execution of the method grows linearly when the amount of points in the scene grows.

This algorithm was tested during the course of this work, and due to its efficiency and precise results, even on partially occluded objects, it is a good candidate for performing pose estimation—the initial problem of tracking. An example of pose estimation on test objects can be observed in Figure 2.8.

This method is used for initial pose estimation.

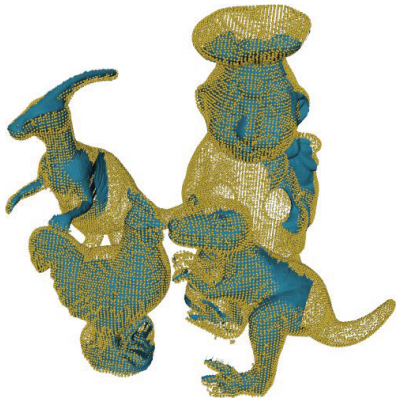

Figure 2.8: *Object initial pose estimation using efficient RANSAC 3D. The test scene depicts models (blue solid shapes) fitted into a set of points (yellow dots). From [Papazov and Burschka, 2011].*

2.6 Interaction with Everyday Objects: Existing Systems

In this section, systems used for studying interaction with objects are presented. Emphasis is made on their capabilities for tracking and sensing touches, as well as their deficiencies. The later give way to the proposed system in this work.

Systems used for studying interaction with objects are presented.

Early work on detecting the interaction with everyday objects, focused on being aware of which object the user was interaction with rather than how. Systems such as *ReachMedia* [Feldman et al., 2005] used Radio Frequency Identification (RFID) technology to detect objects held or touched by people. Although useful, these early systems do not allow full interaction to be studied. There is no information provided about the position of the object with 6DoF while us-

Early systems do not allow for full interaction to be studied.

ing 3D models, and at the same time information on touch interaction in specific areas. Not much can be inferred from user interaction with an object if actions such as moving, rotating or touching are unknown.

Following systems rely on RFID technology or sensors, which modify the physical affordances of the object.

Following work such as *Everyday Play* [Zhang and Hartmann, 2007] and *OnObject* [Chung et al., 2010] explore appropriation by attachment. Here everyday objects can easily gain life through the use of physical sensors. RFID tags and RFID readers identify objects and an accelerometer allows object gestures to be detected. In the case of Everyday Play, objects can easily become controllers for games by attaching a motion-sensing device. Similarly, OnObject allows gestures such as grabbing, shaking or swinging to be distinguished by attaching a motion-sensing device to arbitrary objects. Both systems allow basic motion information on interaction to be inferred; nevertheless, full interaction cannot be studied through these systems. The position of objects; their precise movement and rotation in space; and the touches people generate, cannot be sensed. The following presented systems aim at allowing to investigating a deeper level of touch interaction between people and everyday objects.

2.6.1 KinectFusion—[Izadi et al., 2011]

Creates the 3D model of a scene. Sets of points can be tracked and touches can be detected.

KinectFusion allows recreating the geometry in real-time of a scene through the use of a Kinect depth camera. Using ICP, subsequent set of depth images obtained by the Kinect depth camera are combined, and a dense point cloud of a scene can be computed. Objects can be segmented from the rest of the scene, and touch can be detected on their surface as they are detected. In order to detect touches, depth distance thresholding is used.

Since it has no model of the tracked objects, no semantic value can be assigned to movement or touches.

It is important to mention that, although touches are recorded, a way to assign semantic value to them is not presented. Considering KinectFusion does not store information about objects on the scene, rather than the scene as a whole surface, touches cannot be assigned to a specific object—not to mention specific areas of an object. Although objects are successfully tracked, due to the fact that

no model of the objects is known, the movements cannot be characterized as well. In Figure 2.9, a segmented object from the scene and touch recognition can be observed.

Figure 2.9: *An object being tracked (right) and touches being detected (left) by KinectFusion. From [Izadi et al., 2011]*

Having a model of the identified objects would enable applications to respond to changes in the object position, orientation, or to touches in specific areas. For instance, a system could interpret a touch in the upper half of a pen as one action such as increasing the TV volume, and in the lower part as another such as lowering the TV volume. In this work, tracking and touch interaction with semantic value is explored, taking the work of KinectFusion a step further in this regard.

A model of the object would allow to distinguish its direction, or which part is being touched.

Lastly it is important to mention that, due to the use of only a Kinect depth camera, objects do not require to be prepared in order for the system to work. Additionally, the system does not intrude in the user's normal activities while working.

Does not require object preparation

2.6.2 Invoked Computing—[Zerroug et al., 2011]

Invoked computing allows for people to re-appropriate objects and use them as computing devices. In a scenario where there is an incoming phone call in a person's home, the gesture of picking up a banana and putting it in her ear clearly indicates the intention of the user to answer the call. Invoked computing allows users to perform gestures and invoke computation in everyday objects, through

Invoked computing allows object re-appropriating and using them as computing devices.

the use of hidden directional speakers, microphones, and projectors—as seen in Figure 2.10. This allows for objects to serve as devices capable of outputting sound and video, and to receive input through touch.

Requires high preparation of the room. Allows tracking an touch detection.

This system allows objects to be tracked and touches to be sensed upon them. Although preparation for the system to work is high, due to mounting of the aforementioned indispensable hardware, the user does not perceive the system in the room while interacting with objects.

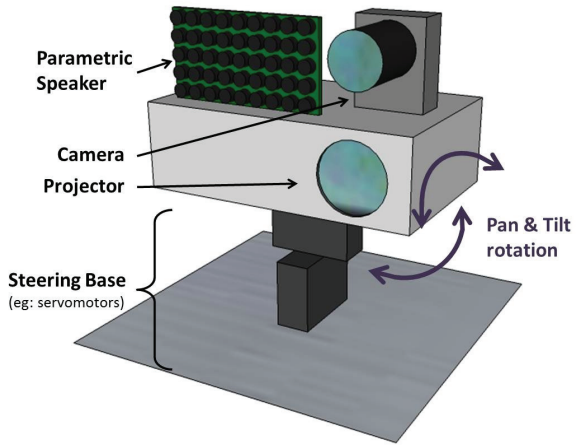

Figure 2.10: *Invoked Computing setup. A projector is mounted along with a camera and parametric speakers. The setup can rotate with all three degrees of freedom. From [Zerroug et al., 2011]*

Example: a pizza box as a laptop computer.

Another presented example is the case where a pizza box is opened and, given its similarities with a laptop computer, it can operate as one by displaying a keyboard in the lower cover and outputting images in the top cover. The conceptual difference between Invoked Computing and IUIs is that a person uses everyday objects as devices themselves and not as device controllers. As with KinectFusion, by not having a characterization of objects on the scene, interaction is limited to predefined scenarios.

2.6.3 DisplayObjects—[Akaoka et al., 2010]

DisplayObjects allows everyday objects to be augmented, with the goal of using them as rapidly-built prototypes. Objects are tracked through the use of a VICON system, and a series of projectors augment the surface of the object with a desired interface. Users of this system can interact with the constructed prototypes by pointing at specific areas of the object or by touching its surface. Gestures such as pointing, tapping, clicking, dragging, pinching, and wiping can be recognized by the VICON tracking system through the use of infrared reflective markers in the user's finger. The latter are used as well for object tracking.

Meant as a system for rapidly building prototypes. Allows tracking through the use of a VICON tracker.

Figure 2.11: *DisplayObjects setup, composed by: augmented fingers and objects by VICON markers; and VICON tracking cameras*

The aim is to be able to rapidly prototype by augmenting physical mockups, such that users can experience the functionality of a device in their hands. To create a prototype, firstly the user constructs the physical model, augments it with infrared reflective markers on its surface, and then registers the model in the system. From a virtual pallete, also tracked and augmented by projected images, the user grabs different UI elements and places them onto the surface of the object. Interaction with the object needs to be explicit by the creator through connecting input elements with output elements. This can be done in the Quartz Composer on a laptop, on the digital palette, or on the physical

IR reflective trackers are added to objects, and they are painted in a solid color. UI elements can be displayed onto objects in order to simulate a UI.

object itself. Output elements are by default reproduced video files, but the system is flexible and allows subpatches to be used, and since these run Objective-C code, virtually anything can be outputted.

Easy to use, but requires extensive object preparation and has a high cost.

The main advantage of DisplayObjects is that users with no previous experience can easily and quickly create physical prototypes and paint a user interface on top of them without the need of programming the interface. The user can get a vivid feeling on the physical feel and the look of the final product by the use of the generated prototypes. A major limitation of this system is that small infrared reflective spheres must be added to the object surface and the users' hands, in order for tracking and touching to work. This might interfere with the interaction between the user and the object. In addition, the cost of implementation of this system is relatively elevated.

2.6.4 Wear Ur World (WUW)—[Mistry et al., 2009]

It is easier to augment than to modify objects in order to allow interaction.

Mistry et al. [2009] state that "it is impractical to modify all physical objects and surfaces into interactive touchscreens" but that "it is possible to augment environment around us with visual information". With this in mind, a computer-vision based wearable and gestural information interface is constructed. *WUW* aims at augmenting everyday objects such that they can display information interactively, as well as allowing the user to interact with the object directly.

WUW consists or a camera and projector mounted in the user's body. Gestures are recognized and information can be displayed.

Technically, WUW consists of a camera and a small projector mounted on a hat as seen in Figure 2.12. The user wears markers on her fingers in order to allow the system to detect performed gestures. The projector is used to output information onto walls and other physical objects. Some applications presented include: displaying a map on a wall, and allowing the user to interact with it through clicking or pinching; taking a picture by making a frame gesture with both hands; and displaying a video on a special area of a sheet of paper. Fingertips and objects where images can be projected on are tracked by the system, and also touches on interactive surfaces are recognized.

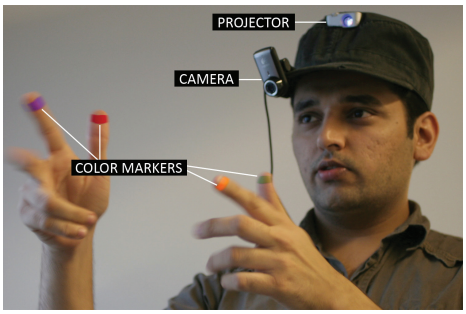

Figure 2.12: *Wear Ur World setup. A wearable camera identifies user interaction with objects and a wearable projector augments them.*

WUW allows controlling everyday objects through direct object manipulation on them, but does not allow using them as controllers for other devices, as proposed in IUIs. Passive objects gain life through the display of information on top of them and the user can interact with the displayed information by touching, moving the augmented object, or by performing iconic gestures in mid air. The main disadvantage of this system is the intrusiveness it imposes on the user.

Allows for object manipulation, but not using objects as controllers. The system requires large preparation and is intrusive for interaction.

2.6.5 dSensingNI - Depth Sensing Natural Interaction—[Klompmaker et al., 2012]

dSensingNI - Depth Sensing Natural Interaction is a framework for detecting mutlitouch and tangible interaction with arbitrary objects. Interaction with arbitrary physical objects allows for users to modify their everyday environments using their everyday tools, providing at the same time high awareness for human and social interactions. The goal of having such framework is to enable application developers to interpret tangible interaction techniques using arbitrary objects and surfaces.

Framework for detecting interaction with objects.

Having as input source a depth camera, dSensingNI tracks fingers and palms of hands on vertical surfaces, allowing interaction for grasping, grouping and stacking objects. The basic requirements set in mind at the time this frame-

Allows detecting moving, grouping and stacking of objects, as well as touches. It uses a Kinect camera.

work was built, was to detect finger touches on arbitrary surfaces; detect hand, arm and finger gestures above and in front of a display; allow to work on arbitrary tangible objects; detect gestures used in everyday life; and to be robust in terms of occlusion.

Arms and fingers are detected as in OmniTouch

The first step dSensingNI carries out is to detect arms and fingers on the scene. By storing a background image, hands on top of a surface can be segmented in the same way it is done in OmniTouch. By approximating the blobs on top of the surface to polygons, the hand and arm contours can be deducted. Fingers are detected as the vertices of the identified hand. To determine the pointing direction of the hand, the centers of the hand and arm clusters are taken into account. The use of depth image offers better background separation.

Tangibles are detected in top of a surface

The next step is to detect tangibles on top of the surface. As in OmniTouch, by subtracting a background depth image, objects on top of a surface can be individualized. Using the contour of such objects, oriented bounding boxes are created around them such that width, height, depth, and position of the object are set. Following, the fingertips of the detected hand are determined, so that the touch points of the hand are made available. In Figure 2.13 a fingertip and two objects detected by dSensingNI can be seen.

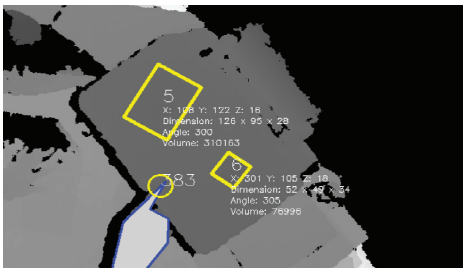

Figure 2.13: *A fingertip and two objects recognized by dSensingNI.*

dSensingNI can handle temporary occlusions by hands.

dSensingNI can handle occlusion of objects by arms and hands. After detecting the arm and hand contours in the scene, they can be subtracted from the current image and replaced by pixel-values of previous frames. This way the framework does not lose track of objects even if they are

being occluded, assigning it a constant ID through various frames.

Tracked objects are assigned an ID, maintained even with temporary occlusions.

In the case the user grabs the object, it detects it as it disappears and it is assigned to the occluding hand. When a tangible is assigned to a hand, the framework can detect stacking and merging with other tangibles. By storing the components of the new assembled tangible, it allows disassembling and tracking of the components with the same IDs prior to the merging or stacking.

Interaction is limited to table tops. Tracking is performed in 2D, although staking of objects is also supported. Mid-air gestures are also recognized.

All in all, dSensingNI is a useful tool for basic interaction with tangibles. Nevertheless it has major limitations regarding tracking: it is limited to 2D interaction. Even if it can track stacked objects knowing its height, it cannot track movements in 3D. If the objects are lifted from the surface, tracking is lost. Touch detection is also limited since touches only on the top surface of the object can be recognized. On the other side, a useful characteristic of this framework, although not directly related to everyday object manipulation, is that it enables detection of freehand gestures in mid-air. The framework is easily integrated into other developed software through the use of the TUIO library[3].

2.6.6 Using a Depth Camera as a Touch Sensor—[Wilson, 2010]

Touches are detected by recording the depth of a surface and comparing depth values of fingers.

In this work, the use of depth cameras as touch sensors is explored. Here a technique similar to background image subtraction is used—an initial snapshot of a surface per-pixel depth is taken and used to compare it to the depth of a detected finger. By using a depth threshold, the hovering of a finger over a surface can be distinguished from a touch, as shown in Figure 2.14. This approach enables touch on non-flat surfaces, and, at the same time, brings touch to objects outside the classical touch enabled screens. The main disadvantage of this method is that precision is not as accurate as in traditional touch screens.

[3] http://www.tuio.org/

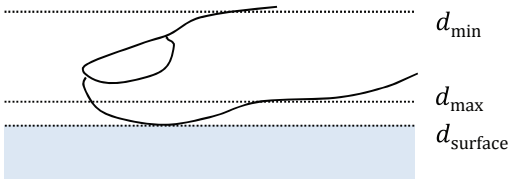

Figure 2.14: *Threshold used to detect finger touches on a surface at a predefined depth. From [Wilson, 2010].*

2.6.7 OmniTouch—[Harrison et al., 2011]

The derivate of depth images is used to detect cylinders—the pattern for a finger.

Yet another approach on touch detection is presented by Harrison et al. [2011], where computer-vision algorithms are applied to depth images in order to detect touches. The derivate of the depth image is computed in both axis using a 5 by 5 pixel window. In order to find finger candidates, vertical slices of cylinder-like objects are searched. A candidate is a region that shows a steep positive derivate, followed by a smooth region and finally a steep negative derivate. This pattern can be observed in Figure 2.15. The advantage of this approach is that it does not require an initial depth snapshot, allowing it to function with moving target objects.

Tracking is performed to adjust the projection of information on objects where touch is detected.

Since information is displayed on objects, touches are characterized according to the touched projected image. Although augmented objects where information is displayed are tracked, the observed scene is not treated as a 3D set of points. Only basic tracking is performed in order to correct projections when the user moves an object. It is important to mention that the set-up of the system is intrusive to the user, due to the fact that the latter has to constantly wear the system on its shoulders.

2.6.8 iCons—[Cheng et al., 2010]

Fiducials are used to augment objects and use them as auxiliary input devices.

iCons allow everyday objects to become input controllers for computer programs, through the use of fiducial markers for tracking object movement. Cheng et al. [2010] state that everyday objects might not be adequate as major in-

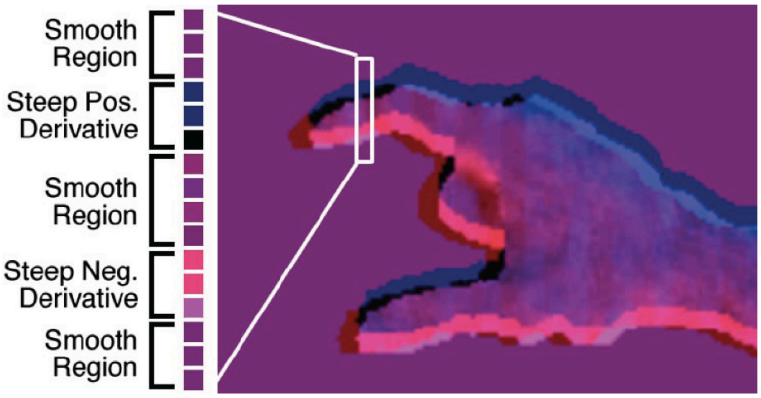

Figure 2.15: *Pattern on depth derivate used to identify a finger candidate in [Harrison et al., 2011]. A smooth region is followed by a region with a steep positive derivate, a smooth region, a region with steep negative derivate and finally another smooth region.*

put devices because they are not designed for this purpose. Nevertheless, some objects can be used as auxiliary input devices in a multi-task work environment.

Regarding the setup, two different possibilities are presented: using fiducial markers attached on top of objects; and placing a camera on top of the desk, using a semi-transparent table and cameras on the ground facing the desk surface from bellow. Both set-ups include attaching markers to objects and detecting them through the use of cameras. The approaches differ on how they intrude the user's daily tasks on their desk.

The camera that detects the markers can be set on top of the objects or on the bottom of a transparent surface.

Gestures such as clicking, rotating and dragging are recognized. To prevent triggering actions when the user is using the object for its original intended use, a Time To Live (TTL) mechanism is implemented. If the gesture is performed between a certain threshold time values, it is recognized as a gesture, otherwise ignored. For instance, covering an object is not interpreted as a click on its surface if the user places its hand on top for an extended period of time. Gestures are associated with actions on a computer using an application designed for this purpose.

A TTL mechanism prevents accidental triggering.

Scenarios include changing the music volume or switching between windows.

The goal of *iCons* is to turn everyday objects into auxiliary controllers. Some of the presented scenarios include manipulating the music volume or switching between windows. It is acknowledged that critical or sensitive tasks, such as deleting objects or sending e-mails, should not be controlled by these auxiliary controllers. False triggering could have major undesired repercussions and should thus be avoided.

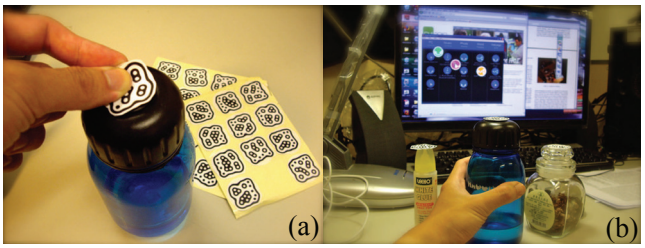

Figure 2.16: *iCons setup. On the Left: a user adding fiducial markers to an everyday object. On the Right: a user controlling an application through the use of the prepared object. From [Cheng et al., 2010]*

The system tracks in 2D and detects objects being lifted, but there is no 3D tracking. No touches are recognized.

The system allows a set of restricted gestures to be performed using everyday objects, bounded to the object affordances. Interaction with the augmented object is limited both on tracking and touching. Although rotation and panning gestures are tracked, once more this is bounded to the 2D plane. The system is also capable of detect objects being lifted, but it does not track their position in 3D. At the same time, touching is not detected on the object. The click action is restricted to raising and lowering the object. These limitations arise from the use of a 2D camera and fiducials, which makes difficult having information on the object as a volume in space to track or detect touches. Even if simplistic in its implementation, this method does not allow studying the complete interaction between people and everyday objects as it is intended in this work.

2.7 Literature Design Space

As a summary of the reviewed literature, and to specify the different explored dimensions in this work, a design space was created. Each presented system is depicted by a set of polygons connected by lines. The position of these indicate the values of each dimension that the system comprehends. The included dimensions are:

A design space places this system in the context of previous work.

- **Tracking**: the capacity of the system to track objects, and whether tracking includes both position and orientation. For each of the two latter sub-dimensions, there is the division of whether the system is capable of adding semantic value or not to tracking. If the system can only report the position and orientation, along with its changes, it is considered that there is no semantic value added. On the other hand, if the system is capable of interpreting movements or rotations in order to perform actions when they are generated, then it is considered that there is added semantic value.

- **Touch**: the ability of the system to recognize touches on objects, and whether these have a semantic value or not. Adding semantic value to touches allows for the user of the system to associate an action to a touch performed in a determined location of an object, it comprehends more than only reporting that a touch was performed on the surface of an object.

- **Internal representation**: how the system treats objects internally, whether it is as 3D or 2D models. If the system uses depth data but still treats objects as 2D models, the dimension value is placed in between both possible values of this dimension

- **Intrusiveness**: describes how much the user needs to change its interaction style with an object due to the presence of the system. Divided into three possible values—low medium and high. Low indicates that the user is not intruded at all, or that it is barely intruded; medium indicates that the user needs to change its habits in order to interact with the object;

Design space dimensions: tracking of both position and orientation, touch detection and the ability to assign semantic value to touches, internal representation of objects (2D and 3D), intrusiveness the system imposes on the user, degree of preparation of the tracked objects and the cost of deployment.

and high indicates that the user must redefine most of the aspects of its interaction with the object because of the system presence. The latter consists of, for example, limitations imposed by specific ways to hold and touch objects, as well as whether the user should wear a device.

- **Preparation of the objects**: indicates how much the objects must be prepared in order to be detected by the system: low is used for objects with little or no preparation; medium for slight modifications to objects; and high when the object changes its physical affordances because of the system presence.
- **Cost of deployment**: divided into three categories: low, which represents solutions with costs under a couple of hundreds of dollars; medium is dedicated to systems that require components raging up to a thousand of dollars; and high is reserved for systems that require more than a thousand dollars.

The object tracking and touch sensing system is located in the presented Design Space of Figure 2.17 as a system with low cost, low intrusiveness and low object preparation. Both touches and tracking have semantic value, as this requirement is one of the fundamentals aspects of the system. Lastly, the representation of the object is done as a 3D model.

Related systems have been presented. Based on deficiencies, a new system for studying everyday interaction emerges.

Systems that enable studying interaction with objects have been presented in this section. Some of these, such as KinectFusion, Invoked Computing, Wear Ur World or dSensingNI, provide similar functionalities or use similar technologies as the presented system. Nonetheless, none of them provide a solution, with a low cost of deployment, that allows to assign semantic value to movement performed to the objects with 6DoF and to touches in their surface, without intruding in the user's space. For this reason, a new system for studying everyday object interaction has been envisioned.

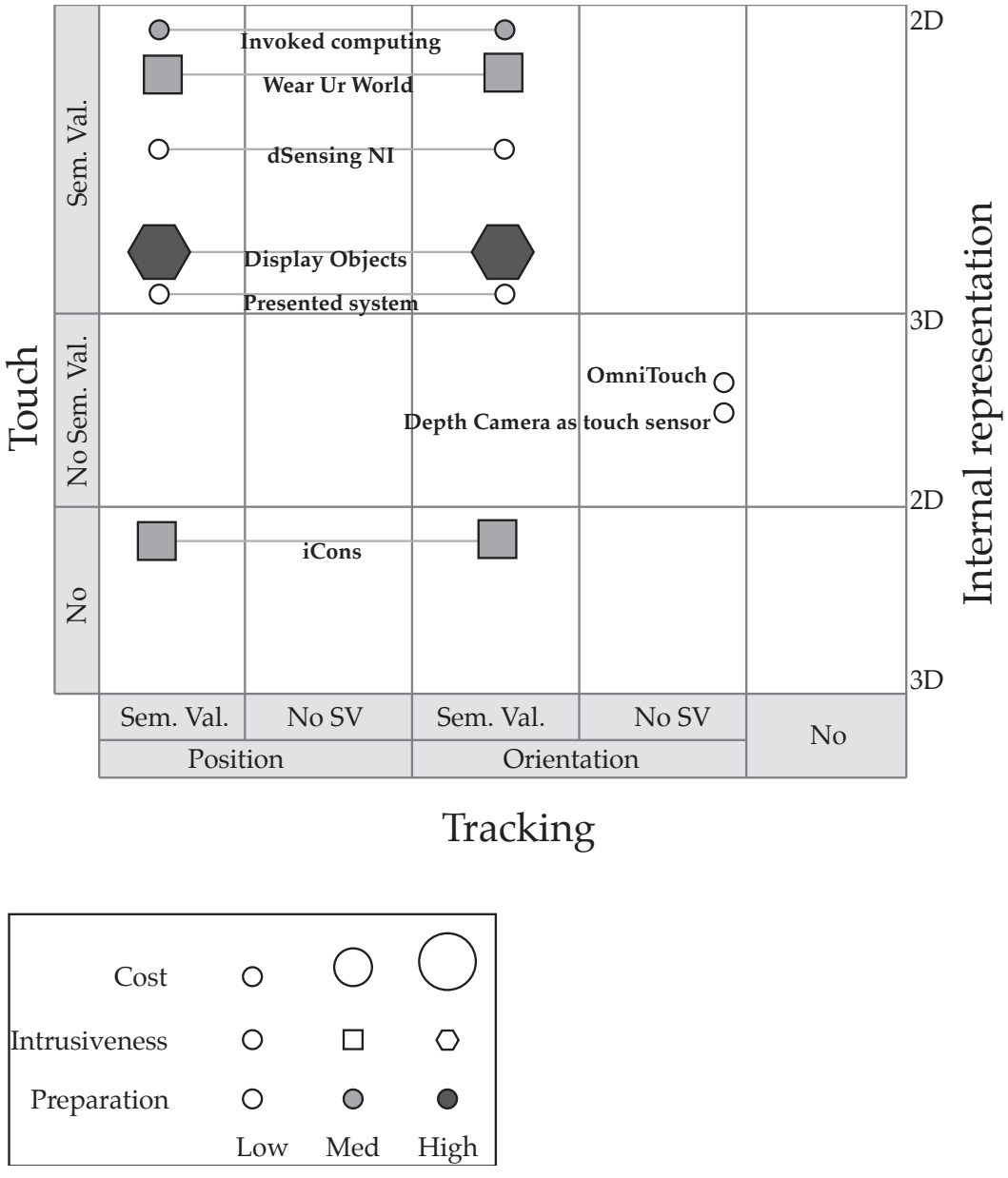

Figure 2.17: *A Design Space for the literature review.*

Chapter 3

Object-tracking and Touch-sensing System Implementation

In this chapter, the system is presented both from a conceptual and a technical implementation point of view. Firstly the system structure is presented from an abstract perspective, elaborating on all the steps involved—raging from data acquisition to yielding the object position in 3D and touch points. Following, the technologies used in each step are presented, along with technological issues encountered on the process of developing the system.

The system is presented from a conceptual and an implementation perspective.

The main characteristic that sets this tracking system aside is its internal object representation. Having a 3D model of the tracked object allows for characterization of its position, orientation and of the touches users perform in its surfaces. At the same time, the system has a low cost of deployment and does not intrude the user typical interaction with objects.

A 3D model of the tracked object allows for characterization of user interaction. The system is unobtrusive and inexpensive.

3.1 3D Space Definition

Space defined by the presented system output.

The output of the presented system is, in part, the position and orientation of tracked objects in 3D. This data is reported in a defined 3D space, described in this section. With respect to the point of view of the Kinect camera, the Z axis grows towards the back of the depth camera, the Y axis grows upwards, and the X axis grows towards the right. The selection of this coordinate system follows the conventions defined by the Kinect camera itself. Axes are depicted in Figure 3.1. With respect to the distances, one reported unit corresponds to one millimeter in real world space. The origin of the space is the same as the one provided by the depth camera, in the case of the Kinect this is its lens.

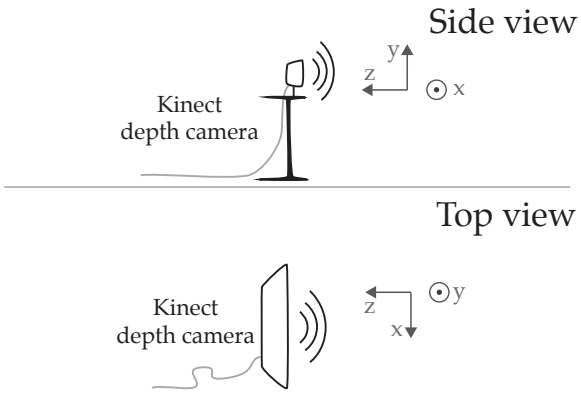

Figure 3.1: *Axes of to the space defined by the presented tracking system.*

3.2 Conceptual Model

Two phases are distinguished.

The tracking and touch detection process is divided into two disjoint phases—an offline phase where objects soon-to-be-recognized are scanned, having as output their 3D model; and an online phase, in which a depth camera is used to sense a real world scene and track the scanned objects. The output of the second phase is the object position and orientation in the scene throughout several frames.

3.2.1 Offline Phase

The offline phase involves scanning the object and obtaining its 3D model.

Firstly, the 3D model of the object to be tracked needs to be acquired. It must include points, faces, and normal vectors of the scanned object. Although in order to track objects only points are later used, both faces and normal vectors are necessary for the initial pose estimation process. In Figure 3.2, the 3D digital scan of an object that includes points and triangular faces can be observed.

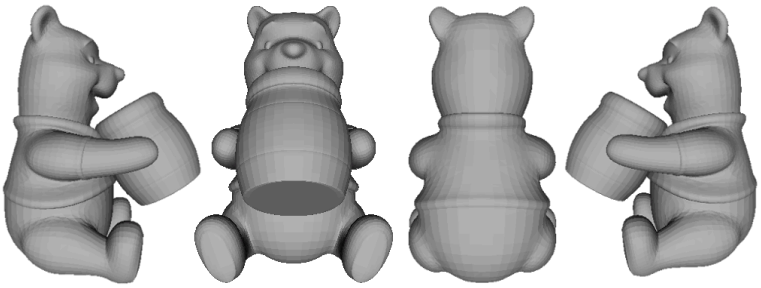

Figure 3.2: *3D model of a toy bear holding a pot, composed by points and triangular faces. From left to right: right side view, front view, back view and left side view of the model.*

3.2.2 Online Phase

Steps of the online phase: data acquisition, data pre-processing, plane segmentation, initial pose estimation, object tracking and surface touch detection.

The series of steps performed in this phase are outlined below and elaborated in this section individually.

1. **Data acquisition**: the medium is sensed and raw data is acquired.

2. **Data pre-processing**: raw data obtained from the sensing device is transformed into a 3D scene.

3. **Plane segmentation**: a previous optional step, used to reduce the amount of processed points in each frame. The system allows the user to select three points on screen in order to determine the plane where tracked objects are placed. All points below this plane are discarded in subsequent frames.

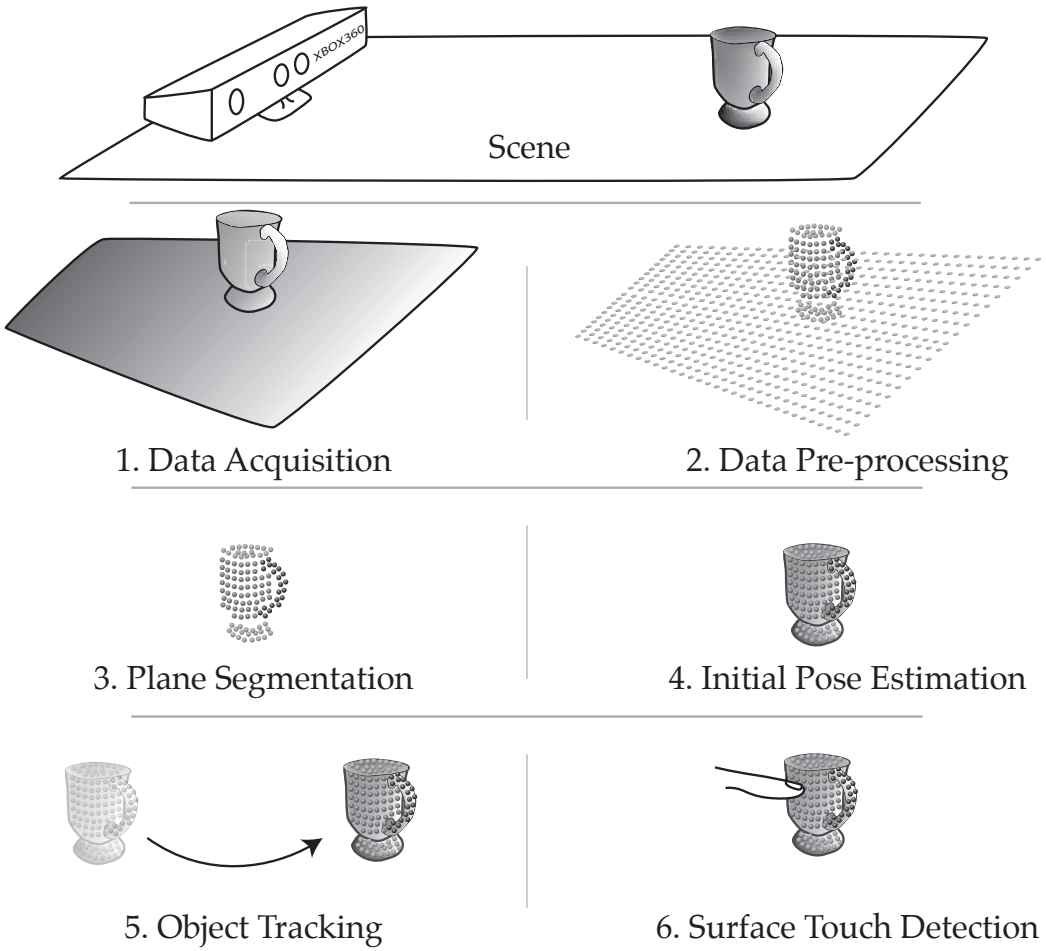

Figure 3.3: *Steps of the online phase.*

4. **Initial pose estimation**: the scanned object is found in the scene, and its pose is estimated as best as possible.

5. **Object tracking**: the detected object is tracked. Its position and orientation are updated and its movement with 6DoF in the real world is accompanied. This step is continuously performed for each detected frame.

6. **Surface touch detection**: objects between the tracked object and the depth camera, which are in the vicinity of the tracked object, are characterized as fingers. Their tip is detected in order to determine contact with the object. This step is also continuously performed for each detected frame.

Data Acquisition

Obtaining a RGBD image of the scene through a depth camera.

Data is acquired through the use of a depth camera. The information provided in each frame is an RGBD image composed of a two dimensional matrix of pixels, in which each pixels contains color information; provided as values for the red, green and blue channels; and the depth at which that pixels is sensed in the scene.

Data Pre-processing

Computing a 3D point cloud of the scene.

One of the innovative and distinguishing aspects of this tracking system is how it models the scene and the tracked objects. Both are modeled as a set of points in a 3D defined space. For this reason, the RGBD input image is converted into a point cloud. In order to achieve this, an image perspective unprojection can be performed on the raw input data, using the depth channel as the depth buffer. For details on the projection/unprojection models, the user can refer to Appendix A.

The obtained point cloud is adapted to match the space of the tracking system.

The 3D space of the computed point cloud must match the 3D space of the presented tracking system—as specified in Section 3.1. This might require a rotation of the scene and scaling, depending on the data provided by the input device.

Plane Segmentation

A plane is detected from three user-defined points

In order to perform plane segmentation and remove all points below it, the same technique as described in Liu et al. [2012] is used. By determining three points in the scene that belong to the plane, it can be modeled and thus later all points below such plane can be removed.

Method for obtaining a plane from three points.

Given the three obtained points A, B and C, two vectors are computed—$\vec{v_1} = \vec{AB}$ and $\vec{v_2} = \vec{AC}$. Given two points, the general form of a vector composed by them can be found in Equation 3.1.

Method for obtaining a plane from three points.

Given the three obtained points A, B and C, two vectors are computed—$\vec{v_1} = \vec{AB}$ and $\vec{v_2} = \vec{AC}$. Given two points, the general form of a vector composed by them can be found in Equation 3.1.

$$Given\ P = [p_1, p_2, \cdots, p_n],\ \ Q = [q_1, q_2, \cdots, q_n]$$
$$\vec{PQ} = Q - P = \begin{pmatrix} q_1 - p_1 \\ q_2 - p_2 \\ \vdots \\ q_n - p_n \end{pmatrix} \tag{3.1}$$

Two vectors defined by three non co-linear points define a unique plane.

Together $\vec{v_1}$ and $\vec{v_2}$ determine a unique plane. By computing their dot product, as shown in Equation 3.2a, the normal vector of such plane is obtained; by computing their cross product, as shown in Equation 3.2b, the distance from the origin to the plane is obtained. Such plane P can be expressed as $P = ax_1 + bx_2 + \ldots + nx_n + d$, where $(a, b, ...n)$ is its normal vector and d its distance to the origin.

The distance from a point to a plane reveals whether the point belongs to the plane or to one of the subspaces it defines.

Through this representation of the plane, it is trivial to determine whether a point belongs to the plane or not, and in the latter case, to which subspace defined by the plane the point belongs to. By simply computing the minimum distance from the point to the plane as their dot product, one can differentiate three cases: when the distance is positive, it belongs to one of the defined subspaces; when it is zero, it belongs to the plane; and when it is negative, it belong to the remaining subspace.

How to compute a the cross product and dot product of two vectors.

Given the vectors $\vec{a} = (a_1\ a_2\ \ldots\ a_n)^T$ and $\vec{b} = (b_1\ b_2\ \ldots\ b_n)^T$, let $(i\ j\ \ldots\ n)^T$ be the basis vectors of $\vec{a}$ and $\vec{b}$. Both vectors can be expressed as the sum of three orthogonal components parallel to the standard basis vectors, such that $\vec{a} = (a_1 i + a_2 j + \ldots + a_n n)$ and $\vec{b} = (b_1 i + b_2 j + \ldots + b_n n)$. The dot product $(a \cdot b)$ and the cross product $(a \times b)$ of the vectors $\vec{a}$ and $\vec{b}$ are defined as in Equations 3.2a and 3.2b respectively.

$$(a \cdot b) = (a_1 i + a_2 j + \ldots a_n n) \times (b_1 i + b_2 j + \ldots b_n n) \tag{3.2a}$$
$$(a \times b) = \sum_{i=1}^{n} a_i \times b_i \tag{3.2b}$$

Initial Pose Estimation

Once the 3D model of the object is available and the data from the sensor device has been adapted, the object pose can be estimated in the scene. Although ICP can be used for pose estimation, the time required and the precision of the result would be unacceptable. Considering that the scene contains a large amount of vertices, typically with an upper bound of 200.000 for scenes obtained by the Kinect, the time required to find a fit would be extremely large. Additionally, the reader might recall that ICP converges to a local maximum when no good initial transformation is provided. Due to the fact that no estimation can be provided, the algorithm most likely converges to the wrong result.

ICP is not adequate because there is no information on the pose of the object on the scene. Using ICP on the whole scene would lead to convergence to a local minimum.

Papazov and Burschka [2011] provide and efficient and effective object pose estimation method. As mentioned in Section 2.5.2, this method computes in an offline phase a descriptor for a set of point pairs from a 3D model of an object. Later in an online phase it computes the same descriptor for pairs of points in the scene point cloud, matching them with the descriptors from the model. When matches are found, hypotheses are generated and later verified. This check is performed by fitting the model according to the point pair matches and checking for outliers.

Efficient RANSAC is used for initial pose estimation.

Object Tracking

Once the initial pose of the object is estimated (Figure 3.4/1), the system is ready to track it throughout different frames. As a first step, the number of points used for tracking is reduced—points that do not belong to the object are meaningless for tracking. The object is placed in the scene 3D point cloud using the initial pose estimation, and its bounding box is used to crop the points that are not part of the object on the scene (Figure 3.4/4a). A small tolerance value is added such that the points in the vicinity are also included when tracking. When the object is moved, its points are not cropped in the next detected frame. This set of points corresponds to the *model* point cloud that later the ICP will use as input.

By considering only the scene points inside the bounding box of the registered model in the previous frame, reduces the amount of processed points for tracking. This is used as the *model* for ICP.

CROPPING TOLERANCE VALUE ON THE SCENE POINT CLOUD AROUND THE MODEL:
Tolerance value for cropping the point cloud around fitted model

Parameter: *Cropping tolerance value on the scene point cloud around the model*

ICP yields better results when both the template and the model are the same point cloud, but differ only in their position and orientation. In order to meet this criteria as much as possible, the set of points from the fitted 3D object that face the sensing device are segmented (Figure 3.4/4b), later used as the *template* on ICP. To test whether a point of the model belongs to this set or not, the same approach as described in Section 3.2.2 - Plane Segmentation is taken.

The front facing side of the registered model is used as the *template* for ICP.

At this stage, both the *model* and *template* are ready to be processed by ICP (Figure 3.4/5). Due to the fact that the object has been previously found and registered on the scene in the previous frame, the template position and orientation is similar to the one in the real world, hence to the one of the registered model. Once ICP runs, it yields the transformation that fits the provided template into the model (Figure 3.4/6).

ICP is executed. Since two consecutive frames are similar, it yields the correct transformation.

The next step is to apply the transformation to the digital model of the tracked object and update it on the scene acquired by the depth camera. Even with strange objects placed in between the camera and the real world tracked object, ICP is robust to a certain extent and it yields a correct transformation. At this stage, the system is ready to process touches on the surface of the object.

The transformation is applied to the model, overlaying it on the point cloud of the scene in the correct position.

Surface Touch Detection

The reader might recall that the bounding box of the fitted model is used to crop the input data from the depth camera. As a consequence, only fingers that are in the near vicinity of the object can be detected. This can be adjusted through the parameter *Cropping tolerance value on the scene point cloud around the model*.

Points belonging to the finger are cropped inside the bounding box of the registered model on the scene.

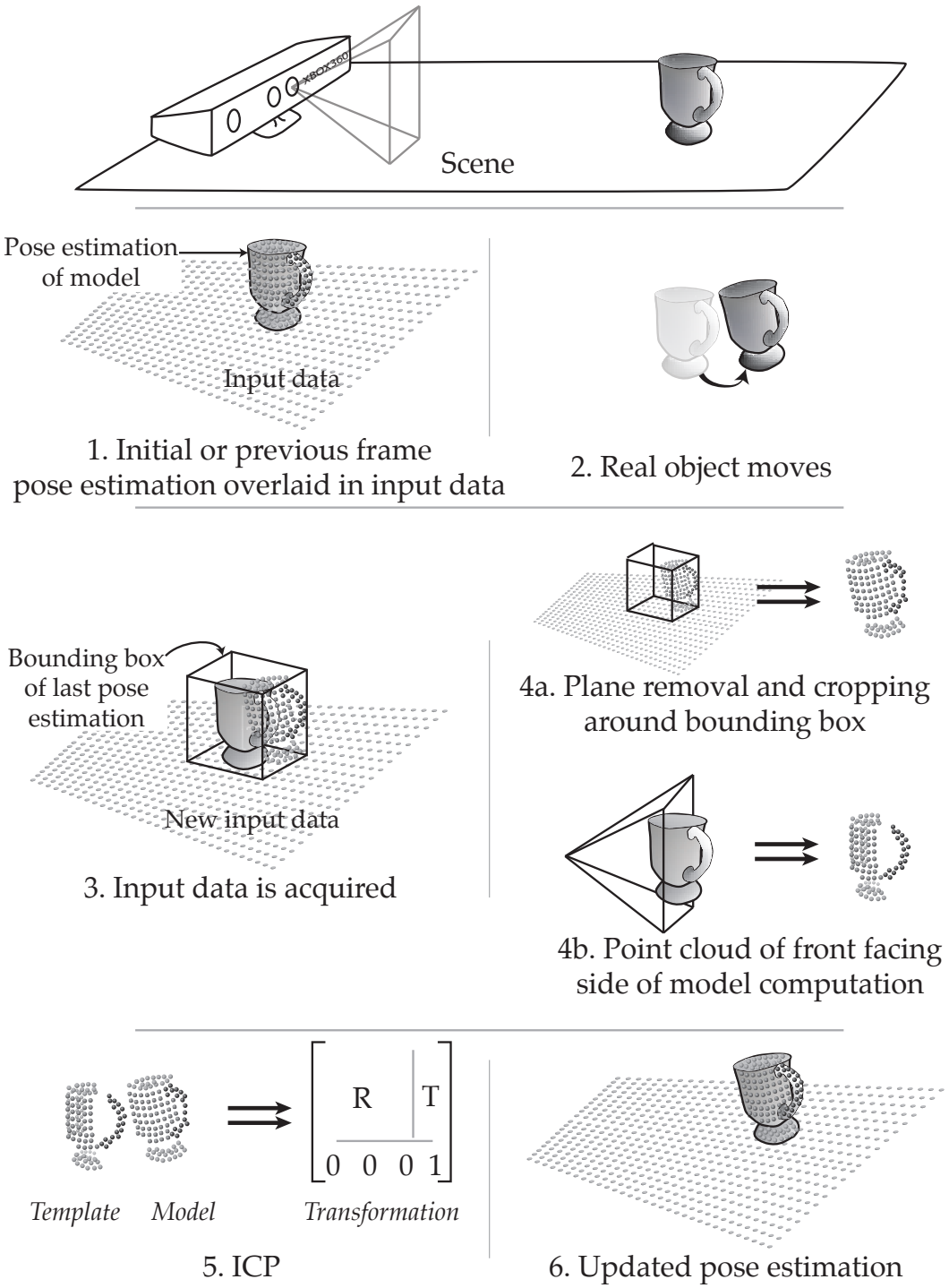

Figure 3.4: *Steps of tracking.*

Color filtering is used to segment the model points from the hand points.

The first problem is to determine which of the points belong to the tracked object and which to the user hand. In this system, a simple but effective approach was taken—color filtering. Since the model has a solid color, it can be segmented by analyzing the color value of its points. Taking advantage that the object is 3D scanned in order to obtain its digital model, its solid color can also be easily determined without adding further steps.

The HSV model is robust to changing light conditions. A threshold tolerance value is used for this purpose.

Light conditions on the input scene might vary, as well as the tracked object might create shadows on itself. In order to have a more robust color filtering, the Hue, Saturation and Value color model (HSV model) is used. The color of the detected scene points is compared to the solid color of the model. A small tolerance value is taken into account when comparing H, S and V values as mentioned in the parameter *HSV tolerance threshold*

Parameter: *HSV tolerance threshold*

HSV TOLERANCE THRESHOLD:
Hue, Saturation and Brightness tolerance values used when comparing solid color of the object to the color of the object points in the detected scene

Hand points are projected into a 2D image.

Once the fingers are segmented from the tracked object (Figure 3.5/2), they are projected into a 2D binary image (Figure 3.5/3). For details on the projection/unprojection models, see Appendix A. With simple computer vision algorithms, a single finger can be identified. The system is at the time limited to detecting only one finger, thus multitouch interaction is not recognized.

Convex hulls are computed, the largest one is considered as the finger.

In order to detect the fingertip, an approach similar to the one presented in *dSensingNI* Klompmaker et al. [2012] is used. Firstly, the contours formed on the binary image obtained from the 2D projection of the finger of the hand are detected (Figure 3.5/4). Finding contours is analogous to finding clusters of points on the input image. It is assumed that the largest convex hull corresponds to the finger projection. The minimum hull size can be adjusted through the parameter *Minimum hull size*.

MINIMUM HULL SIZE:
Minimum hull size value, expressed in pixels, while detecting hulls on the projection of the user's hand finger.

Parameter: *Minimum hull size*

Following, the center of the largest convex hull is computed as the weighted sum of the coordinate of each point. A good estimation for the fingertip is the most distant point of the convex hull center, although this point does not always correspond to the fingertip. It might be the case that the most distant point corresponds to the opposite end where the fingertip is, located actually towards the palm of the hand. Assuming that the user enters the tracked object bounding box from the sides, the furthest point, which at the same time is the most distant from the edges, corresponds to a better estimation of the fingertip (Figure 3.5/5).

The center of the largest hull is computed and the fingertip is identified.

As a last step, the tracked model is also projected in the same camera model as the finger was projected (Figure 3.5/6). The point found as the fingertip in the projection of the finger, corresponds to the touched point on the projection of the model (Figure 3.5/7). The depth of the point in both projections is retrieved and compared, if their difference is small enough for the finger to be touching the model, a touch is reported with its location in 3D. Such distance can be adjusted through the parameter *Fingertip touch minimum distance threshold*

A 2D projection of the model is computed. Depth values are used to distinguish finger touches distinguished from finger hovering.

FINGERTIP TOUCH MINIMUM DISTANCE THRESHOLD:
Minimum distance from the finger to the object in order to recognize an overlapping of both as a touch

Parameter: *Fingertip touch minimum distance threshold*

3.3 Implementation

In this section, the implementation steps taken in order to build the tracking and touch sensing system are detailed, along with encountered problems and their respective solutions. Firstly a series of technical prerequisites are presented followed by the online and offline phases of the system.

The implementation of the system is presented.

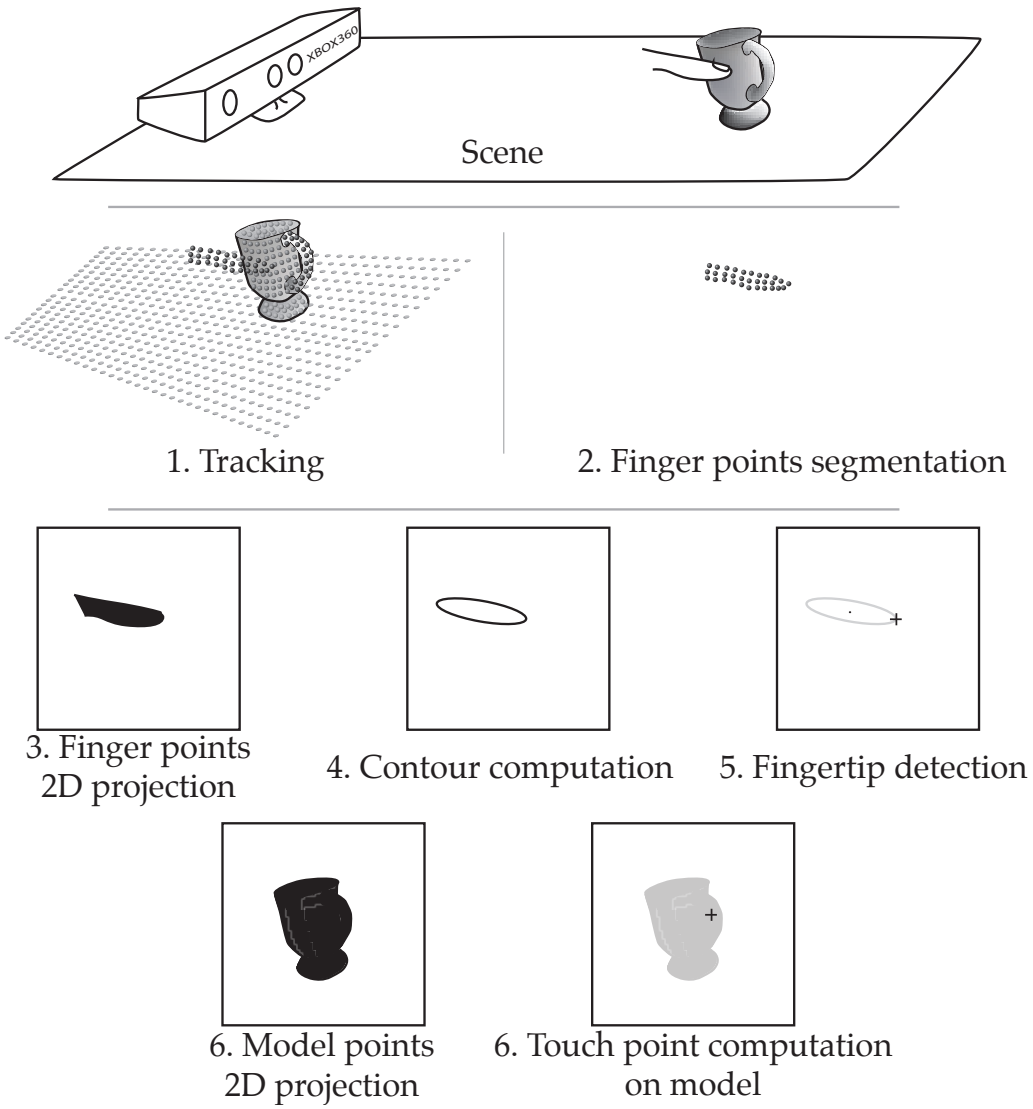

Figure 3.5: *Steps of surface touch detection*

3.3.1 Technical Requirements

Technical requirements are listed.

Necessary and optional hardware components are listed, along with external used frameworks, without including their own dependencies.

Hardware Requirements

- **Microsoft Kinect**: necessary to obtain depth images.
- **CUDA enabled NVidia graphics cards (optional)**: the initial pose detection can be fastened through the use of parallelized code written for the parallel computing platform CUDA. [1]

Hardware requirements: Kinect and optionally, a CUDA enabled graphics card.

External Used Frameworks

- **OpenNI**: used to obtain data from the Kinect. [2]
- **Nestk**: used to pre-process the obtained data from the Kinect and construct a point cloud of the scene. [3]
- **VTK**: used for visualization of point clouds and meshes, as well as to perform operations with sets of points ,such as applying transformations to them. [4]
- **lib-icp**: library used to perform ICP on two point clouds [Geiger et al., 2012]. [5]
- **openCV**: used for simple computer vision algorithms while finding the fingertip of the user's hand [6]

External used frameworks: OpenNI, Nestk, VTK, lib-icp and OpenCV.

Figure 3.6 shows the different components of the system, and where the aforementioned external frameworks are used.

3.3.2 Offline Phase

In order to obtain the 3D model of a real world object, a FabLab 3D laser scanner—Fabscan[7]—was used. The data

[1] http://www.nvidia.com/object/cuda_home_new.html
[2] http://www.openni.org/
[3] http://github.com/nburrus/nestk
[4] http://www.vtk.org/
[5] http://www.cvlibs.net/software/libicp.html
[6] http://opencv.org/
[7] http://hci.rwth-aachen.de/fabscan

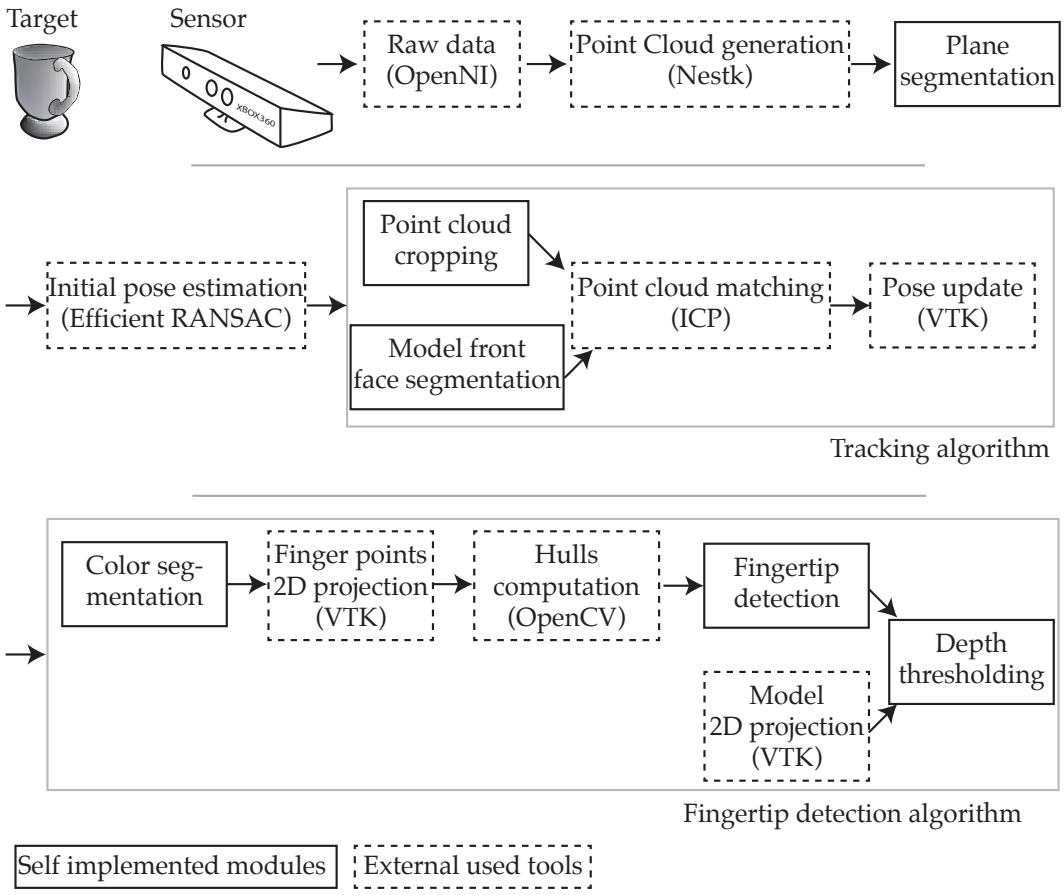

Figure 3.6: *System components. Self implemented modules are enclosed in continuous lines, whereas external used tools are enclosed in dashed lines.*

A *Fabscan* 3D laser scanner is used to obtain the 3D model of the object.

is obtained in the STereoLithography (STL) file format, including the set of points, triangular faces and point normal vectors of the scanned model. This information is transformed into the Polygon File Format (PLY), since it is the most convenient file format to work with and can hold all the necessary information for the system. It is important to highlight that any 3D scanner capable of providing only a set of points can be used, due to the fact that later both faces and point normal vectors can be computed by software.

The model should be placed inside the folder *models* with the name *model.ply*. The user might note the console output when the system is initialized in order to check if the provided file contains errors.

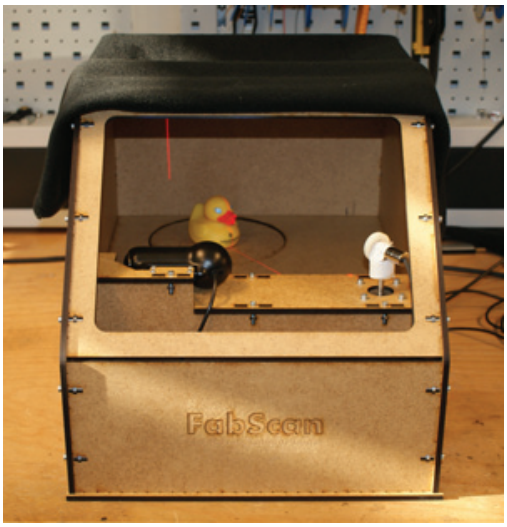

Figure 3.7: *Fabscan 3D scanner.*

Lastly, the obtained colors of the modeled are averaged and stored for hand point segmentation.

Solid color of the model is stored.

3.3.3 Online Phase

Once the model for tracking and the object solid color for finger surface touch detection are available, the online phase can be carried out.

Prerequisite: the model of the object.

Data Acquisition

Data acquisition is made through the use of a Kinect depth camera. Nevertheless, any depth camera can be used, as long as OpenNI can read the data from it. Also, it must provide images with the same level of resolution as the ones obtained by the Kinect, as mentioned in Section 2.3, or higher.

A Kinect depth camera us used to sense the medium.

Data Pre-processing

The 3D point cloud of the scene is computed.

Once the data from the Kinect is available, the next step involves transforming it into a 3D point cloud. The Nestk framework provides a function to do this directly from the Kinect data, thus this functionality was not self-implemented.

The 3D point cloud is transformed such that it matches the 3D space defined by the system.

The 3D space defined in the point cloud provided by Nestk has as origin the lens of the Kinect camera, and it uses as distance units meters. Its axes are the same as the ones defined by the Kinect, and the presented system—as detailed in Section 3.1. Due to the fact that distances obtained in the provided point cloud are expressed in meters, a scaling factor of 1000 is applied; as a result distances are expressed in millimeter.

Plane Segmentation

The ground plane is computed from three user-specified points.

In this step, the system firstly exposes a captured frame from the Kinect to the user, displayed as a 3D point cloud of the scene. By clicking on three on-screen points, the user can define the ground plane. Points considered to be on top of the ground plane are shown highlighted from the rest of the points, and later serve as input for the system. An example of the result is shown in Figure 3.8. The user can iteratively select three points until the plane is placed where she desires.

Initial Pose Estimation

A provided implementation of Efficient RANSAC was used.

The source code for performing Efficient RANSAC pose estimation as described in Papazov and Burschka [2011] was provided by the authors, integrated into the system, and compiled along with it. A GPU accelerated version of this code was also provided, and it is also made available in the system. To use this implementation, the parameter *Use CUDA* must be set accordingly.

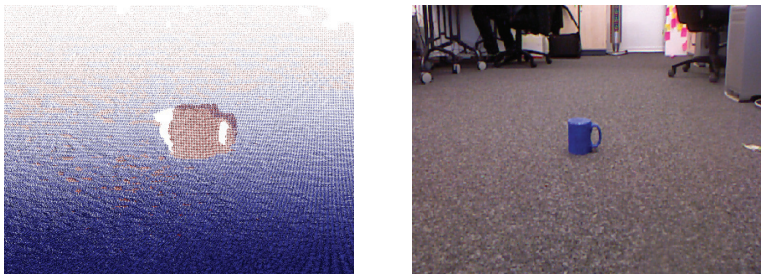

Figure 3.8: *Plane segmentation and removal of ground points. Points on top of the selected plane are colored in red, whereas points on the plane and below are colored in blue (left). The original image (right) contained a mug on top of the floor.*

USE CUDA:
Instructs the system whether the initial pose estimation should be performed using a GPU parallelized implementation or not

Parameter:
Use CUDA

The result of running the code for pose estimation is a rigid transformation that translates and rotates the scanned model from its own defined space into its position and orientation in the real world scene (Figure 3.9). The user may choose to perform initial pose estimation more than once until the estimated pose of the object matches the real one. It is not rare that the first pose estimation does not match the correct one, a fact that the user might realize and desire to correct. This initial position is the starting point for later running ICP.

The model is registered, according to the computed transformation.

Figure 3.9: *Initial pose estimation. The point cloud obtained from the Kinect (left) is used to determine the pose of the loaded model (middle). The result is an estimation of the model on the scene detected by the Kinect (right).*

A temporary space transformation is used to feed the provided Efficient RANSAC implementation.

Due to the fact that the provided source code for initial pose estimation does not accept as input negative Z values, a rotation of 180 degrees on the Y axis is applied before feeding the point cloud of the scene into the algorithm. After the initial pose estimation has been computed, the inverse rotation is applied to the yielded transformation.

Object Tracking

Repeated for each frame.

This is the last step for performing tracking, and it is repeated throughout all obtained frames from the Kinect.

The scene is cropped to include only the point inside the bounding box of the registered model.

Once the scanned model is fitted in the scene, its bounding box is used to segment the latter and obtain only the points in it which belong to the tracked object. The boundig box of the object can be easily obtained from the loaded VTK model, and in a simple loop all points outside the box are discarded. Here the parameter *Cropping tolerance value on the scene point cloud around the model* is used to increase the size of the bounding box and avoid cropping of parts of the object when it moves.

The front facing side of the model is obtained.

The next step involves obtaining the front facing side of the model, for which the position of the camera is tested against each point of the model as described below.

Let P be a point such that $P = [x_P, y_P, z_P]$ and N be its normal vector such that $N = [a, b, c]^T$. Both define a plane that satisfies Equation 3.3, where X is a generic point in space.

Method for computing the distance from a point to a plane, hence, its position in space with respect to such plane.

$$n * (X - P) = 0 \tag{3.3}$$

As mentioned in Section 3.2.2, it is simple to test whether a point in space belongs to the plane or not by examining its distance to the plane—if it is zero, the point lies on the plane, if it is positive it lies on one of the subspaces the plane defines, and if it is negative it lies in the other defined subspace. Given Equation 3.3, when N is a normalized vector, the result of substituting X by a point is its shortest distance to the plane. This method is applied to each point of the model and its corresponding normal vector to test

whether the camera, located at the origin $O = (0, 0, 0)$, is on the front of the model or not.

The cropped point cloud of the scene obtained from the Kinect, containing the points of the model in the scene, is used as the model input parameters of the ICP algorithm; the point cloud containing the front-facing side of the model is used as the template input parameters of the ICP algorithm. The reader may refer to Section 2.5.1 for details of the ICP algorithm.

The points of the object in the scene are used as the *model* for running ICP. The front-facing side of the registered model is the *template*.

Once ICP finishes running, the yielded rigid transformation is applied to the overlaid scanned model on the scene. To apply the transformation, the VTK system provides primitives that take as input both a rigid transformation and a mesh, and yield the mesh after applying the transformation. Nevertheless, given the linear transformation expressed as a rotation and translation, a transformation matrix expressed in homogeneous coordinates can be constructed— transforming the model can be done by simply multiplying each point by the transformation matrix.

The yielded transformation is applied to the model of the object, and it is overlaid on the scene point cloud in the correct location.

Surface Touch Detection

Using the solid color of the model, the points that belong to the hand can be segmented from the points that belong to the model, by using simple color segmentation. Given the Hue, Saturation and Brightness (H, S and V respectively) values of the model solid color, colors with similar H, S and V values are accepted as points belonging to the model. All other points are considered to be of the hand. Color filtering is applied considering a tolerance value defined in the parameter *HSV tolerance threshold*.

Points belonging to the finger are segmented using color filtering.

A problem that arises at this stage is the object boundary ambiguity as mentioned in Section 2.3. The depth of foreground and background pixels near boundaries are confused by the Kinect, and it reports parts of the background objects as being on the foreground objects. In simple words, it snaps part of the background into the foreground.

Due of the object boundary ambiguity problem, parts of the background are sticked to the side of objects.

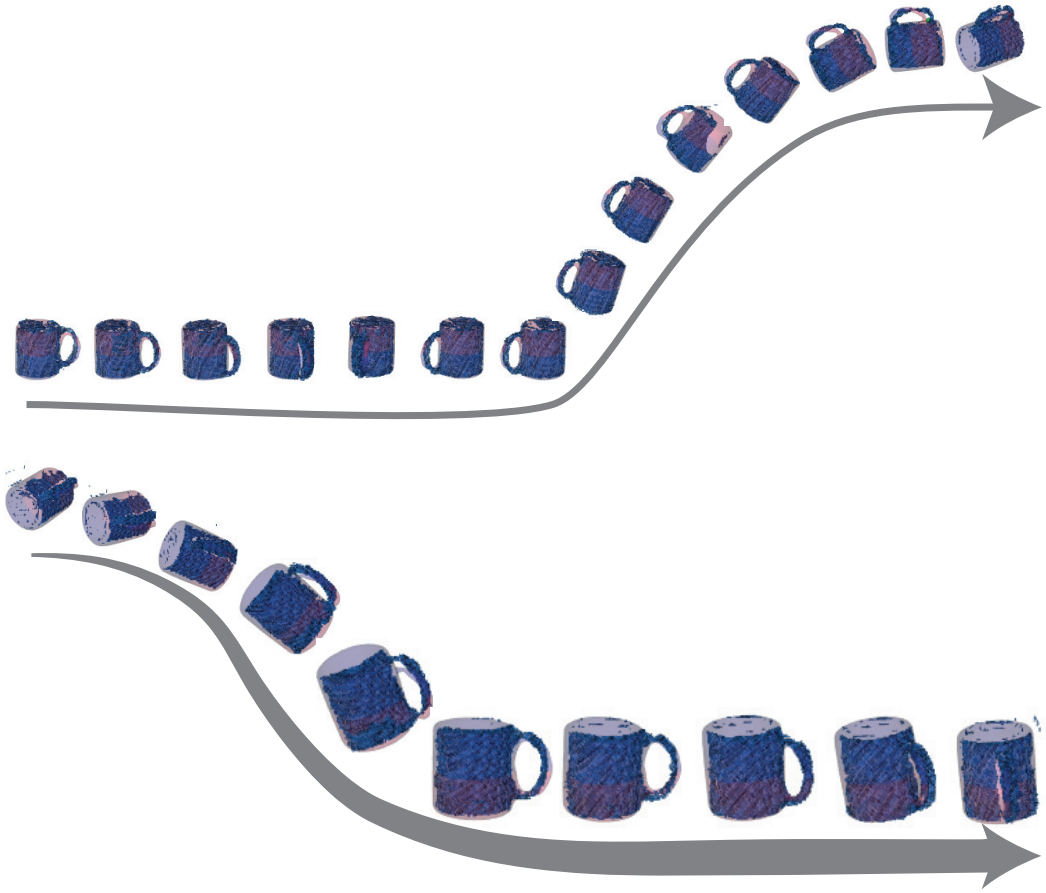

Figure 3.10: *Sequential frames of a tracked object. The point cloud of the scene corresponding to the mug is overlaid with the estimation of the position of the object in 3D. The object was turned, lifted, and finally brought closer to the camera while being turned and lowered and the same time.*

To solve this problem, points at a certain distance from the camera are segmented and considered to be part of the finger.

In order to solve this problem, once the points from the user's hand are segmented, another segmentation step is conducted. By taking the closest point to the depth camera from the point cloud composed by the user's hand points, only points within a certain distance from the camera are considered for the surface touches detection. This distance can be set in the parameter *Finger width*. Since the points corresponding to boundary ambiguity are snapped to the sides of the object, they will be cropped out using this approach.

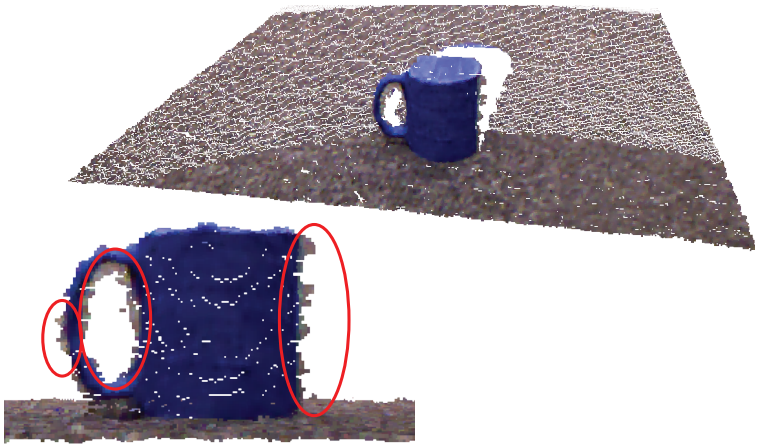

Figure 3.11: *Object boundary ambiguity problem. A scene point cloud as seen from the Kinect (top); it can be observed how part of the floor sticks to the side of the object (bottom).*

FINGER WIDTH:
Distance used to crop points in the hand point cloud, in order to segment only a portion as wide as a finger from the hand point cloud. Starting from the closest point to the camera, points further away from this point are cropped out (more distant in the Z axis)

Parameter: *Finger width*

The perspective projection (Appendix A) of the hand cropped points is computed, and using simple computer vision algorithms provided by the OpenCV library, the fingertip can be easily found. The final image of the projection is 640 pixels high and 480 pixels wide.

The perspective projection of the finger points is computed.

Firstly a binary image is computed using the function `threshold` provided in the OpenCV library. Following, the convex hulls of the projected image are found through the use of the function `convexHull`. The center of the largest convex hull, other than the image itself, is computed as the weighted sum of all the points inside it. The furthest point in the hull, the one that is at the same time most further from all edges is reported as the fingertip.

Using simple computer vision algorithms, the finger contour and the fingertip are determined.

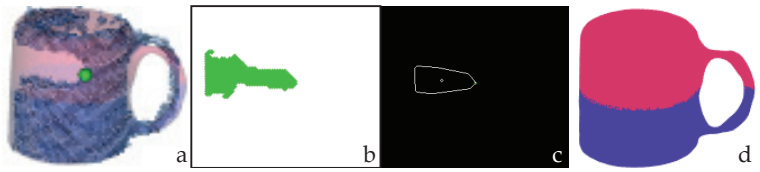

Figure 3.12: *Surface touch detection. In the point cloud of detected blue points (a), it can be clearly seen how there is a segment missing, which corresponds to the overlaid finger. The perspective projection of the points corresponding to the finger (b) is obtained, and the largest convex hull is computed (c). The fingertip is found (green dots on images a and c) and its depth value is compared to the depth value of the same point in the perspective projection of the model (d).*

Depth values in the 2D projections of the finger points and the registered model are used to distinguish finger hovering from actual touch.

Lastly, the perspective projection of the model placed on the scene is also computed. The depth of both the point corresponding to the fingertip found on the perspective projection of the finger and the same point on the perspective projection of the placed model on the scene are compared. To accept the overlapping as a touch, the value of the parameter *Fingertip touch minimum distance threshold* is taken into account.

3.3.4 System Parameters

System parameters.

This section provides an enumeration of the already mentioned input parameters required by the system, including their default values.

- **Cropping tolerance value on the scene point cloud around the model**: to reduce the amount of processed points, the ones that correspond to the tracked object in the point cloud of the scene are cropped. For this, the model is placed in the scene as determined by the previous tracking step, or the initial pose estimation. A tolerance value for cropping around the fitted model is used in order not to discard points belonging to the edges of the object when it is moved. Default value: 20 millimeters.

- **Finger width**: once the hand point cloud is made available, only the closest points to the camera are considered. This eliminates the *object boundary ambiguity* problem produced by the Kinect. It is intended to crop the finger such that, its shape can be projected, and only the closest points to the camera are needed. The finger width parameter is used to segment from the closest point of the Kinect camera, up to the value specified by the parameter. Default value: 15 millimeters

- **Minimum hull size**: while detecting fingers in the perspective projection of the user's hand, the convex hulls of the image are identified. A minimum hull size value, expressed in pixels, is used to discard small hulls that correspond to noise. Default value: 1000 pixels.

- **Fingertip touch minimum distance threshold**: after the fingertip is identified, it is necessary to differentiate simple overlaps between the fingertip and the object, from actual touches. This parameter sets the minimum distance used to differentiate a finger hover from a finger touch. Default value: 0.11 millimeters

- **HSV tolerance threshold**: in order to segment the hand points from the object points, a color filter is used. Hue, Saturation and Brightness values from the detected points on the scene are compared to the solid color of the object. A tolerance value is used to reduce effects caused by changing light conditions. H, S and B values are in the range $[0, 1]$. Default values: H = 0.1; S = 0.2; B = 0.

The system has been exposed from both a conceptual and a technical point of view. It is intended that researchers adopt it for studying user interaction with everyday objects. For this, accuracy of its capabilities and its limitations must be known. These are explored in the following chapters.

Accuracy is studied in the next chapter.

Chapter 4

Object Tracking Evaluation

The presented system is capable of performing two distinct but related functions—object tracking in 3D space and touch sensing on the surface of the tracked object. Each capability is evaluated separately; in this section the evaluation corresponding to object tracking is presented. It is important to highlight that tracking speed is not evaluated in the study, due to the fact that tracking speed varies according to the used hardware and thus it is not a metric intended to be reported. Nevertheless, tracking speeds obtained while using a specific hardware setup are presented as a way of showing a rough value of this metric.

Tracking and touch detection are evaluated separately.

4.1 Study Design

In order to evaluate the tracker, the data obtained by it while tracking an object needs to be compared with the real world data corresponding to the movement of the object. The need for ground truth data obtained (a) independently from and (b) without interrupting the normal function of the presented tracking system arises. A VICON motion tracking system is used to provide accurate data on a tracked object position and orientation as ground truth data.

Data from the system needs to be compared with real object data. Data must be obtained independently from and without altering the system behavior.

Does the system provide functionality to study user interaction without intruding users?

It is intended to evaluate if the system provides the necessary functionality to study user interaction with everyday objects, without intruding or interrupting such interaction. For this purpose, tracking must be performed effectively on objects and at the same time do not oblige the user to use the tracked object in a specific way.

Evaluation metrics: absolute differences between both tracking systems, maximum position error, average position accuracy and orientation error.

Based on reports from evaluation of tracking systems such as [Ehara et al., 1995] and [Windolf et al., 2008], the system position tracking performance is reported in terms of:

1. Position error of the tracked object, expressed as the mean of the absolute value of the differences between distances obtained by both tracking systems.
2. Maximum position error.
3. Average position accuracy.
4. Orientation error of the tracked object.

It is also intended to investigate:

Also investigated: effect of distance over accuracy and angular speed, effect of visibility of symmetric features.

1. The effect of the distance from the depth camera to the tracked object over error of the tracking system when tracking both position and orientation. As mentioned in Section 2.3, there is precision loss with increasing distance when using depth cameras, thus it is suspected that precision will be affected by this factor.
2. The effect of angular speed when tracking both position and orientation, as perceived by the depth camera, over error of the tracking system.
3. The effect produced by having visibility over symmetric features over orientation error when tracking.

The evaluation was not conducted with users.

Before diving into the evaluation, it is important to highlight that the performed study is focused on evaluating the accuracy of the system tracking capabilities, and not in user interaction with it—there were no users involved in the tracking performance study. Also, the system parameters were set to their default value during the evaluation.

4.1.1 Evaluation Method

It is essential to define how the error of the system for tracking position and orientation is determined. In this section both error metrics are defined.

Position error of the system is defined as the difference between the data obtained from the presented system and the VICON motion tracker. Each axis is evaluated independently. Average accuracy is reported as the deviation of the measured values from the ground truth, determined as the *root mean square* of all errors—the magnitude of the varying error. The root mean square is used due to the presence of both positive and negative differences obtained between both systems. In Equation 4.1, accuracy of the X axis computation is shown. x_i corresponds to the value obtained by the presented tracker whereas x_i^0 corresponds to the value obtained by the VICON tracking system—the ground truth provider.

Position error is computed for each axis independently as the absolute difference from ground truth data.

Accuracy is computed as the RMS of the differences form ground truth data.

$$a_x = \sqrt{\frac{1}{m-1}\sum_{i=1}^{m}(x_i - x_i^0)^2} \tag{4.1}$$

In *Metrics for 3D Rotations: Comparison and Analysis*, Huynh [2009] compares six different metrics for measuring distances between 3D rotations. All six presented metrics are equivalent between each other, thus the same results would be reached by using any of them. The *Norm of the Difference of Quaternions* metric was chosen for this evaluation, and is represented with the Greek letter θ. Given two rotations expressed as quaternions, q_1 and q_2, the norm of the difference of quaternions is computed as in Equation 4.2.

Rotation error is computed as the norm of the difference of quaternions between both obtained rotation angles.

$$\theta(q_1, q_2) = min\{||q_1 - q_2||, ||q_1 + q_2||\} \tag{4.2}$$

By using this metric, rotation in all three axes is reported as one unique value. The metric θ has the range: $[0, \sqrt{2}]$.

Rotation differences are evaluated for all axes in one metric.

4.2 Setup

Objects are moved repeatedly following the same path on top of a track.

In order to collect enough data for comparison between both the presented system and the VICON tracking system, measurements must be taken while the tracked object performs the same movement repeatedly. For this purpose, a track that allows the tracked object to perform the same movement repeatedly was constructed.

A VICON motion tracker and the presented system were calibrated around the track.

Five VICON Bonita cameras were mounted as part of the tracking infrastructure and a standard calibration process was conducted. A Lego toy train track was used with the purpose of moving objects in a fixed path at constant speed. The tracks were set as in Figure 4.1 in an oval position with a small outer bump. The closest point from the track to the Kinect camera was of 60 centimeters, while the furthest 122 centimeters. The sides of the tracked spanned a total of 728 centimeters, distributed as 343 centimeters towards the negative direction of the X axis and 385 centimeters towards the positive direction of such axis—with respect to the tracking system reference axes. A diagram of the setup can be observed in Figure 4.2.

Figure 4.1: *Picture of the setup used for evaluating tracking error. Five VICON Bonita cameras are laid around the tracks, shaped as an oval with a small bump. The Kinect depth camera is placed in front of one of the sharp curves of the tracks.*

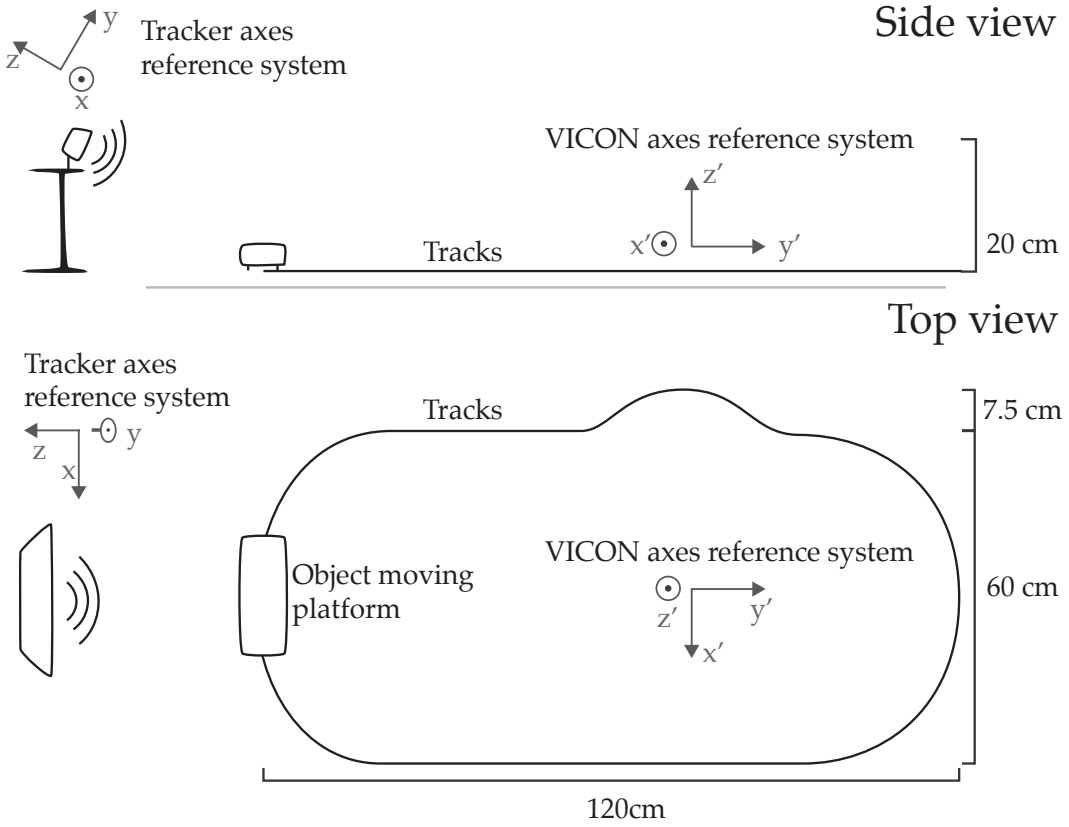

Figure 4.2: *Diagram of the setup used for evaluating tracking error. A toy Lego train track was laid on the ground and assembled in the shape of an oval with a small bump. The Kinect depth camera was placed in one of the sharp curves of the track, at about 60 centimeters of the track.*

Due to the fact that two independent systems are used, data is expressed in two different 3D spaces—each one corresponding to each system. Thus, the *absolute orientation* problem must be solved in order to compare both reported results in a unique reference system.

Both obtained datasets must be expressed in the same space.

The absolute orientation problem, also known as rigid alignment problem, refers to the problem of determining the transformation between two different coordinate systems, as mentioned by Horn [1987]. In practice, one of the two coordinate systems is selected as the absolute one, and a transformation that expresses the points of the remaining set in the coordinate system of the absolute one is computed.

A rigid transformation to transform one space into the other is computed.

A set of corresponding points in each system is obtained.

A rigid transformation between the VICON tracker and the presented system coordinate systems can be determined through the use of a set of three or more corresponding points in each coordinate system. Naturally, the coordinates of the points in each set are expressed in their own coordinate system.

A closed-form solution of absolute orientation is used.

A closed-form solution of absolute orientation is introduced by Horn [1987]. Since this is a side problem and does not concern the system itself but its evaluation, the full numeric method is outside the scope of this work. The reader might refer to the source for a complete and detailed explanation. The practical implementation used[1] of the closed-form alignment can be found in Appendix B.

Forty corresponding points are used for computing the transformation between the space of the VICON into the one of the presented system.

For the evaluation, once both systems were placed and calibrated individually, a set of forty points were obtained in each coordinate system. Due to the introduction of noise generated by the Kinect, a larger set than one with only three points was needed. Using only three points lead to large errors when transforming points that were distant from the used calibration points. When selecting the calibration points, distances in all three axes among them were varied in order to cover all the tracked spaced. The calibration points can be found in Table 4.1 and the used resulting transformation matrix T can be found in Equation 4.3.

$$M = \begin{bmatrix} 0.99944 & 0.02587 & 0.02136 & 31.09353 \\ 0.01581 & 0.19823 & -0.98003 & -828.6708 \\ -0.02959 & 0.97981 & 0.19771 & 419.31825 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.3)$$

Two objects with different geometric complexity were used in the evaluation.

Two objects were used for the tracking evaluation—an everyday coffee mug and a plastic toy duck. The first object has less complexity in its 3D model whereas the second presents a higher amount of curves and edges. Although the effect of tracking models with a higher degree of geometrical complexity is not studied, objects with different geometrical shapes were used in order to reduce the chance that this factor influences the obtained data as a confounding variable. As seen in Figure 4.3, both objects were

[1] http://isiswiki.georgetown.edu/zivy/educationalMaterial.html

Kinect			VICON		
X	Y	Z	X	Y	Z
-400.7960	29.0900	-1722	-398.8401	836.1707	119.7003
-394.1850	-88.1818	-1741	-400.3378	833.1764	1.6789
-544.4210	-88.1818	-1741	-545.2522	826.6237	5.4548
-550.4580	18.2020	-1706	-541.7155	828.9904	120.3427
559.0920	27.1108	-1665	553.5726	784.3387	101.6965
553.7200	-95.8956	-1648	551.1520	780.3543	-16.2804
419.6360	-96.3609	-1656	406.1254	781.9776	-11.5884
410.1680	24.9874	-1665	410.5264	785.1924	103.2491
-391.3900	-62.0840	-1206	-388.7451	315.3060	134.0378
-388.7490	-170.5070	-1219	-389.7903	309.8441	16.0999
-539.7390	-182.1900	-1215	-534.8512	307.6694	19.2582
-539.1140	-67.0139	-1173	-531.7722	312.3771	134.0872
535.6980	-31.3225	-1347	540.9319	494.7785	110.5428
542.0610	-136.5150	-1363	539.6690	491.0000	-7.4589
400.9700	-149.0410	-1379	394.7416	497.6149	-4.3487
387.5070	-27.2290	-1368	397.9916	500.5592	110.5343
-231.6550	-101.2010	-908	-231.0111	56.1128	146.5461
-230.0020	-211.6050	-940	-232.9227	51.1553	28.5968
-383.4150	-213.3680	-940	-377.7604	43.4312	33.0405
-376.8930	-103.5610	-904	-373.8162	47.7268	147.8592
384.2840	-101.4560	-850	393.6283	-16.3200	135.1173
384.2820	-233.0480	-843	391.8315	-21.6597	17.1828
246.7150	-226.8880	-854	246.8779	-26.9526	21.4029
244.1580	-115.6360	-829	250.7012	-22.3195	136.2126
59.4390	113.2250	-1947	50.7455	1118.0900	137.4312
66.4866	2.6937	-1969	47.9139	1116.0166	19.4144
-91.2601	-7.6050	-2026	-96.6754	1104.9828	24.8481
-81.2614	100.7300	-1969	-91.8247	1106.5172	139.7021
100.6170	-17.0073	-1219	113.0466	362.3277	155.1200
105.5360	-136.3610	-1223	107.8763	360.4980	37.1785
-35.7082	-137.6560	-1223	-37.0076	360.8261	45.2966
-30.5681	-20.2938	-1202	-29.9255	361.9608	160.0393
69.2067	-102.8100	-802	78.5142	-47.8999	165.5193
66.5486	-208.3840	-825	73.5085	-49.9554	47.5745
-66.8712	-212.5000	-829	-71.3327	-53.7812	55.5602
-72.7240	-97.3157	-786	-64.3930	-52.3677	170.3085
97.8297	-102.4200	-745	113.8408	-101.2382	168.4346
94.0966	-212.4300	-759	102.7189	-118.4785	52.1619
-45.3487	-206.4030	-711	-30.2050	-172.1767	74.6358
-35.7682	-99.7994	-700	-17.2470	-155.2184	187.6042

Table 4.1: *Calibration points used to solve the absolute orientation problem. The left set of points corresponds to the presented system coordinate system whereas the right set to the VICON tracker coordinate system.*

Figure 4.3: *Used Models during the tracking evaluation. Left column: different views of the used coffee mug; right column: different views of the used plastic duck. Rows from top to bottom: perspective view, side view and top view*

painted in dark blue, although this is necessary only for finger touch detection and does not affect the tracking process. VICON tracking markers were added in order to allow tracking by the VICON motion tracking system, placed such that tracking by the presented system is not considerably affected.

4.3 Procedure

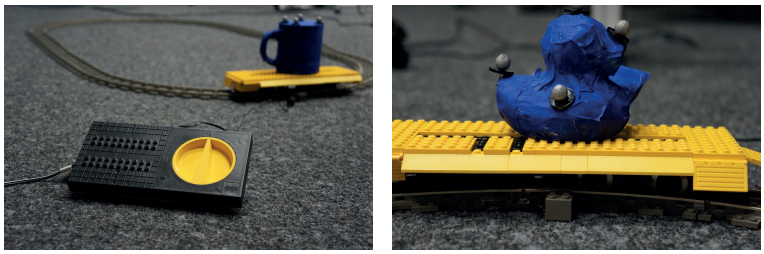

Figure 4.4: *Two objects placed on top of the Lego toy train ready to be tracked. On the left: the coffee mug on the background and the toy Lego train speed controller on the foreground; on the right: the plastic duck.*

For each object the same procedure was carried on.

1. The object was placed on the Lego toy train at a defined starting position, as shown in Figure 4.4. All six starting positions of the two objects are shown in Figure 4.5, relative to the point of view of the Kinect depth camera.

2. The train was set off to a defined and constant level of speed, level 1 on the Lego train speed controller, towards a defined side. Three complete runs were recorded.

3. The direction of traveling was changed and three more complete runs were recorded.

4. The object was rotated by 60 degrees clockwise and the whole procedure was repeated for the new starting point.

Procedure: the object is placed on the tracks, three laps in each direction are recorded, the object initial position is changed and the procedure is repeated.

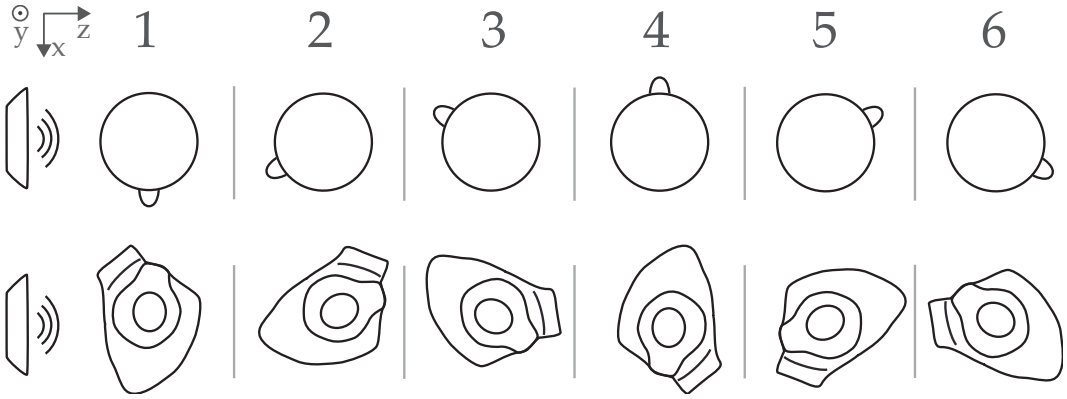

Figure 4.5: *The six starting positions of the two tracked objects used in the tracking evaluation, with respect to the Kinect depth camera. Top row: initial positions of the coffee mug. Bottom row: initial positions of the plastic duck. The difference between each starting position is of sixty degrees. On the top left the axes corresponding to the coordinate system of the presented tracking system are shown.*

Twelve measurements were recorded.

In total twelve measurements were recorded: six different angles combined with two running directions. In the case of the coffee mug 4535 frames were recorded, whereas for the plastic duck 3894. The significant difference is due to the weight of the objects; the coffee mug being heavier than the plastic duck slows down the speed of the train, thus allowing more frames to be recorded.

4.4 Results and Analysis

Position and orientation error are analyzed separately.

In this section, collected data is (a) summarized, (b) presented, and (c) analyzed. Firstly, the obtained data set from the presented system was transformed, such that its coordinate system matched the one of the VICON tracking system, as mentioned in Section 4.2. Tracking error for position and orientation are presented separately.

4.4.1 Position Error

The absolute value of the differences of both system measurements is computed.

The differences between the measured distances of each tracking system were expressed in absolute value, such that

the deviation of the tracking system from the ground truth data would be expressed regardless of its direction.

The average error found in tracking for each axis is shown in Table 4.2. As it can be observed, the largest error comes from the Y axis of the VICON tracking system, which corresponds to the depth in the presented tracking system.

Largest error corresponds to depth axis.

	VICON axes	Tracking error (mm) (avg. ± SD)	Maximum error (mm)	Average accuracy (mm) (avg. ± SD)
Coffee Mug	X	(16.27 ± 12.51)	67.71	(20.53 ± 31.28)
	Y	(24.04 ± 15.65)	74.50	(28.69 ± 47.23)
	Z	(4.32 ± 3.06)	21.51	(5.30 ± 4.29)
Plastic Duck	X	(15.22 ± 13.57)	94.60	(20.57 ± 29.88)
	Y	(28.73 ± 17.70)	125.61	(33.75 ± 56.27)
	Z	(4.11 ± 3.80)	35.02	(5.60 ± 7.69)

Table 4.2: *Tracking error in each axis, expressed on the ground truth data provider coordinate system.*

Effect of Distance and Angular Speed on the Position of the Tracked Object

The effect of the distance from the Kinect depth camera to the tracked object over tracking error can be observed in Figures 4.6 and 4.7 for the coffee mug and for the plastic duck respectively. The tendency of the error is to stay constant, despite of the increasing distance. There is a small variation for instance on the X axes of both tracked objects, although it is not significant.

Position error stays mainly constant as depth increases.

Taking a closer look at the obtained data, the reader might perceive that there is a larger accumulation of error on certain areas in each axis. In the X and Z axes, there is a larger error in the edges, which corresponds to the curved segments of the used track. In the Y axis there is a larger error in the center, which corresponds to the straight segments of the used track. This effect is not due to an error of the system itself but due to misalignments when recording measurement for the evaluation.

Error values change systematically due to misalignments when measuring.

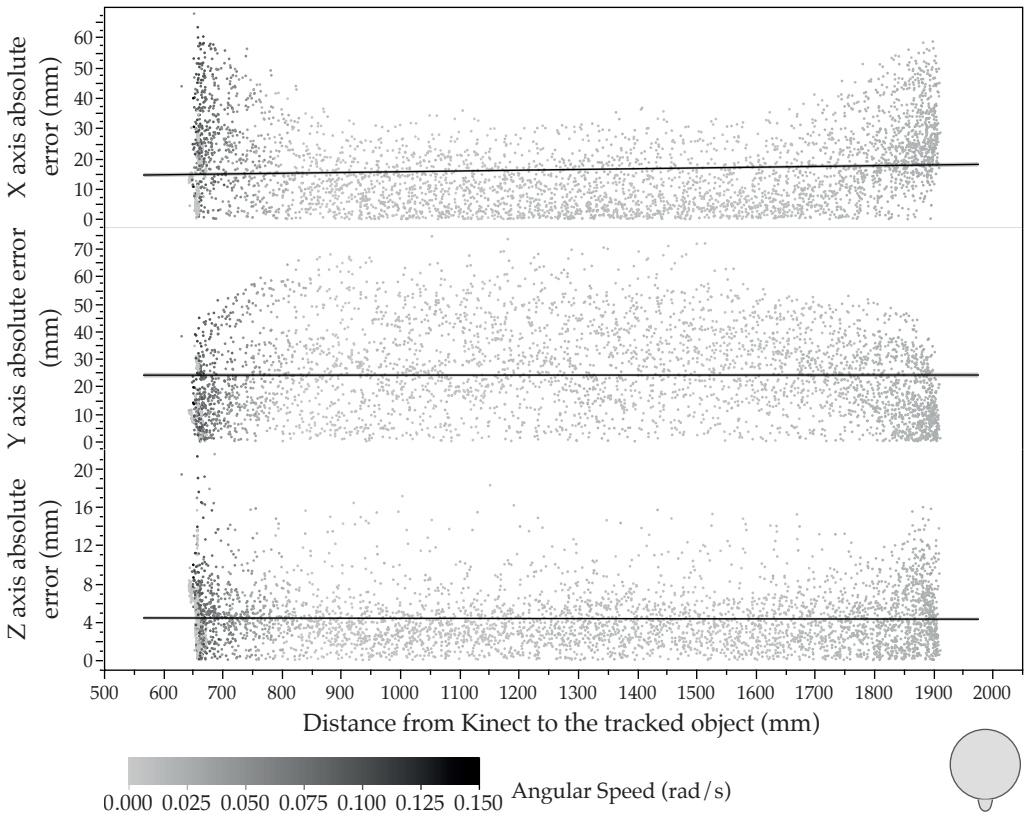

Figure 4.6: *Position error when tracking the coffee mug. Data is expressed in the ground truth provider space. Error is expressed as the absolute value of the difference between the distances measured by both tracking systems, and it is plotted as the distance from the Kinect depth camera to the tracked object changes. Data is segmented by the angular speed of the tracked object with respect to the Kinect camera.*

Position is reported as the center of the object on both systems. The centers do not correspond in the real world.

The recorded position of both objects corresponds to the center of their tracked object model. Due to the fact that both systems consider different points as the center of the model, there is a small deviation between both reported values. See Figure 4.8 for a diagram on this effect. The difference in each axis of both centers changes as the object moves.

Centers differing in their position cause systematic errors in each axis.

As a consequence of both reported centers differing, an error from measurement is accumulated. On the curved edges of the track, there is a greater difference in the Y axis due to the different positions of the centers, whereas in the left and right sides of the track, as seen from the Kinect camera, this effect is smaller. This can be easily detected when

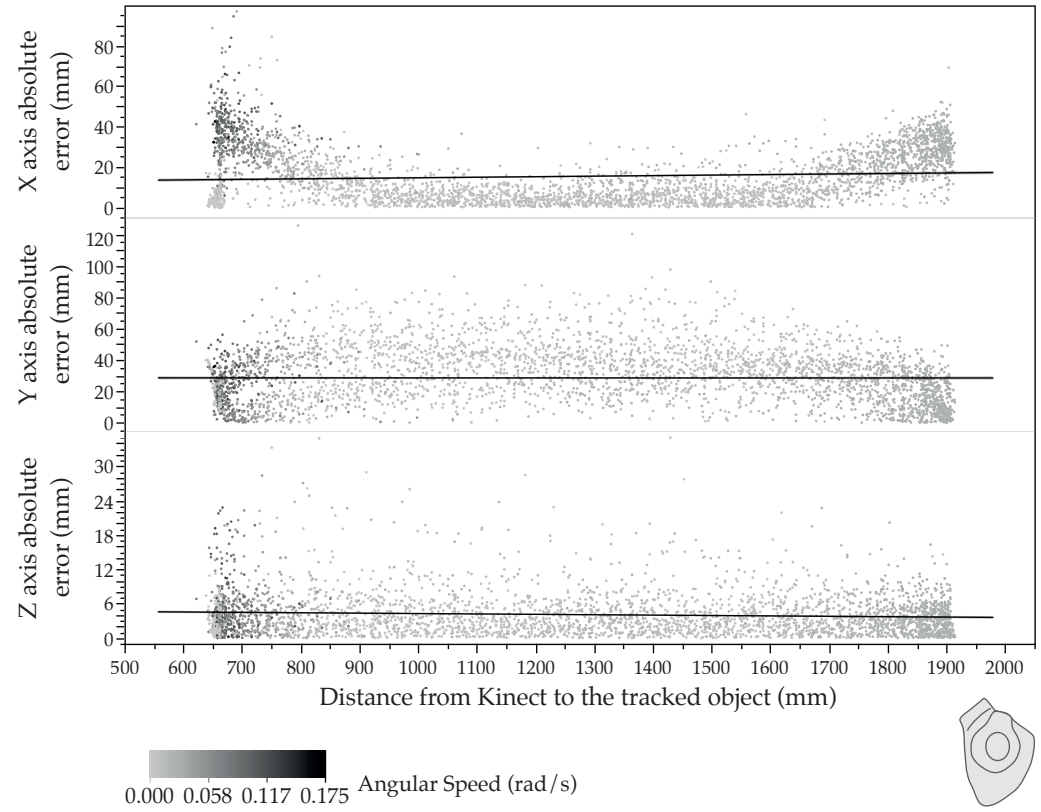

Figure 4.7: *Position error when tracking the plastic duck. Data is expressed in the ground truth provider space. Error is expressed as the absolute value of the difference between the distances measured by both tracking systems, and it is plotted as the distance from the Kinect depth camera to the tracked object changes. Data is segmented by the angular speed of the tracked object with respect to the Kinect camera.*

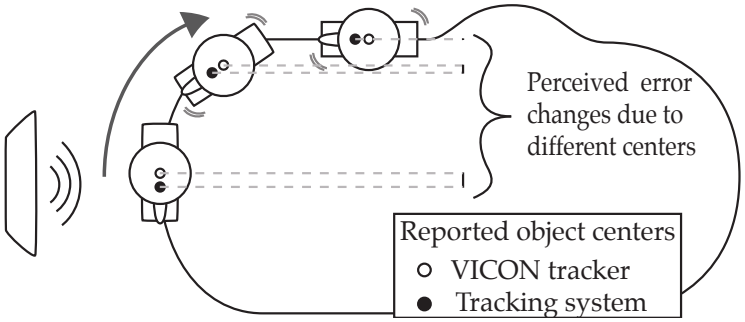

Figure 4.8: *Example depicting the systematic increasing and decreasing differences in the tracked object position, due to the different reported centers, by both the presented tracking system and the ground truth data provider during the tracking study.*

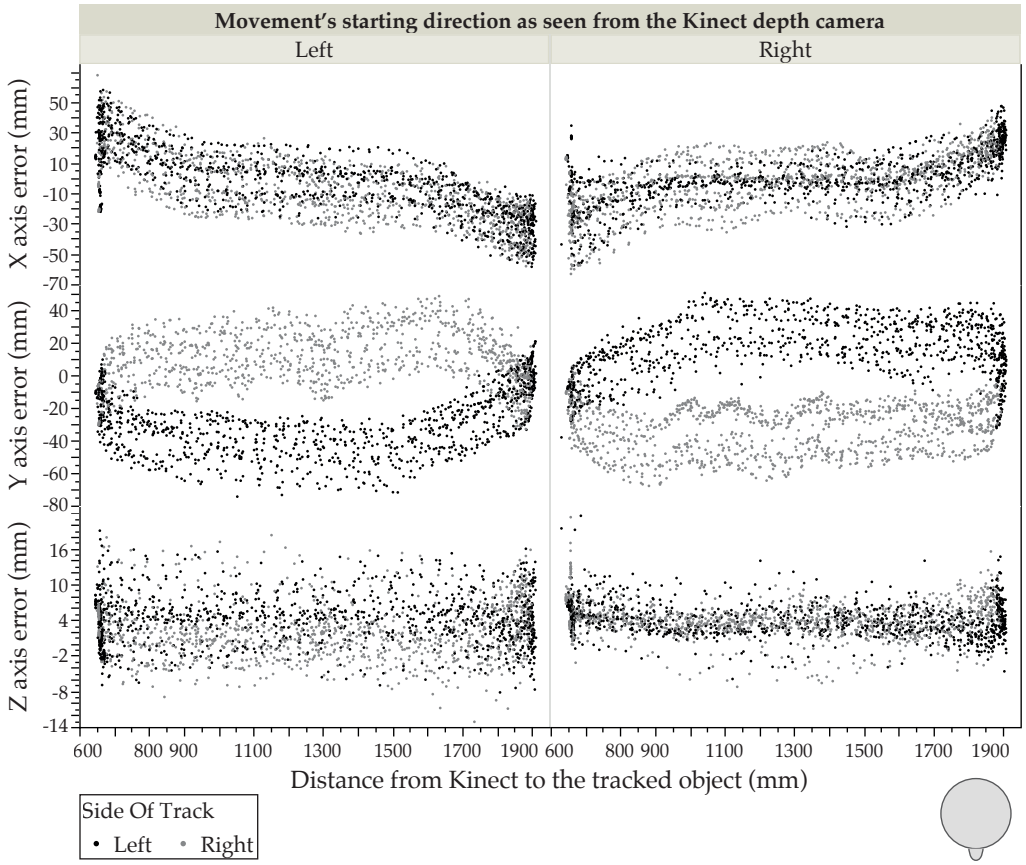

Figure 4.9: *Raw position error when tracking the coffee mug. Data is expressed in the ground truth provider space. Error is expressed as the difference between the distances measured by both tracking systems, and it is plotted as the distance from the Kinect depth camera to the tracked object changes. Data is segmented according two criteria: the starting direction of the object when moving on the track and the side of the track, both views being relative to the Kinect depth camera position.*

taking a look at the raw error points, without considering their absolute values. In Figures 4.9 and 4.10, which depict the differences in each axis before computing their absolute value, the data points corresponding to the Y axis clearly shows how there is a systematic offset on the tracked position throughout all laps when the object starts moving towards one direction.

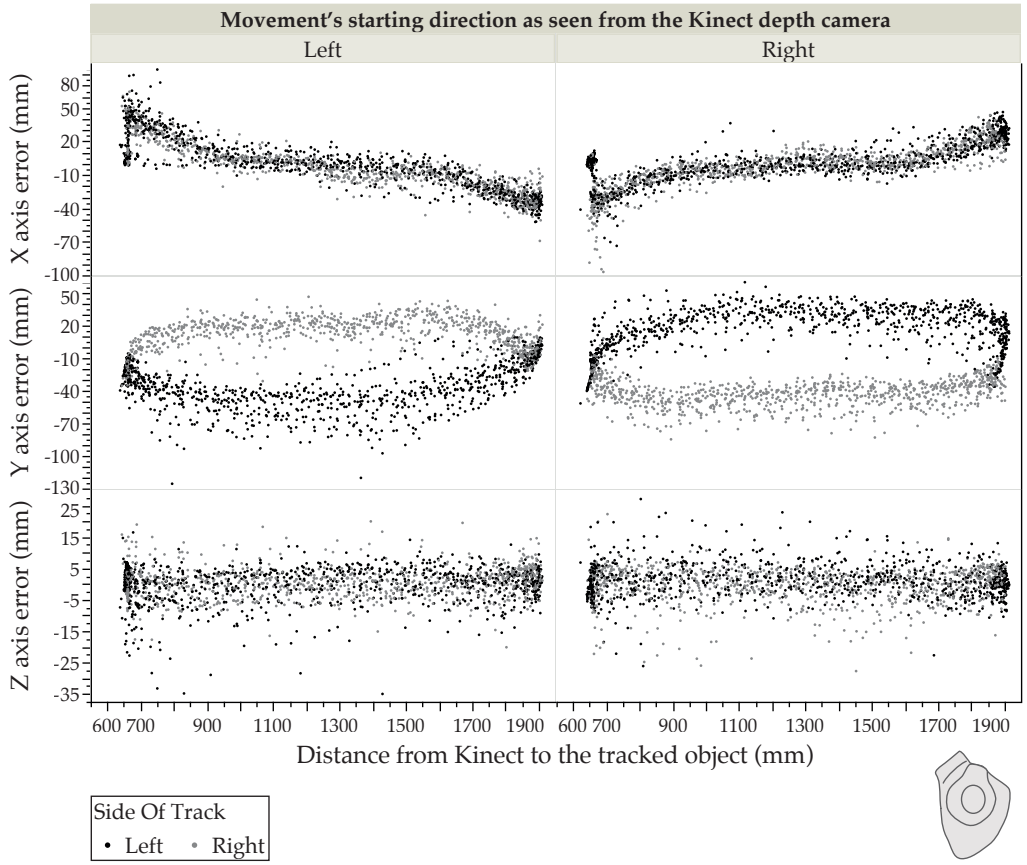

Figure 4.10: *Raw position error when tracking the plastic duck. Data is expressed in the ground truth provider space. Error is expressed as the difference between the distances measured by both tracking systems, and it is plotted as the distance from the Kinect depth camera to the tracked object changes. Data is segmented according two criteria: the starting direction of the object when moving on the track and the side of the track, both views being relative to the Kinect depth camera position.*

Even without considering the introduced error by the differences on the centers while recording positions of the object, there is no trend on error. No significant increase on the error is observed in neither of the used objects during the test, in a range of up to two meters.

Nevertheless, in a range of up to two meter error does not increase.

With respect to angular speed, as expected, higher speed was detected near the Kinect depth camera. As seen on Figures 4.6 and 4.7, angular speed did not significantly affect error.

Angular speed does not increase error as long as the system can follow the tracked object.

4.4.2 Orientation Error

The difference of the rotations expressed as quaternions will be used.

As aforementioned, the norm of the difference of the rotations reported by both tracking systems, expressed as quaternions, were analyzed in order to measure orientation error. Their values are in the range $[0, \sqrt{2}]$.

An initial offset caused by different initial positions of the models in each system is corrected.

After obtaining the measurement data, an initial offset in both tracked objects was identified and corrected. This problem is similar to the one of the centers when evaluation position tracking; in this case both tracking systems differ in the initial orientations of the objects. This error was corrected by shifting the data towards the origin as much as the amount of the offset degrees, and shifting the remaining negative values of the data on the higher half of the range. This is possible due to the periodicity of angle rotations. In the case of the coffee mug the initial offset was of ninety degrees, easily visualized in Figure 4.11 as a line at half the range with a higher accumulation of values. In the case of the plastic duck the offset was of 180 degrees (Figure 4.12).

Looking at runs individually, a trend on error is observed. When a symmetric feature is not in sight, rotation error increases.

At a first glance the rotation differences seem random, although taking a closer look at each individual test, sense comes of the data easily. In Figures 4.13 and 4.14, the individual runs for both tracked objects are shown. Data points are connected by lines, showing the order in which data was collected. The general behavior is that the tracking system introduces error in the orientation when a symmetric feature of the object is not visible from the Kinect camera

Examination of a single run and the system behavior.

By examining an individual run, the general behavior of the system can be identified. Figure 4.15 depicts an enlargement of the second run of the tracking evaluation for the coffee mug. The enumerated and highlated data points indicate a set of sequential frames, which correspond to the enumeration of Figure 4.16. In the first frame, an asymmetric feature of the object is present—the handle of the mug. The object starts rotating clockwise and its handle becomes out of sight from the perspective of the camera, as it is occluded by the object itself—Figure 4.16/1–4. While the tracked object moves and there are no asymmetric visible features, the system incorrectly estimates its orientation as shown in Figure 4.16/5–8. During this time, the han-

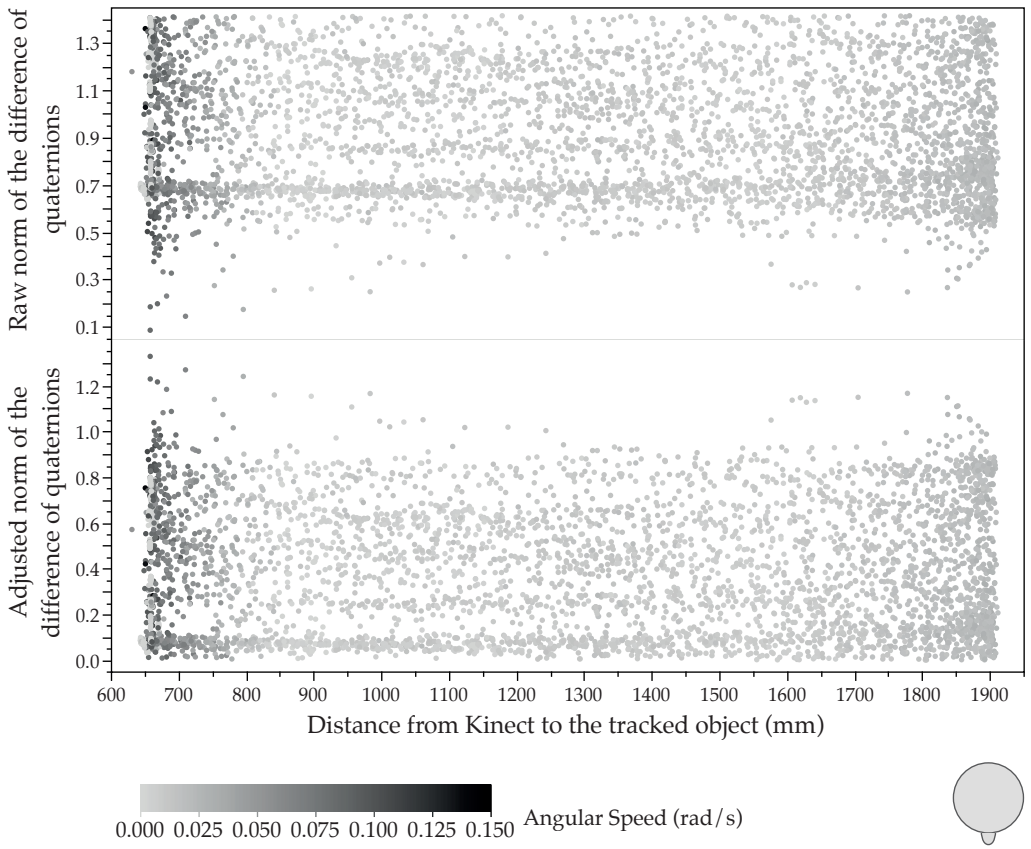

Figure 4.11: *Orientation error when tracking the coffee mug. Error is expressed as the norm of the difference of the angles between both tracking systems expressed as quaternions. Data is segmented by the angular speed of the tracked object with respect to the Kinect camera. Top: raw obtained data. Bottom: adjusted data without the initial 90 degree offset.*

dle in the estimation of the model does not correspond to the handle of the real object. When the assymetric feature starts reapearing, Figure 4.16/9–12, the system can again correctly estimate the orientation of the object. While such feature remains visible, the rotation estimation accompanies the real rotation, Figure 4.16/13–15.

Although this behavior is the most common one, there is the case where asymmetric features are visible and the tracking system yields incorrect orientations. This in most of the cases is due to an incorrect estimation after an asymmetric feature disappears, which is not corrected in the following frames when it becomes visible again.

There are cases where regardless of the visibility of an asymmetric feature, rotation estimation is incorrect.

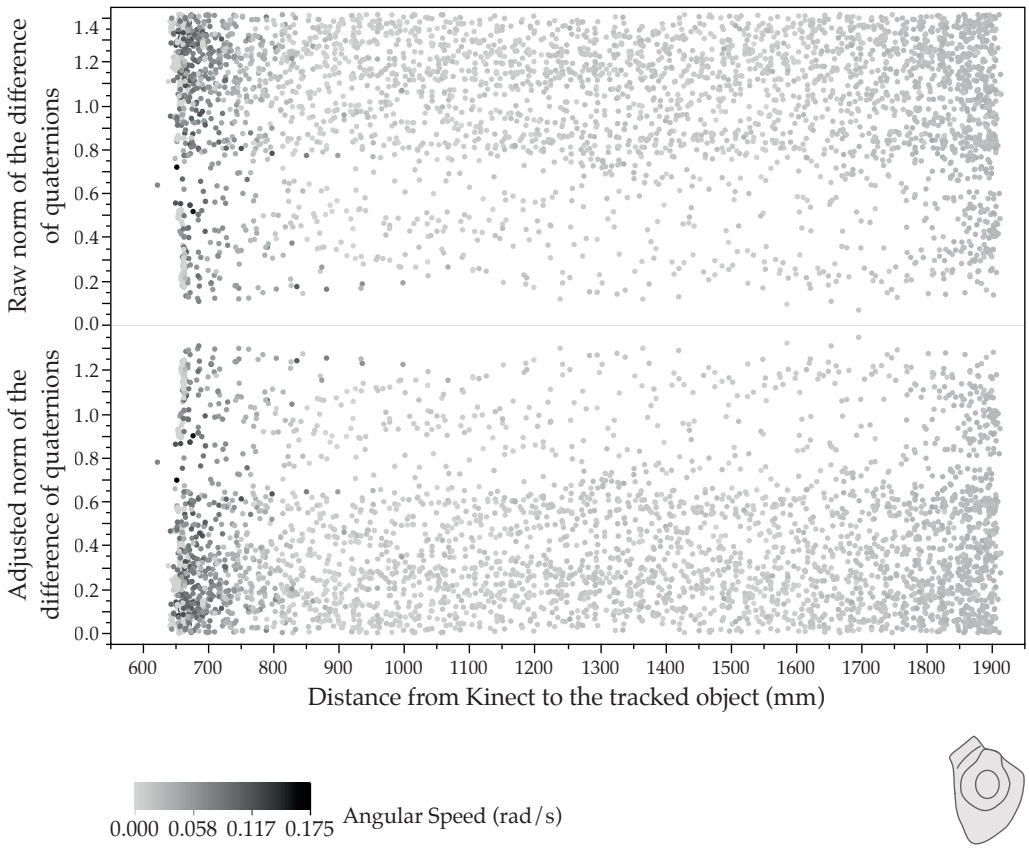

Figure 4.12: *Orientation error when tracking the plastic duck. Error is expressed as the norm of the difference of the angles between both tracking systems expressed as quaternions. Data is segmented by the angular speed of the tracked object with respect to the Kinect camera. Top: raw obtained data. Bottom: adjusted data without the initial 180 degree offset.*

Rotation estimation is not corrected because the asymmetric feature appears far from where it disappeared.

The inability to correct the orientation arises due to the asymmetric feature reappearing at a different and far location from the current estimate of the tracking system, leading ICP to converge to a local minimum and not to the correct solution. This explains why the paths in Figures 4.13 and 4.14 do not follow a constant pattern.

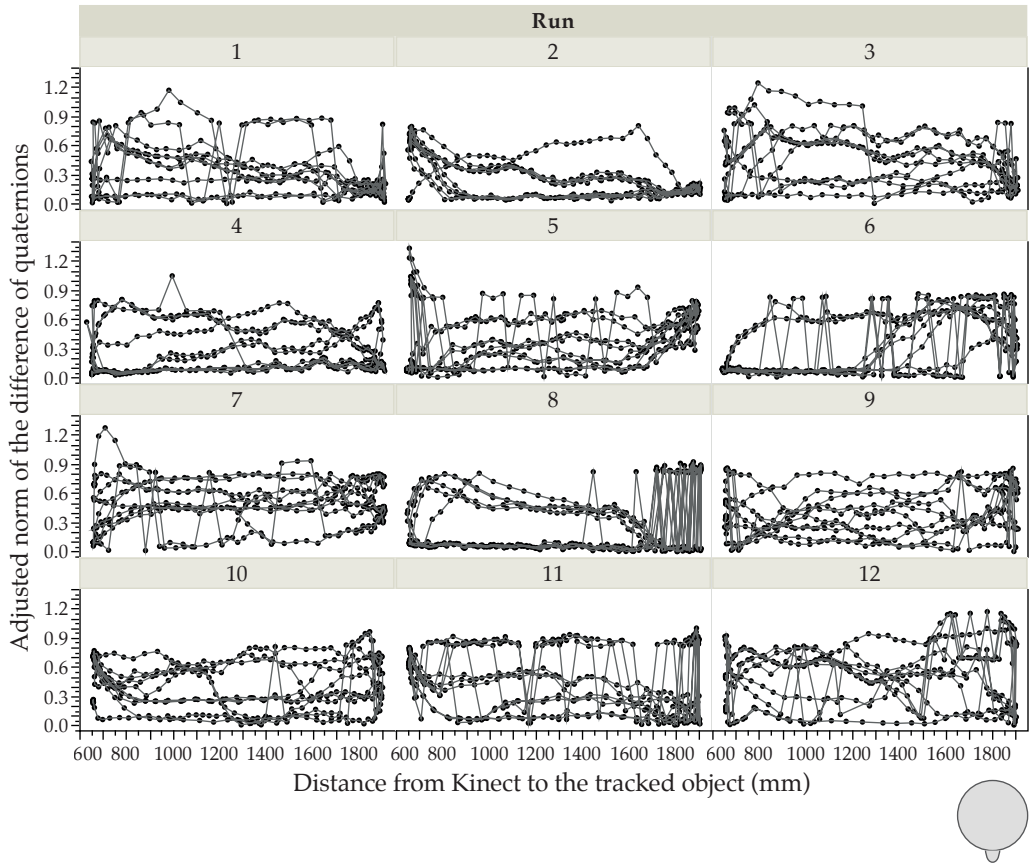

Figure 4.13: *Orientation error of the coffee mug for each test run, measured as the norm of the difference of the angles between both tracking systems expressed as quaternions. Each graph corresponds to the orientation error of a run. Points are connected by a line according to the order in which the data was generated, depicting the variation of the error as time increases.*

4.5 Discussion and Summary

Data obtained from the system while tracking objects has been compared to ground truth data obtained by a VICON motion tracking system. Tracking error has been presented as the absolute value of the differences between both systems. The average and maximum position errors, as well as the average accuracy as defined by Equation 4.1, are presented in Table 4.2. Orientation error is expressed as the norm of the differences in angles expressed as quaternions. A single run is analyzed in detail in Figure 4.13.

Tracking error has been presented and analyzed for position and orientation.

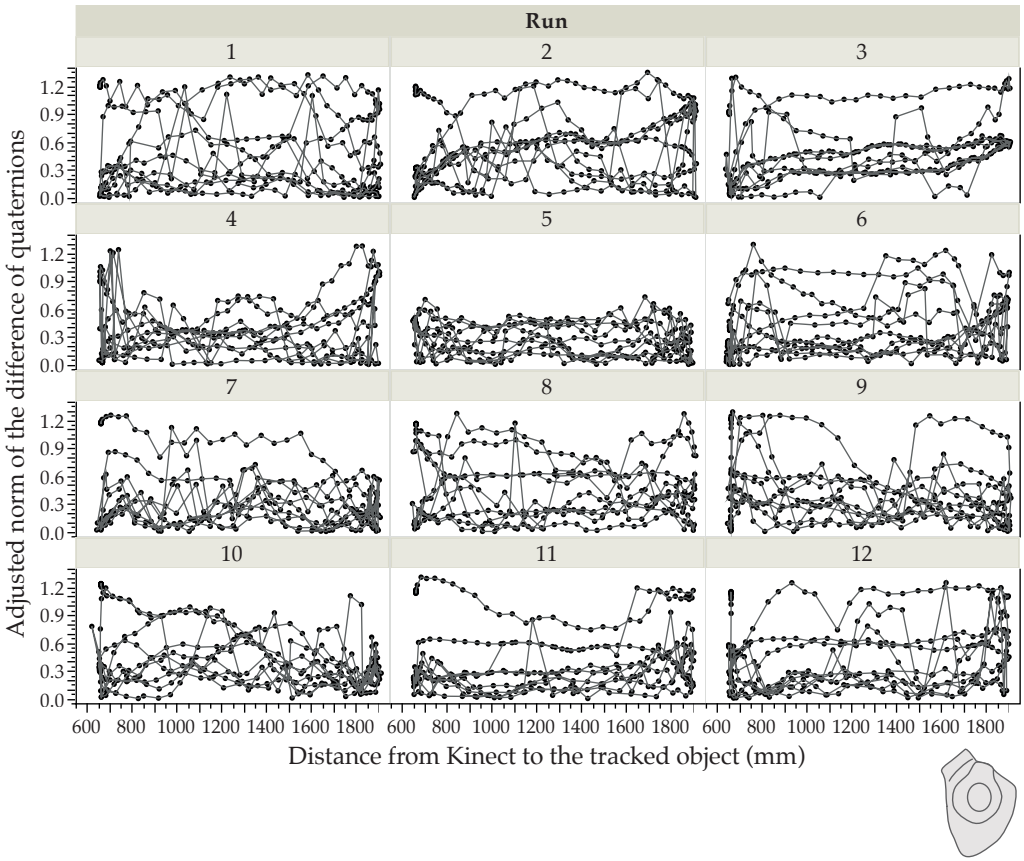

Figure 4.14: *Orientation error of the plastic duck for each test run, measured as the norm of the difference of the angles between both tracking systems expressed as quaternions. Each graph corresponds to the orientation error of a run. Points are connected by a line according to the order in which the data was generated, depicting the variation of the error as time increases.*

Can the system be used for studying user interaction with everyday objects?

Still, the main question of whether the presented system could be used to study user interaction or not is still unanswered. The typical tasks performed by users when interacting with everyday objects, shown in studies performed by Corsten [2012], were: flipping pages of a book, pointing with a hand-held object, turning the cap of bottle, flipping an object by rotating it ninety degrees, tilting an object by rotating it approximately forty-five degrees, and touching buttons on objects.

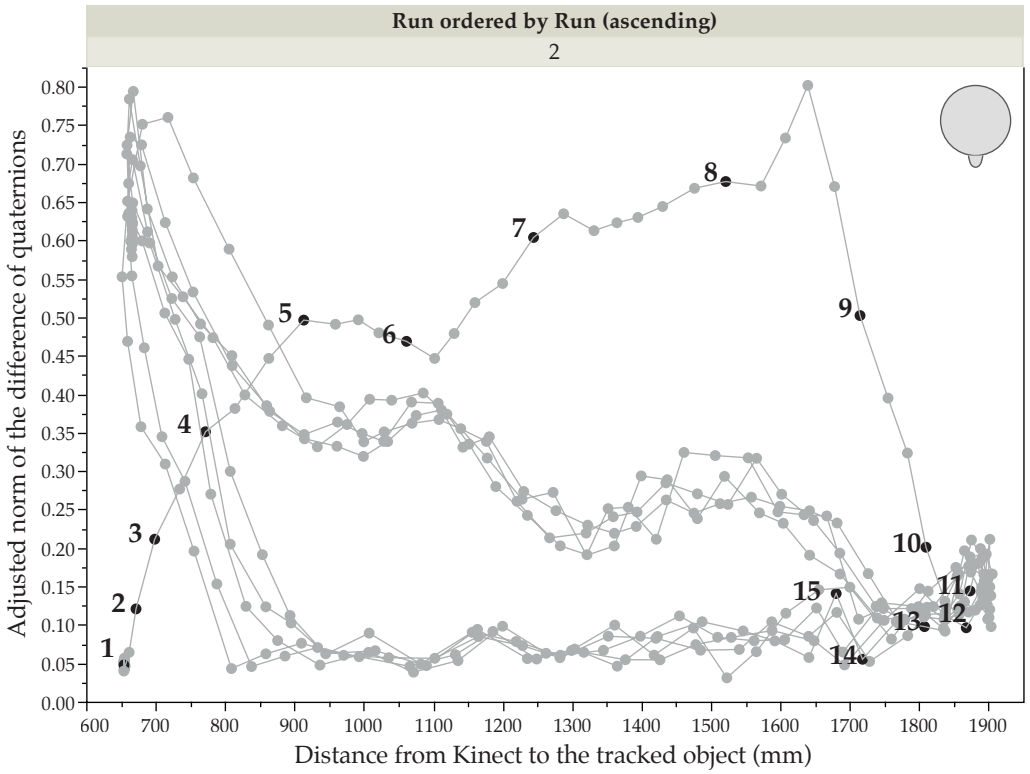

Figure 4.15: *Enlargement of the orientation error for the second run of the tracking evaluation using the coffee mug. Enumerated and highlighted points indicate a set of sequential frames.*

With respect to tracking, in a volumetric space of two meters depth; seventy centimeters of width; and twenty centimeters of height, the system would allow to detect movements of objects with the aforesaid level of error, making it suitable to study user interaction with everyday objects. The case of pushing buttons nevertheless would require a very high level of precision to be detected by tracking. For this reason touch detection is also implemented as part of the system.

Tracking capabilities allow for detecting interaction with everyday objects, as previously defined by C. Corsten, in a defined volumetric space.

Aside from the determined position tracking errors, it was discovered that:

- The largest error when tracking the object position is introduced by the distance between the object and the Kinect, rather than by the distance towards the center of the camera in the two remaining directions.

It was also discovered that: largest error is on depth axis, distance to the object and angular speed do not affect error significantly, rotation accuracy heavily depends on visibility of asymmetric features of objects.

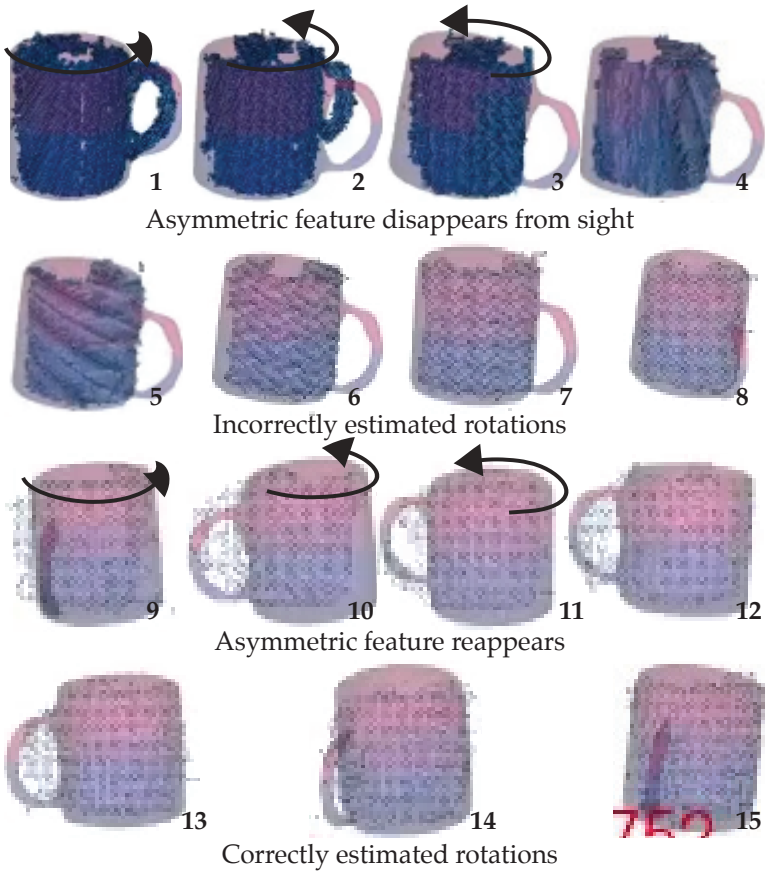

Figure 4.16: *Sequence of output frames obtained by the tracker during the second run of the tracking evaluation when using the coffee mug. The solid bi-colored model corresponds to the estimation of the object, whereas the overlaid blue dots correspond to the input scene data provided from the Kinect depth camera. Arrows on top of the models depict the rotation of the real object. The difference in the blue point cloud density is due to the increasing distance from the tracked object to the Kinect depth camera as the former moves.*

- Distance from the camera to the tracked object had no significant effect on error over tracking both position and orientation, in a range of up to two meters.
- Angular speed presented no significant effect for speeds in a range of [0–0.18] rad/s.
- When an asymmetric feature is lost from sight, an incorrect orientation is yielded by the tracker. Recovery happens once the asymmetric feature is in sight once again. Nevertheless, this is not guaranteed to happen as a result of ICP converging to a local minimum.

- The system performed better when tracking positions than orientations.

Time performance for a specific hardware setup.

To complete the tracker evaluation, processing times with a fixed hardware setup are presented. Analyzing the obtained data while tracking both objects, the frame processing time was of (247.81 $\pm$ 37.88) milliseconds—4 frames per second (FPS). The hardware where this test was performed was composed of: two 2.6 GHz Quad-Core Intel Xeon processors, 6 GB of RAM memory and an NVIDIA GeForce GTX 580 with 3072MB of RAM. The reader should note that the used system is not parallelized, and only initial pose estimation runs in the GPU.

Chapter 5

Touch Sensing Evaluation

The second capability provided by the presented system is to sense touches on the surface of tracked objects. In order to evaluate touches, the surface of the object is divided into different areas. The aim of this study is to investigate the performance of touch detection as the touch areas in the objects decrease in size while increasing in number. It is important to highlight that users were not actually required for this evaluation, since it is not their mental model what is being tested, but were nevertheless used. A machine which repeatedly touches the different touch zones could have been built and used. The decision to use users instead arose from the advantage of having a more realistic scenario, where different users have different fingers and touch objects in different ways.

An object was divided into touch-enabled areas. The effect of decreasing the size of the areas over accuracy is explored.

5.1 Study Design

For this study, only one object was used—the everyday coffee mug. The mug was divided into one, two, four and six touching zones. Once more, it is intended to evaluate if the system provides the necessary functionality to study user interaction with everyday objects without intruding or interrupting such interaction. In the case of touches, this

The coffee mug was used. Can the system be used for studying user interaction?

means that the user should be able to touch the tracked object surface as she would normally do.

When a touch is detected by the presented system, it reports its position in 3D. While it could be evaluated whether this touch corresponds to the user fingertip position in 3D, this was not the taken approach. Throughout studies conducted by Corsten [2012], users touched everyday objects in specific areas in order to interact with them. For instance, users divided an object in two areas, upper and lower, and touched such areas to perform an action which increased and decreased a value in steps—such as changing the channel of a TV set. For this reason, the focus of the touch study is to investigate how the presented system performs when an object surface is divided into a concise number of touch-enabled areas, where each touch surface corresponds to a different functionality.

Previous studies surrounding IUIs by C. Corsten show that people divide objects in zones when touching them.

In this study, the effect of increasing the number of touch enabled surfaces, which as a consequence decreases the size of each touch-enabled area, over touch detection accuracy is studied. Performance is reported in terms of overall recognition accuracy for each amount of enabled touch zones.

Accuracy when the size of touch-enabled zones decreases is studied.

5.1.1 Evaluation Method

In order to test the touch detection capabilities of the system, its accuracy is evaluated. The object was divided into one, two, four and six touch-enabled areas. Accuracy is computed as the amount of correctly recognized touches over the total amount of performed touches.

Accuracy is expressed as the amount of correctly recognized touches over the total.

The number of touch-enabled areas changed in each condition. The user had to touch a zone for five seconds, ten times each zone in each trial. Conditions were balanced using a four by four latin square. The order of touched touch zone in each condition was randomized, reducing possible effects produced by a predecessor touch zone, learning and boredom.

The amount of touch zones change in each condition. Each touch lasted for five seconds. Each zone required ten touches (randomized).

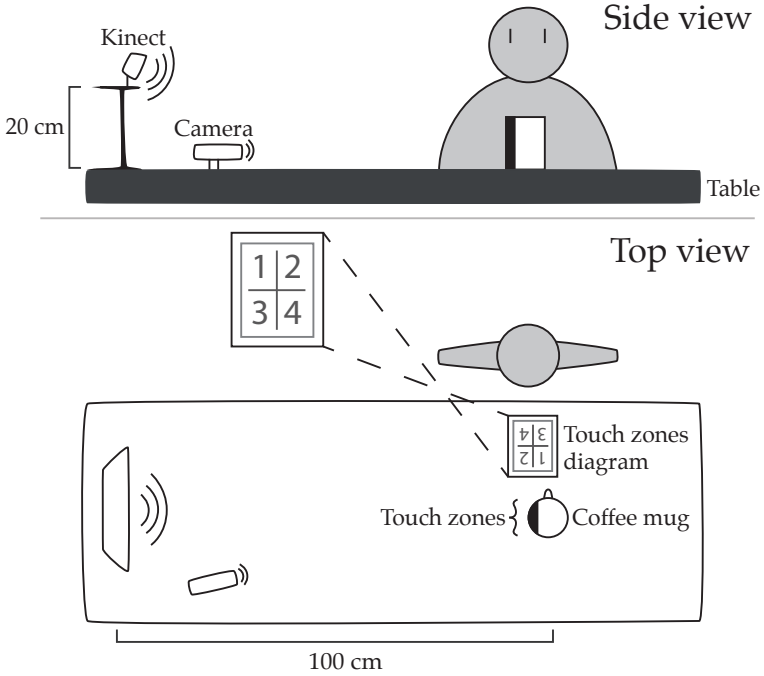

Figure 5.1: *Diagram of the setup for evaluating touch accuracy of the presented system. The Kinect depth camera is placed on a table, facing the object used during the study. A small VGA camera is placed in front of the object in order to record user interaction at all times. The user faces the mug from the side, such that the touch zones are visible by the Kinect. A small diagram is laid in front of the user with the touch areas of the current trial.*

Users were instructed to touch a certain zone, and once a touch was detected, all frames in the consecutive five seconds were recorded. The user touched the instructed zone until it was told not to do so anymore by the system. The reason for this is to ensure that there is an actual touch in a series of frames, in order to be able to use this data as ground truth. Users wore headphones from which they heard the instructions on which zone to touch and when to stop touching.

Users had to touch a zone, hold the position for five seconds and then remove their finger for each trial.

5.2 Setup

Two objects were prepared.

Two coffee mugs were prepared for the touch detection evaluation, each one used for evaluating two conditions. Front and back of the mug were used as seen on Figure 5.2. On a table, the Kinect depth camera was placed in front of the mug at a distance of one meter.

Figure 5.2: *The two mugs used in the touch evaluation study. Top: front side of the two mugs used for the one touch zone and two touch zones conditions. Bottom: back side of the two mugs used for the four touch zones and six touch zones conditions.*

Procedure: the user heard a zone to touch, touched it until it heard *stop* and removed its finger.

The user sat next to the mug, ready to be instructed on how to perform the evaluation. For each trial, the user extended one of its arms, the one she normally uses when touching touch enabled devices, and touched the indicated zone. A small VGA camera was placed in front of the mug in order to record the touches performed by the user. With this recording, it is possible to later discard erroneous touches produced by users.

Due to the fact that the user is facing the mug from the side, not all the touch zones are visible at a first glance. Before each condition, the user was shown the object with its marked zones and it was provided with a small diagram of the mug and the location of each zone. This diagram stayed in front of the user at all times.

A diagram of the touch zone locations was visible to users throughout the study.

5.3 Procedure

Firstly the user was instructed to touch different touch zones of the cup, as a training phase. She was instructed to touch a zone when she heard its number, and to hold the touch position until she heard instructions to stop. The user was instructed not to block the view from the Kinect camera to the coffee mug, which would cause tracking to be lost. Small movements of the mug were allowed, due to the fact that the system was tracking the object while detecting touches. Once the user understood the procedure and felt comfortable, the study commenced.

A training phase was used to reduce initial error due to procedure unfamiliarity.

An ordered set of conditions was withdrawn from the latin-squared balanced set of conditions. The user was presented with the first cup, corresponding to the first condition. Through the use of headphones, the user received instructions on which zone to touch. Once the first touch was registered, frames were recorded during a five second span. After this time was finished, the user heard the word *"stop"*, which indicated that the trial was finished. At this time the user removed its finger and waited to hear the number corresponding to the touch zone of the next trial. This action was triggered by the principal investigator.

Conditions were performed one by one, as drawn from the latin square. Instructions were provided through headphones for isolation.

Once a condition was finished, the user heard the phrase *"finished, good job!"* which indicated that she could remove the headphones. The cup corresponding to the next condition was presented and the study was repeated, after a short pause if the user required one. When all four conditions were completed, the user was compensated with candy and drinks. Users can be observed performing the study in Figure 5.3.

Between conditions the user was given a chance to rest.

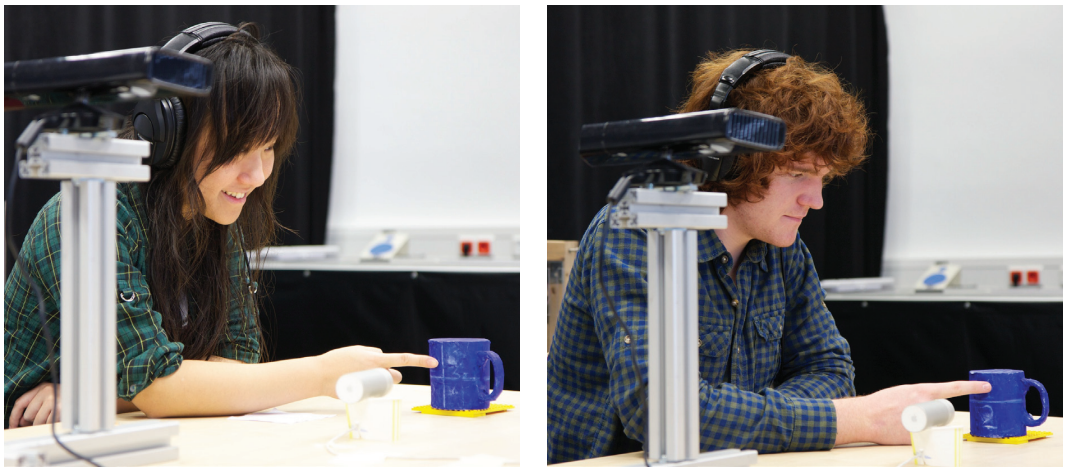

Figure 5.3: *Photograph of two users performing the touch performance evaluation study. Foreground: the Kinect depth camera on the left and the VGA camera on the center-bottom. Background: the user wearing headphones and touching the coffee mug. Photographs taken during the evaluation.*

5.4 Results and Analysis

Users demographics.

The study was performed by 16 users (15 male), ages ranged between 19 and 34 (mean 26). Regardless of the participants diverse backgrounds, all were familiarized with touch-based input devices (at least one used on a daily bases) except for two.

A contingency table shows clearly detected touches for each instructed zone to touch.

After data was collected, a contingency table indicating the hits and fails of touch detection was constructed as shown in Figure 5.4. As it can be observed, the system yielded the correct touched zone most of the times. Nevertheless, there was a large amount of times the system detected a false touch when there was an actual touch. From the contingency table, it can be observed at a quick glance how the system performs best when less touch zones are available.

Touch detection accuracy decreases as touch-zones increase.

Regarding accuracy on touch detection, the tendency is to decrease as the number of available touch zones increases. This was expected due to the fact that each touch zone becomes in turn smaller. Accuracy results for each condition can be easily compared in Figure 5.5. Although false negatives correspond to a considerable part of the error, error is dominated by adjacent touches and not by false negatives.

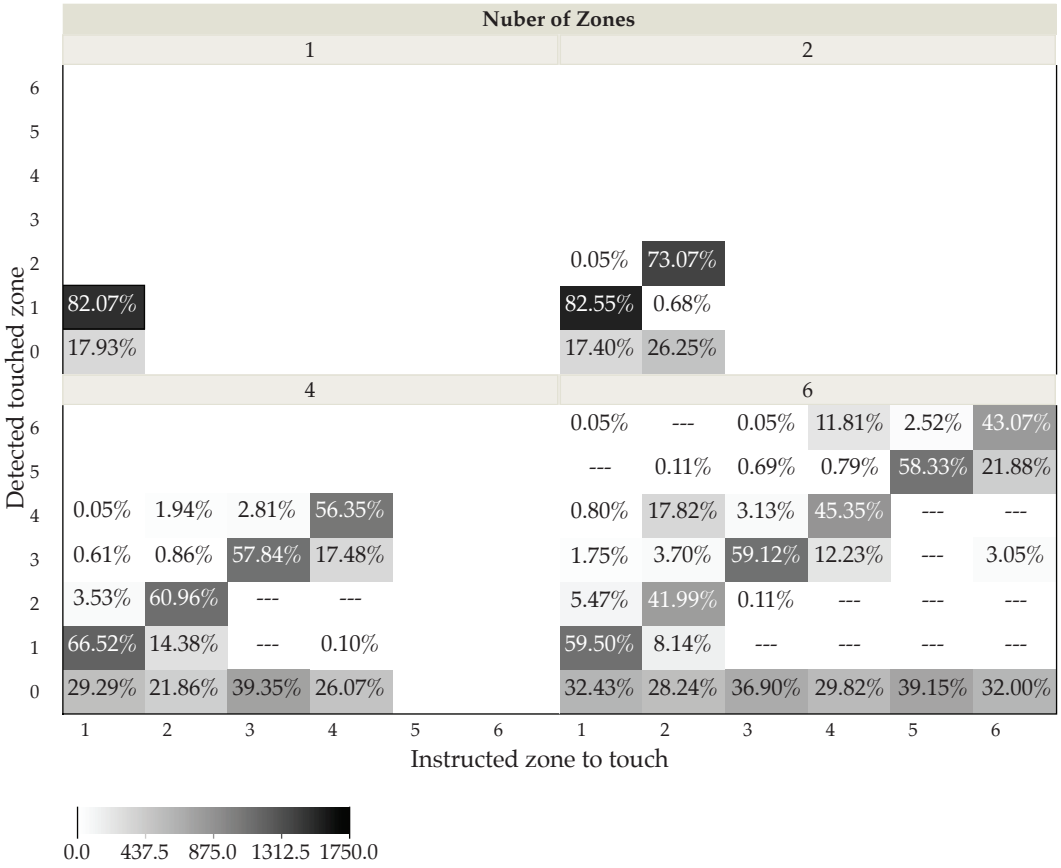

Figure 5.4: *Contingency table of touch detection accuracy, depicting detected touches (Y axis) for the instructed zones to touch (X axis). For each condition, the table shows the amount of hits for each possible touch zone, as well as the percentage it represents among the total of touch zones. Zeros indicate that no touch was detected.*

In general most of the error came from a neighbor touch zone—60% in average. Accuracy and its segmentation into false negative touches and touches in a neighbor zone can be found in Table 5.1.

5.5 Discussion and Summary

Although large error was found, overall the system performed acceptably. Further analysis of raw data revealed that, when a user intended to touch a zone, the system sometimes detected a touch and a false negative interca-

42% of the time a false negative was followed by the correctly recognized zone.

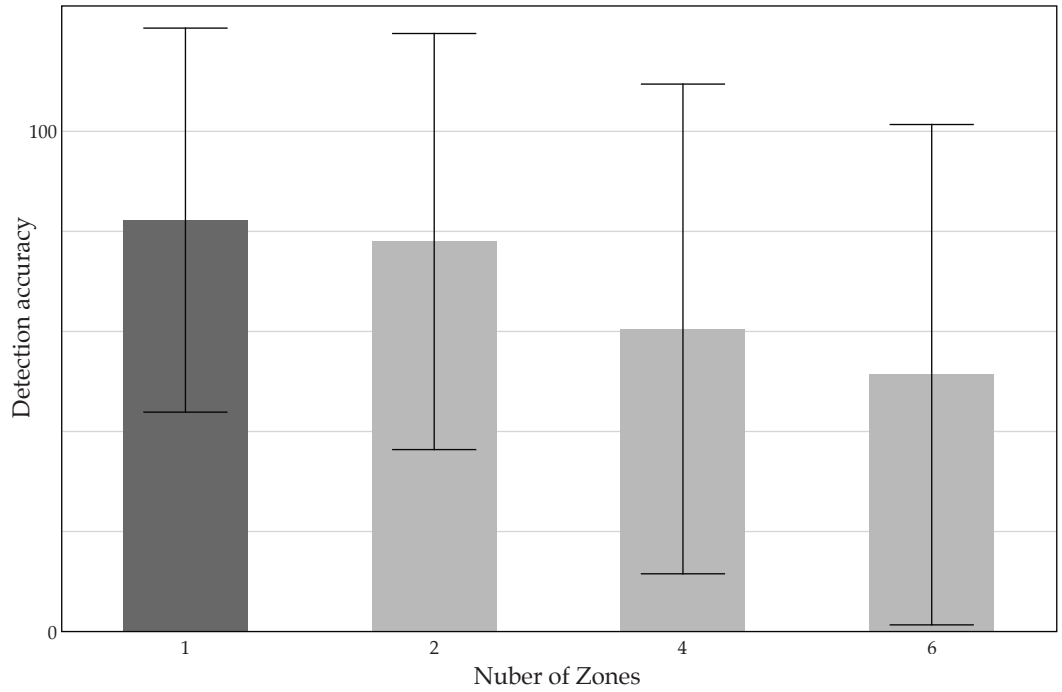

Figure 5.5: *Accuracy of touch detection by number of touch zones. Bars indicate standard deviation.*

lated. This intermittent detection of the touched zone, explains why in some cases, up to almost half the touches are recognized as false negatives. Nevertheless, in a real case scenario this would still allow to recognize touches. Although a touch is not detected in the exact frame it is generated, it is possible it is be detected in the next, making touch detection half as fast. In 42% of the occurred false negatives, the next frame was recognized as a touch, and in 60% of these cases, a touch was recognized in one of the two subsequent frames.

Left side of cup performed better than right side, explained by fingers hovering over touch zones when touching the right side.

An interesting finding from looking at the data is that, in the cases where there were touch zones placed at the left and right of the coffee mug, detection was performed better on the left side of the cup. This side corresponds to the side where the user introduced its finger. This systematic error is explained by the fact that there is a finger hover over touch-enabled zones, generating false touches on neighbor zones. When the user touches a zone that requires her finger to cross over other touch zones, there is part of her fin-

	Number of Zones			
	1	2	4	6
Size of each zone	10 cm x 8 cm	5 cm x 8 cm	5 cm x 4 cm	3.33cm x 4cm
Accuracy (%)	82.07 ± 19.18	77.80 ± 20.79	60.33 ± 24.46	51.19 ± 25.00
False Negatives (%)	17.93 ± 38.37	21.83 ± 41.32	29.21 ± 45.47	37.07 ± 47.05
False negative on neighbor zone (%)	100	100	100	14.29 ± 35.00

Table 5.1: *Touch detection accuracy by number of touch zones. It is broken down into false negative recognitions and false negative recognitions on a neighboring touch zone*

ger which remains occluding such zones. Due to the fact that the contour border of the user's finger in some cases was confused with the tip, false touches were generated.

Acceptability of performance depends on the application of the system.

Whether the system performs acceptably or not depends on its application. One and two zones will most likely have acceptable accuracy recognition in general. When having a greater number of zones, attention must be paid to the system performance while studying interaction with everyday objects. It is important to consider the size of each touch zone when interpreting the results. The user of this system might consider adapting the number of touch zones such that detection is performed with a desired level of accuracy.

Chapter 6

Summary and Further Work

6.1 Summary and Contributions

An inexpensive and unobtrusive system for tracking objects in 3D and sensing single touches was constructed, implemented, and evaluated; during the course of this work. The purpose of having such system is to be able to investigate user interaction with everyday objects, in an affordable way and without disturbing such interaction. Current systems do not present all this features all together.

An inexpensive and unobtrusive system for tracking objects in 3D and sensing single-touches was constructed, implemented, and evaluated

Some of the current tracking systems for studying user interaction require objects to be augmented, either by adding electronic components or markers to them. When an object is modified, in order to allow user interaction with it to be studied, it is possible that its physical affordances are changed as well. The presence of foreign elements can have an effect when understanding interaction with everyday objects—they might be held differently, such that markers are visible to a camera; or its weight or form might change.

When augmenting objects, chances are its physical affordances are changed, thus affecting interaction.

Systems which do not require objects to be augmented can track objects but to a limited extent. There are generally restrictions on where the objects should be placed and how they can be manipulated or touched.

Systems which do not require objects to be augmented are limited in functionality.

Most systems do not have a model of the tracked objects, thus not allowing characterization of the actions performed on them.

Other current systems that can track objects and detect finger touches without them being augmented do not use a model of the tracked object. This does not allow to assign semantic value to movement and touches. The system can track a set of points moving in 3D, but since it has no information on the object itself, how can it interpret that, after a rotation is applied, the object is being held upside down? Or how can the system interpret that a touch has been made on the handle of a teapot and not on its lid? Questions as such were the starting point of this system.

This system allows characterizing user actions on objects, thus allowing for interaction to be studied. It is not intrusive to interaction.

The contribution of this work is a system that allows assigning semantic value to changes in 3D position and orientation to objects, as well as to touches. Having a 3D model of the object allows to describe and assign an action to a change in position, orientation, or a touch. At the same time, by using a Kinect depth camera, objects do not have to be prepared in order to track them, thus allowing interaction to be investigated without modifying objects.

There is a trade-off: accuracy.

Nevertheless, there is a trade-off to not intruding user interaction: tracking accuracy. Having the measuring tools decoupled from the objects introduces error due to sensing through the distance. Accuracy of the system was evaluated in two studies—one for tracking and one for touch detection.

First study: object tracking accuracy. Obtained data is compared to ground truth data.

In a first tracking accuracy study, two objects were tracked while moving repeatedly on top of a fixed path. This path ensured that the object fully rotated and moved in all directions from the perspective of the used Kinect depth camera. A VICON motion tracker system was used at the same time to track the object, and its data was used as ground truth. The data obtained from both trackers had to be transformed such that it was expressed in the same 3D space. The difference from the values obtained by both systems was considered as the error. In the case of distances, the absolute value of the reported differences per axis was considered as the error, whereas for rotations, the norm of the difference of the rotation quaternions was used as error measurement.

Although the Kinect depth camera introduces error as the distance to the tracked object increases, it was observed that this had no effect over the tracking error in a range of up to two meters. Nevertheless, error was higher in the axis corresponding to the depth of the Kinect camera than in other axes. Angular speed has no effect on error, as long as the used hardware allows for keeping up with the speed of the tracked object.

In a range of up to two meters depth, tracking error did not increase with distance. Angular speed had no effect as well.

Regarding rotation accuracy, it was discovered that as long as an asymmetric feature from the tracked object remains visible to the camera, the system is able to detect rotations correctly. Once such feature is lost, rotation estimation does not correspond to the real world rotation. Rotation tracking is recovered as ICP iteratively converges to the correct position, when an asymmetric feature is once more visible to the camera. Nevertheless, due to the local minimum convergence of ICP, there is the case where rotation tracking does not recover and an incorrect rotation is yielded for some time.

Rotation estimation accuracy relies heavily on visibility of asymmetric features.

The second study involved detecting touches on the surface of an everyday object. A coffee mug was divided into touch enabled zones, and users were required to touch them alternatively. It was discovered that having up to two zones of five by eight centimeters of size each, the system yielded acceptable results. For more zones, and thus smaller ones, a considerable amount of false touches were detected.

Second study: touch detection accuracy.

The reason behind dividing the surface of an object into touch enabled zones, and evaluating the accuracy of the system to correctly detect touches in those zones, is that it was observed in past studies by Corsten [2012] that users touched objects in different zones in order to perform different actions. For example, a packet of tissue papers was divided into its upper and lower zones to change the channels of a TV set. This is the type of interaction that the presented system allows to investigate regarding touch.

An object was divided into touch-enabled zones, and accuracy was investigated over the size of zones based on previous studies on IUIs.

Limitations of the system are, that firstly the model of the tracked objects must be known. Also, that objects must be of a solid color in order for touches to be recognized. Due to the fact that a Kinect camera is used, not all light conditions are suitable for tracking to work. When there is direct sun-

Limitations: the model of the tracked objects has to be known. Objects must be of solid color for touch to be detected.

light for example, which contains a considerable amount of IR light, the Kinect camera is blinded.

The system is suited for studying everyday object interaction.

All in all, the system allows for studying user interaction with everyday objects. Tasks observed while studying user interaction such as pointing with a hand-held object, turning objects by a certain amount of degrees, titling objects, and touching parts of it can be recognized, with a certain level of accuracy. Although there is a considerable amount of error, it is expected that in the future it can be reduced by improving the used technology, as mentioned in the next section.

6.2 Future Work

Improvements to studies: studying the effect of geometric complexity and degree of symmetry of objects on accuracy; touch accuracy using the real 3D point of touch information.

Variations to the carried out studies, enhancements to the presented system and features that can be incorporated are presented in this section. When studying tracking accuracy, only two objects were used. Different objects could be used to perform the same tests, in order to investigate if having more geometrically complex objects has and effect on tracking accuracy. With respect to rotation, further tests could be performed to objects with different levels of symmetry to determine its effects on accuracy as well. Touch studies as well can be improved; comparisons with data corresponding to the real 3D touch point of the fingertip could be performed, instead of having objects divided into touch zones.

Current largest limitation: occlusions caused by not dense enough point cloud.

Concerning the system itself, its largest limitation at the moment is not having a more dense input point cloud, and thus occlusion caused by user hands affects tracking. Occlusion can be lessened by having more than one Kinect camera detecting the objects in the scene from different angles.

In terms of Hardware, with the use of better depth cameras, accuracy of distance measurements would be improved. Also the aforementioned boundary ambiguity problem, where depth values at the edges of objects, could be removed.

Use of better depth cameras.

In terms of Software, the implementation can benefit from certain enhancements. In the first place, there is room for parallelization of the code of the system, especially in regard of the ICP method. This would improve the speed at which the system tracks objects, allowing for faster movements to be detected. Secondly, as mentioned in Section 2.5.1, there are several variations of ICP that could be used in order to improve object tracking. Tracking error could be lessened if points would be assigned weight for better ICP convergence, or the minimized error metric could be further refined. Both improvements on hardware and software would increase tracking speed, leading to an increase in accuracy as well.

ICP parallelization for improving speed and variations for accuracy.

As well as position tracking, also rotation tracking could be improved. Even when asymmetric features of the tracked object are not visible from the depth camera, its surface colors could be used to infer how the object is being rotated.

Rotation can use color information for increasing accuracy.

With the mentioned improvements incorporated into the system, new questions on its applicability arise. With respect to tracking, would the level of accuracy be increased, even in partially occluded scenes? And with respect to touch detection, could it be achieved without the need to have a solid color model?

New questions arise: better accuracy? no need for solid colored models?

Another aspect mentioned in Corsten's work but not tackled by this system is objects capable of changing its shape. As aforementioned, this is one of the current trends for TUIs. Could it also be possible to detect deformations on objects? According to Holman and Vertegaal [2008], a organic user interface "uses a non-planar display as a primary means of output, as well as input". Bending objects allows for new ways of interaction, but the problem of detecting the current deformation state of the object arises. As an improvement to the presented system, it is proposed that in the future it is extended such that deformations on surfaces of objects can be detected.

Could deformations also be detected?

This system would enable providing feedback to the user through projecting on tracked objects.

Everyday objects need not to only be used as input devices, but as well devices can convey information to the user through them. Many dedicated input devices also have a way of outputting information through the use of embedded displays. By having precise information about the position and orientation of the object, it is possible to project an interface on its surface, which would provide the user with feedback of its actions. This could also prevent accidental use of the everyday object as an input deice, since the system could alter the user that an action will be performed or it could ask for confirmation directly on the reappropiated object.

Applications in other fields: hand tracking.

The presented system can have other applications besides studying everyday object interaction. If hand joints can be modeled and deformations detected, this system could be used for hand tracking. A hand could be tracked and the positions of its fingers known while they move. This would also certainly improve touch detection, since the tip of each finger would be modeled as well.

Applications in other fields: touchscreen tabletops.

Another field of application could be touchscreen enabled tabletops. Objects placed on top could be tracked by using the presented system, and it would not be necessary anymore to use fiducials to track objects from underneath the table. This also would allow to detect when objects are hovering, so interaction above the tabletop could also be studied. New ways of interacting with desktops arise. New scenarios would become possible, such as lifting an object on top of a stack of virtual paper displayed on the tabletop and having the pages spread out such that they are all visible to the user at once; or pointing to displayed objects on the table.

Complementing the system with other input modalities, such as speech.

Studies of other types of interaction could benefit of this system. For instance, when studying interaction through speech, a tracking and touch sensing system would provide means to know how people are manipulating objects while they express oral commands to them.

In order to inspire researchers that intend to continue this work, many different new fields of applications, as well as improvements in hardware and software have been presented.

Appendix A

Perspective Projection/Unprojection Model

In a perspective projection, a 3D scene is represented in a realistic 2D image, where relative proportions are not preserved. The more distant objects are in the scene, the smaller they appear in the projection. Points are transformed along projection lines that meet at a specific point, called the reference point or center of projection. The image is projected in a 2D plane called the *view plane*, which is located at a known distance from the projection reference point (Figure A.1) [Hearn and Baker, 1997].

A.1 Method

Suppose the projection reference point is set at position z_{prp} along the z_v axis (Figure A.2). The parametric equation describing coordinate positions in this perspective projection are as in Equation A.1, whare u is in the range $[0, 1]$.

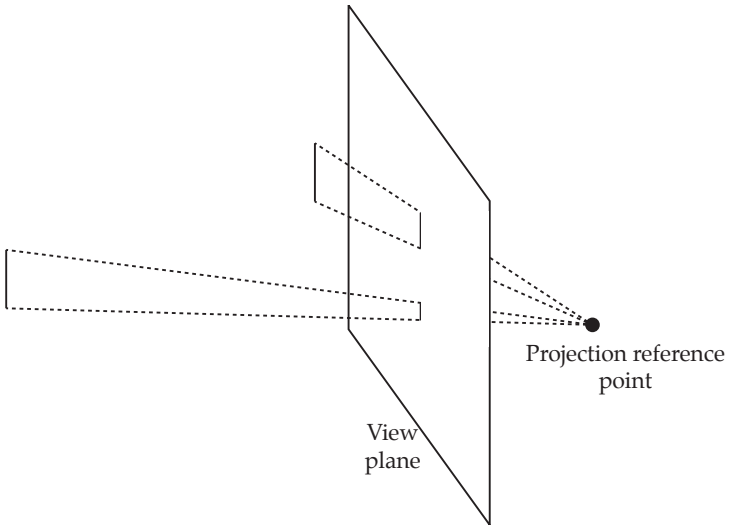

Figure A.1: *Perspective projection of equal size objects at different distances from the view plane.*

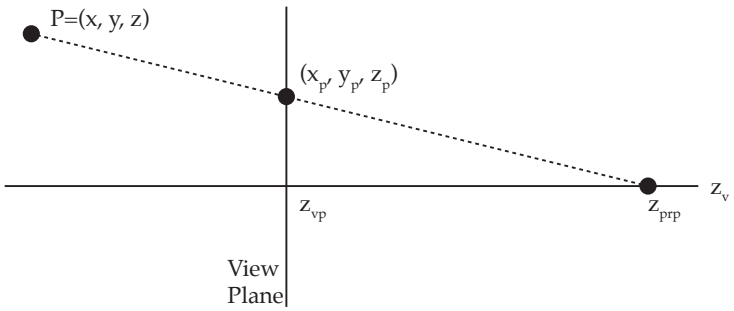

Figure A.2: *Perspective projection of a point P with coordinates (x, y, z) to position (x_p, y_p, z_p) on the view plane.*

$$\begin{aligned} x' &= x - xu \\ y' &= y - yu \\ z' &= z - (z - z_{prp})u \end{aligned} \tag{A.1}$$

The point (x', y', z') represents any point in the projection line. On the View Plane, $z' = z_{prp}$ and z' can be solved when the value of z_{prp} is specified, as seen in Equation A.2.

$$u' = \frac{z_{vp} - z}{z_{prp} - z} \tag{A.2}$$

When replacing u from Equation A.2 into Equation A.1, x_p and y_p are solved, as depicted in Equation A.3.

$$\begin{aligned} x_p &= x(\tfrac{z_{prp}-z_{vp}}{z_{prp}-z}) = x(\tfrac{d_p}{z_{prp}-z}) \\ y_p &= y(\tfrac{z_{prp}-z_{vp}}{z_{prp}-z}) = y(\tfrac{d_p}{z_{prp}-z}) \end{aligned} \tag{A.3}$$

The perspective projection transformation can be written in matrix form, as in Equation A.4.

$$\begin{bmatrix} x_h \\ y_h \\ z_h \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{vp}/d_p & z_{vp}(z_{prp}/d_p) \\ 0 & 0 & -1/d_p & z_{prp}/d_p \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{A.4}$$

, such that the homogeneous factor $h = \frac{z_{prp}-z}{d_p}$.

At this point, x_p and y_p can be expressed using homogeneous coordinates as in Equation A.5.

$$\begin{aligned} x_p &= x_h/h \\ y_p &= y_h/h \end{aligned} \tag{A.5}$$

It is common to set the origin to z_{prp} in order to simplify the above equations. In this case, $z_{prp} = 0$, for which the values of the projected point coordinates are as shown in Equation A.6.

$$\begin{aligned} x_p &= x(\tfrac{z_{vp}}{z}) = x(\tfrac{1}{z/z_{vp}}) \\ y_p &= y(\tfrac{z_{vp}}{z}) = y(\tfrac{1}{z/z_{vp}}) \end{aligned} \tag{A.6}$$

Appendix B

Implementation for Finding the Rigid Transformation Between Two Cartesian Systems based

Let A and B be two sets of corresponding points, having the points in each one expressed in two different coordinate systems. The rigid transformation that applied to the points in A yields the points defined in B can be obtained as described in this appendix. Note that the code below is expressed in MATLAB language.

```
1  function [T] = absoluteOrientation();
2
3  numPoints = size(A,2);
4  meanLeft = mean(A')';
5  meanRight = mean(B')';
6
7  M = A*B';
8
9  M = M - numPoints*meanLeft*meanRight';
10
11 delta = [M(2,3) - M(3,2); M(3,1) - M(1,3);M(1,2) - M(2,1)];
12
```

```
13 N = [trace(M) delta'; delta (M+M'-trace(M)*eye(3))];
14
15 [eigenVectors, eigenValues] = eig(N);
16 [dummy,index] = max(diag(eigenValues));
17
18 rotation = quaternionToMatrix(eigenVectors(:,index));
19 translation = meanRight - rotation*meanLeft;
20 T = [rotation, translation; [0, 0, 0, 1]];
21
22
23 function R = quaternionToMatrix(q)
24
25 R(1,1) = 1-2*q(3)^2-2*q(4)^2;
26 R(1,2) = 2*q(2)*q(3)-2*q(1)*q(4);
27 R(1,3) = 2*q(2)*q(4)+2*q(1)*q(3);
28
29 R(2,1) = 2*q(2)*q(3)+2*q(1)*q(4);
30 R(2,2) = 1-2*q(2)^2-2*q(4)^2;
31 R(2,3) = 2*q(3)*q(4)-2*q(1)*q(2);
32
33 R(3,1) = 2*q(2)*q(4)-2*q(1)*q(3);
34 R(3,2) = 2*q(3)*q(4)+2*q(1)*q(2);
35 R(3,3) = 1-2*q(2)^2-2*q(3)^2;
```

Appendix C

Provided code "Readme"

In this appendix, the structure of content of the provided code with this thesis is provided.

The provided code is written in the Objective-C and C++ languages.

In the folder *ObjRecRANSACnestk*, the XCode project is found. To build this project, open the file *ObjRecRANSACnestk.xcodeproj* and fix all dependencies such that pointers to the required frameworks are correct. It is assumed that the required tools and frameworks are installed in the system where the code is compiled.

The folder *libicp* contains the lib-icp framework used for ICP. This can be replaced by its latest version if desired. In the same manner, the folder *nestk* contains the nestk framework and can be updated by replacing this folder by the current version.

The folder *ObjRecRANSAC* contains the implementation for the Efficient RANSAC algorithm used for initial pose estimation.

In the file AppManager.mm, all parameters can be set as desired.

Once the framework is run, the first screen allows the user to select three points by clicking on them on the screen. This will determine the ground plane where objects are placed. Once the points are selected, the segmentation is shown by depicting the colors on top of the ground on red color. The user may select three points consecutively until the desired ground plane is selected for segmentation. By pressing any key, the ground plane is fixed and the user may click on Track to start the tracking process.

The position and orientation of the tracked object can be seen on the main window as coordinates.

If the user desires to transmit this information to another program, this feature would have to be implemented by modifying the provided code.

Bibliography

DepthSense. URL http://www.softkinetic.com/solutions/depthsensecameras.aspx.

D-Imager. URL http://www2.panasonic.biz/es/densetsu/device/3DImageSensor/en/index.html.

Fotonic. URL http://www.fotonic.com/content/Default.aspx.

Primesense. URL http://www.primesense.com/.

E.H. Adelson and J.Y.A. Wang. Single Lens Stereo With a Plenoptic Camera. *IEEE transactions on pattern analysis and machine intelligence*, 14(2):99–106, 1992.

E. Akaoka, T. Ginn, and R. Vertegaal. DisplayObjects: Prototyping Functional Physical Interfaces on 3d Styrofoam, Paper or Cardboard Models. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 49–56. ACM, 2010.

P.J. Besl and N.D. McKay. A Method for Registration of 3-D Shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

A. Bleiweiss, D. Eshar, G. Kutliroff, A. Lerner, Y. Oshrat, and Y. Yanai. Enhanced Interactive Gaming by Blending Full-Body Tracking and Gesture Animation. In *ACM SIGGRAPH ASIA 2010 Sketches*, page 34. ACM, 2010.

S. Challa, M.R. Morelande, D. Mušicki, and R.J. Evans. *Fundamentals of Object Tracking*. Cambridge University Press, 2011. URL http://books.google.de/books?id=DOKJGnNh9HgC.

K.Y. Cheng, R.H. Liang, B.Y. Chen, R.H. Laing, and S.Y. Kuo. iCon: Utilizing Everyday Objects as Additional, Auxiliary and Instant Tabletop Controllers. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1155–1164. ACM, 2010.

K. Chung, M. Shilman, C. Merrill, and H. Ishii. OnObject: Gestural Play With Tagged Everyday Objects. In *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 379–380. ACM, 2010.

Christian Corsten. Co-Optjects: Instant User Interfaces Through Everyday Objects. Master's thesis, RWTH Aachen University, January 2012.

Y. Ehara, H. Fujimoto, S. Miyazaki, S. Tanaka, and S. Yamamoto. Comparison of the Performance of 3D Camera Systems. *Gait & Posture*, 3(3):166–169, 1995.

A. Feldman, E.M. Tapia, S. Sadi, P. Maes, and C. Schmandt. ReachMedia: On-the-Move Interaction with Everyday Objects. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 52–59. IEEE, 2005.

V. Frati and D. Prattichizzo. Using Kinect for Hand Tracking and Rendering in Wearable Haptics. In *World Haptics Conference (WHC), 2011 IEEE*, pages 317–321. IEEE, 2011.

B. Freedman, A. Shpunt, M. Machline, and Y. Arieli. Depth Mapping Using Projected Patterns, April 3 2012. US Patent 8,150,142.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012.

M. Hansard, S. Lee, O. Choi, and R. Horaud. Time-of-Flight Cameras: Principles, Methods and Applications. 2012.

C. Harrison, H. Benko, and A.D. Wilson. OmniTouch: Wearable Multitouch Interaction Everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 441–450. ACM, 2011.

D. Hearn and P. Baker. *Computer Graphics, C Version*. Prentice Hall, 1997. ISBN 9780135309247.

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25, 2010.

David Holman and Roel Vertegaal. Organic User Interfaces: Designing Computers in any Way, Shape, or Form. *Communications of the ACM*, 51(6):48–55, 2008.

Berthold KP Horn. Closed-Form Solution of Absolute Orientation Using Unit Quaternions. *JOSA A*, 4(4):629–642, 1987.

Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. In *International Symposium on Robotics Research (ISRR)*, 2011.

Du Q Huynh. Metrics for 3D Rotations: Comparison and Analysis. *Journal of Mathematical Imaging and Vision*, 35 (2):155–164, 2009.

H. Ishii. The tangible User Interface and its Evolution. *Communications of the ACM*, 51(6):32–36, 2008.

H. Ishii and B. Ullmer. Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241. ACM, 1997.

H. Ishii, D. Lakatos, L. Bonanni, and J.B. Labrune. Radical Atoms: Beyond Tangible Bits, Toward Transformable Materials. *interactions*, 19(1):38–51, 2012.

S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, et al. KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

K. Khoshelham. Accuracy Analysis of Kinect Depth Data. In *ISPRS workshop laser scanning*, volume 38, page 1, 2011.

F. Klompmaker, K. Nebe, and A. Fast. dSensingNI: a Framework for Advanced Tangible Interaction Using a Depth Camera. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, pages 217–224. ACM, 2012.

V. Lepetit and P. Fua. *Monocular Model-Based 3D Tracking of Rigid Objects*. Now Publishers Inc, 2005.

K. Liu, D. Kaleas, and R. Ruuspakka. Prototyping Interaction with Everyday Artifacts: Training and Recognizing 3D Objects via Kinects. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, pages 241–244. ACM, 2012.

P. Mistry, P. Maes, and L. Chang. WUW-Wear Ur World: a Wearable Gestural Interface. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 4111–4116. ACM, 2009.

D. Norman. *The Design of Everyday Things*. Basic books, 2002.

I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient Model-Based 3D Tracking of Hand Articulations Using kinect. *BMVC, Aug*, 2, 2011.

C. Papazov and D. Burschka. An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes. *Computer Vision–ACCV 2010*, pages 135–148, 2011.

J.L. Raheja, A. Chaudhary, and K. Singal. Tracking of Fingertips and Centers of Palm Using Kinect. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2011 Third International Conference on*, pages 248–252. IEEE, 2011.

S. Rusinkiewicz and M. Levoy. Efficient Variants of the ICP Algorithm. In *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pages 145–152. IEEE, 2001.

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-Time Human Pose Recognition in Parts From Single Depth Images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1297–1304. IEEE, 2011.

M. Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.

A.D. Wilson. Using a Depth Camera as a Touch Sensor. In *ACM international conference on interactive tabletops and surfaces*, pages 69–72. ACM, 2010.

M. Windolf, N. Götzen, M. Morlock, et al. Systematic Accuracy and Precision Analysis of Video Motion Capturing Systems–Exemplified On the Vicon-460 system. *Journal of biomechanics*, 41(12):2776, 2008.

L. Xia, C.C. Chen, and JK Aggarwal. Human Detection Using Depth Information by Kinect. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 15–22. IEEE, 2011.

C. Yang and G. Medioni. Object Modelling by Registration of Multiple Range Images. *Image and vision computing*, 10 (3):145–155, 1992.

A. Yilmaz, O. Javed, and M. Shah. Object Tracking: a Survey. *Acm Computing Surveys (CSUR)*, 38(4):13, 2006.

A. Zerroug, A. Cassinelli, and M. Ishikawa. Invoked Computing: Spatial Audio and Video AR Invoked Through Miming. In *Proceedings of Virtual Reality International Conference (VRIC 2011)*. SHIRAI Akihiko Editors, 2011.

H. Zhang and B. Hartmann. Building Upon Everyday Play. In *CHI'07 extended abstracts on Human factors in computing systems*, pages 2019–2024. ACM, 2007.

Index

Typeset March 11, 2013