Binocular Hand Tracking and Pose Estimation
Stereo baserat handföljning och pose estimering

Javier Romero González-Nicolás
Advisor: Danica Kragic
Examinator: Jan-Olof Eklundh

February 19, 2007

Abstract

Hand gestures are important in human communication. It is difficult to transmit visual or spatial concepts only with oral communication (communication human to human) or by regular user interfaces(communication human to machine): keyboard, mouse, etc. The systems of hand gestures recognition are nowadays mainly oriented to the recognition of predefined sets of gestures. This is useful for applications like virtual mouses guided by the fingertips, sign language recognition, etc. In the context of Programming by Demonstration, these sets of predefined gestures are not enough, since a system should be able to learn any kind of "regular" gesture.

The goal of this thesis is to recover the hand pose from a stereo vision system. Two main problems are faced in this thesis: what features should we extract from the vision system and how can we extract from these features the hand configuration. The feature used in this thesis is the location in 3D of fingertips, and this information is translated into the hand pose configuration by a closed form inverse kinematics solution.

Contents

1 Introduction **1**
1.1 Reasons for estimating the hand pose 1
1.2 Focus of the Thesis 3

2 Problem Formulation and Motivation **5**

3 Related work **9**

4 Modelling **19**
4.1 Hand Model 19
4.2 Hand Model for simple matching 23
4.3 Skin color Model 24
4.4 Vision Model 25

5 System implementation **31**
5.1 System Setup 31
5.2 Hand Detector 33
5.3 Fingertip Detector 34
5.4 3D Reconstruction 35

6 Evaluation **43**
6.1 Depth estimator 43

7 Conclusions **80**

Chapter 1

Introduction

The goal of this thesis is to recover the hand pose from a stereo vision system. The possibility of extracting higher level information, as hand gestures, is also studied here. The main question this introduction is trying to answer is why we need a visual hand pose estimator. At the end of the chapter, the main segments the thesis is focusing on will be explained briefly.

1.1 Reasons for estimating the hand pose

Hand gestures play an important role within human communication. The oral communication is normally reinforced by them, in order to express feelings or emphasize sentences. Actually, hand gestures would transmit easily concepts related with visual or spatial concepts which are difficult to explain only by words. Finally, hand gestures are the main communication channel for deaf people. According to this, we can enumerate three main reasons for extracting hand pose:

- Sign Language Recognition
- Gesture based instructions and human-robot interaction
- Programming by demonstration

The first application is well known, and many systems has been built to solve it. Sign Language is basically based on a set of hand poses that are directly translated into letters. Since this set is fixed, a hand pose estimator can be programmed "from the factory" to recognize this poses and output the correspondent letters.

Both second and third applications goal is to improve the user interface, either for giving instructions and commands (the second one) or for teaching actions (the third one).

User interfaces, as we already know them (mouse, keyboard, touchscreen, buttons) are not suitable for controlling complex devices such as robots, for example. The tasks involving three dimensional actions needs very complex user interfaces, both in the real world (a plane) and in virtual ones (a CAD tool, 3D video games).

In the second application, the system should associate a hand pose in the input with an action in the set of actions it was programmed for. Sign Language Recognition can be seen as a subproblem of this application, since the robot would have a "pronounce" action, and it can be programmed to pronounce a letter when the sign language pose is performed. But it would be programmed for picking a glass when a specific gesture is done. Basically, hand gestures are the substitutes of buttons in this kind of applications.

Finally, the third application works in a different way. Instead of having a preprogrammed set of available actions in the robot, the actions can be taught to the robot by showing it how the action is performed by a human. Why should we do this? We want the robot to do a large set of high level tasks, with different levels of detail. The interface should also be easily used by normal users (non-programmers). A gamepad, that is one of the most used interfaces for robotic purposes, is very limited in terms of possible commands. A touch screen would avoid this difficulty, but the user should learn how to use it and sometimes it would be inefficient (what if the robot is in the kitchen and you in the dining room?). Speech is a good solution, but there are concepts difficult to be explained by speech (shape, colour, directions). A vision-based approach would be a good input for the system since is natural, flexible ,easy to use and could be mixed with speech for better performance. This is an instinctive solution: in a human environment, the best way to communicate with humanoid robots performing human tasks should be the human way (speech+gestures).

There is a question that is still unresolved. Why do we need a humanoid robot?

It seems that nowadays people are much more busy (and stressed) than hundred years ago. People would pay a lot of money for getting rid of cleaning the house, cooking the dinner or just picking the newspaper and a beer when they arrive home. A.I. researchers realize this fact easily since the first question people ask them is: When will we have a robot that performs the home tasks? Old people is another possible market for humanoid robots. They usually need help performing everyday tasks such as putting on shoes or getting dressed. Since our life expectancy is continuously increasing, and

young people are becoming more and more busy, who will help aging people in this tasks? This reasoning is also valid for handicapped people. Finally, it is unintelligent to waste our time in tasks that can be done automatically by robots, either we are too busy or not.

In fact, we are now surrounded by of many kinds of robots performing tasks that, some years before, had to be done by humans. They perform almost optimally their tasks at a very low price. So, where can we get benefit from humanoid robots? Since houses are design for humans, any kind of non-local task is almost impossible for non-humanoid robots. If we want a robot being able to perform a simple task for a human as making a sandwich and carrying it to the dining room we need to have a robot with, at least, is capable of precise object manipulation (open fridge, open bread packet,take cheese, etc) and has good navigation capabilities (avoid obstacles, adapt to new situations, etc). The height is also something to take into account, since the objects in a house are placed preferably where humans can manipulate them. Lastly, a humanoid robots are kinematically able to do a large percentage of things humans can do. So this give us the possibility of include eventually new tasks to do in its schedule. Programming by demonstration can help us to equipp robots with these tasks in a natural way.

1.2 Focus of the Thesis

A programming-by-demonstration environment needs many different components:

- Different input estimators: speech recognition, hand pose estimation, etc
- Higher level action recognition
- Human-Robot adaptation layer
- Initial task planner
- On-line task planner with feedback inputs

As it can be seen in this list, it is a huge problem not feasible for a master thesis. This master thesis will focus in the hand pose estimation problem. This problem will be divided into two main layers:

- Image processing: binocular hand tracking based on color segmentation. This layer is mainly based on the system proposed in [6, 8, 7, 9, 10].

- Hand pose extraction: fit the hand model into the features extracted from images. The hand model is built to facilitate the hand pose recovery from particular features of the system.

Chapter 2

Problem Formulation and Motivation

Since usual programming languages are not feasible for a regular user, teaching by demonstration becomes the best way to teach a robot what to do in order to achieve some goal. Teaching manual actions by demonstration needs definitely to estimate the hand pose precisely, in order to recognize what the teacher is doing, and learning accurately how to perform the action. In this context, the hand pose estimator becomes a key component in the Teaching By Demonstration framework.

If we consider the whole problem of teaching hand gestures by demonstration, we can realize that it is not a trivial problem at all. In order to introduce the problems, we present a simple model for the system in the figure below.

This figures shows:

1. An input device that records information from the input data, usually one or two cameras

2. A module which extract features from the information. Usually this features are extracted from a specific region determined by a hand recognition module. The prediction of the hand pose would be valuable for improving the features extraction.

3. A module which performs the inverse mapping from features $f(hp)$ to the hand pose hp by a complex inverse function $f^{-1}(y)$

4. A tracker which predicts the hand pose for the next step.

5. A module which recognizes the action from either features or hand pose, or both.

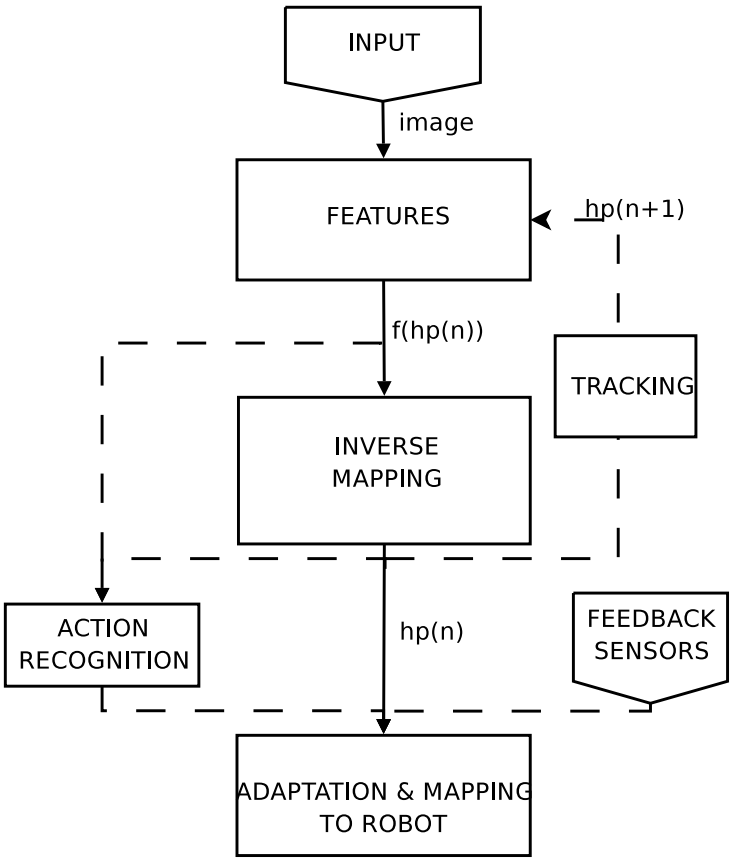


Figure 2.1: Module diagram of the hand pose extractor system

6. Sensors that provide closed loop information to improve task accomplishment.

7. Finally, the module which maps the human joint angles to robot joint angles, taking into account the differences between both, the action which is being performed by the human and the feedback sensors.

After this small summary of the system, we can go deeper into the problems that each module presents to us.

The first step is to recognize what is a hand in order to fix our attention on this part of the image/images provided by the cameras. Considering the hand of a specific person, it is a non-rigid, highly deformable, non-textured object. If we want also to be capable of recognizing hands from different persons, we have to take into account that the color, size and length of different segments of the hands are not constant at all. More difficulties come when we take

into account changes due to the environment: illumination changes, object occlusions, etc. All these reasons convert the recognition of the hand to a difficult problem.

When the hand recognition process is done, we need something similar to a "mask" in the image, saying what is a hand and what is not. Some methods use directly this hand image to extract the hand pose. This methods use training images with known hand parameters in order to compare new samples with the previous ones. This would be done directly by a database, or indirectly by some non-linear learning methods such as Artificial Neural Networks. However,the amount of required training samples increases with the variability of the input. Since the images of a hand would vary in many ways (color, shape, size, viewpoint, etc), a simpler representation would be desirable. Usually some features should be extracted, in order to have a lower dimensional representation of the hand. The election of this extracted features should be done together with the election of the method to extract the hand pose from them, since some feature would be good for some methods, but bad for others.

One feature we should treat specially is the 3D information. Hand gestures have an intrinsic 3D behavior: if you want to explain the robot how to go to your room, the gesture straight ahead (moving the hand in the depth direction) would be too complex to recognize in a 2D scenario. The hand pose can be extracted from a 2D scenario only if we constrain the hand shape to be constant, forbidding different hand shapes. For these reasons, acquiring 3D information is quite desirable in this kind of devices. The drawback of this option is that 3D reconstruction of the scene is not easy at all, so we have to develop a system capable of matching the points we want to extract in both images, and perform the triangulation given the camera parameters. Since cameras are not perfect devices (disalignment, distortion, uncertainties) and matching points are sometimes almost impossible to find(points out of viewpoint in one camera, occlusions, etc), this is a non-trivial problem.

The feature extraction should return a representation of the hand (either 2D or 3D) that makes the inverse mapping easier(extracting the hand pose from the features). Due to the high dimensionality of the hand representation (the hand pose is represented by 10 to 30 degrees of freedom), inverse mapping is a complex task to solve. Usually some optimization and iteration is done in this step, which makes the inverse mapping one of the most expensive tasks in computational terms.

At this point, it should be said that temporal information is very important in this kind of systems. The complexity of the task could be decreased drastically if we take into account the continuity of our gestures and actions. Predictions of the next hypothesis can be made based on the current (and

past) hypothesis. This information is very valuable, for two reasons: it is a first approximated solution, that can be then optimized to the best solution; this prediction can be used to reject solutions that does not keep the assumption of continuity. It should be taken into account that some features are more difficult to predict than others, depending on the model. As an example, the 2D projection is not a linear model of observation, while 3D projection is linear. For this reason, the model for tracking 3D points by a Kalman Filter will be simpler than for the 2D model. The predictions would be done inside a module or between modules. An example of the first way is to extract fingertips from the hand, and predict the position of these fingertips in the following step. This prediction would be done in the second way as following: we extract fingertips, estimate the hand pose from the inverse mapping, predict the next hand pose and get then the position of the fingertips.

After extracting the handpose, there is still a lot of work to do before being able to map the hand pose to a robotic hand. Instead mapping directly the joint angles extracted from human hand to the robot, we still have to cope with some problems: collision between fingers, joint range limits, choose what joint angles should we map if the number of degrees of freedom is different. Actually, we usually want a better mapping than just a direct mapping, since the robot hand will be different from human hand. Some adaptation will be needed in this case. In the case of manipulating objects, this adaptation from human to robot should definitely be mixed with some closed loop observation and adaptation; slight errors in hand pose would mean unsafe grasps that can not be improved without pressure or torque sensors.

In the phase of "adaptation" mentioned before, it is sometimes needed to obtain the information from a higher level of abstraction. The robot would need to know what is the goal of some part of the action. For example, there are two different steps with very different goals during a grasp: approximation and grasp. During the approximation, the main goal is to place the hand close to the object in a pose suitable for grasping. During the grasp, there are some hard constraints (object should not fall) and goals (move, rotate, etc). These different goals should be taken into account in order to perform the action well: during approximation the hand pose is not as important as in the grasping. Mapping can be adapted if we recognize the action performed and the steps to follow. For this reason, a module for action recognition is desirable in such a system. The action recognition input may be image features, the hand pose or both. For some actions, it may be easier to recognize the action from image features (observing fingertips we would recognize pointing gestures easily), but for other tasks the whole hand pose would be necessary to discern the action.

Chapter 3

Related work

Human Computer Interaction has been an active field of research for a long time. The devices we use today for communicating with computers (mainly keyboard and mice) has been used for decades without significant changes, and they are becoming probably a bottleneck in new technologies, i.e. virtual environments in 3D. The first approach to an improved interface between human and computers/robots was automatic speech recognition. However, as we discussed before, speech may be insufficient when we want to use spatial or visual concepts.

Devices capable of extracting hand gestures are complex, and computing the data provided by them would be very expensive in computing terms. This means that the research about hand gestures extraction is relatively modern. During the late 80s new approaches based on hand gestures became popular within the research community. An overview about the available systems in the earlies 90s can be found in [66]. Three kinds of systems are treated there: Optical tracking, Magnetic tracking and Acoustic tracking. Magnetic and acoustic tracking are briefly explained since this thesis is based on optical tracking. Optical tracking had two variants: marked systems based on infrared LED's and silhouette matching. The last one was the only non-invasive system explained in this article, and the most similar to our system. The main problems of this kind of systems were:

- Cameras have low resolution
- Cameras have low frame rate
- It is difficult to deal with finger occlusions
- General immaturity of computer vision techniques

We will explain later the current state of the art related with these problems. Magnetic gloves have classically been a popular approach. They provided accurate measurements (better than 0.1 inches in position and 0.1 degrees in rotation), robustness to occlusion and high frame-rate measurements. On the other hand, these systems need the user to wear additional things (gloves, sensors) and usually to carry cables. Furthermore, the glove needs to have a reasonable number of sensors in order to extract appropriately the hand pose. Acoustic gloves like Power Glove are less accurate than magnetic gloves, but the price of acoustic gloves is much lower. The basics of both gloves are similar. The main disadvantage of gloves is that they are invasive, and it makes glove-based devices unuseful for general purpose interfaces, relegating them to environments and to scenarios where high accuracy is the most important feature (i.e. simulation of surgery). Gloves are useful also for providing ground truth data to other less accurate systems. Some more recent projects based on glove devices can be found in [59] [19].

Nowadays, new hardware and different computer vision techniques are being used to overcome the difficulties mentioned above. Resolution become the smallest problem in modern cameras. Resolutions of 1024x768 are common in the present, and this resolution is more than enough for our purposes. In fact, some experiments show better performance with lower resolution.

Frame-rate is still low in common, non-expensive cameras. We can realize this in the figure below 3.1, taken at 30 frames per second. Also another problem in fast motion is noticeable in the second picture of 3.1: motion blur. When high-resolution is used, the interface bandwidth would become

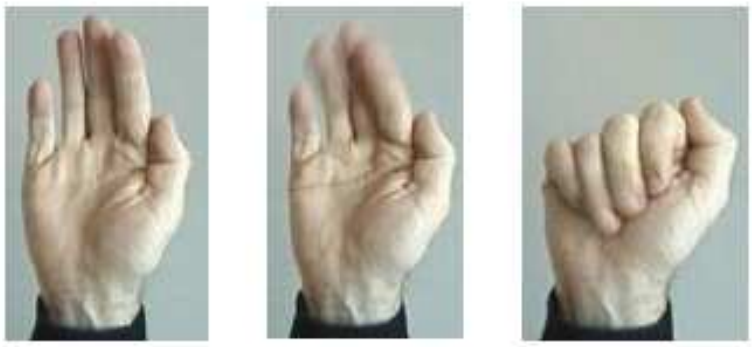


Figure 3.1: Three consecutive frames at 30 frames per second

the bottleneck for the frame rate. In spite of the slow motion assumption still being the most common procedure, new techniques as interpolation [68] have tried to solve the problem.

Tomasi realized in [68] that, when using the sign language alphabet, the assumption of slow motion is usually violated. This usually generates abrupt transition between frames. When the classification of a pose has a residual error (based on the difference between an image and projected features) higher than a threshold, the tracking estimation is solved by interpolation. This procedure solves two problems: abrupt transitions and missclassifications due to noise or motion blur. Each transition between different signs takes at least a minimum number of frames f_{min}, and half of this number is labeled as an error, in order to force interpolation during the transition. In this way, the transition between signs is smoother, and abrupt jumps to other configurations due to noise or motion blur are avoided. If the motion between two consecutive frames is not continuous, problems would appear in the interpolation procedure: while crossing fingers, interpolation tries to output a configuration with collision between fingers. Since the number of different signs in this database is small, crossing fingers signs could be redrawn with a non-problematic pose (with fingers almost crossed). Unfortunately, this is not scalable for general purpose trackers.

Occlusions can be avoided in different ways. The system used in [22] is composed of three cameras. Since the viewpoint of the cameras is very different, occlusions are solved by at least one camera. The drawback of this device is that the "head" size is quite big, forcing this kind of devices to be static. The performance of this system can be seen in the url [18]. The most common way to handle occlusions is via software. Tracking and background subtraction are tools very useful for this purpose. The background subtraction is a field of continuous research nowadays, and can be applied both in simple and cluttered backgrounds. In [58] background subtraction and tracking are mixed in order to track objects through occlusions. Each pixel is modelled by RGB means and deviations, as well as brightness and chromaticity deviations. The differences over three frames are computed and moving objects are extracted to the background. A given pixel is classified into original bg, shaded bg, highlighted bg or foreground, depending on the value of the four components described before.This classification is very useful to avoid false positives while detecting movement. Finally small isolated spots and holes are filtered. This background extraction system is further explained in [32].

After the background subtraction, a high-level tracking is performed. The foreground is grouped into connected components described by the bounding box where the component is placed and an image mask which tell us which pixels belongs to the foreground. Building a distance matrix one can link each foreground regions with existing tracks. Each region can be classified as existing, new, merging or splitting track from distance matrix information.

Once a track is created, an appearance model of the object is initialized for the final appearance based tracking. This appearance model consist on the RGB values for the region, basically a probability mask. The algorithm follow the steps below:

1. Centroid locations are predicted using a first-order model
2. If new collision is detected, it tries to find the location of best fit for each object
3. Given the best-fit location, disputed pixels are classified using maximum likelihood classifier with a simple spherical RGB model
4. Fewer disputed pixels imply greater depth. Those with few visible pixels are marked as occluded.
5. All pixels are reclassified, assigning disputed pixels to the foremost object.

After this, localization step is performed in depth order.

In [25] a non-parametric model for the background and foreground color is used. Gaussian kernels are used to compute the probabilities of being foreground or background. The foreground is segmented in different levels in order to separate persons (into head, torso and bottom) and to segment groups of people. These segmentations are valuable to assign relative depths and solve the overlapping problem.

In [47] a new method for tracking objects is explained. Kalman Filters are used to update an intensity model of the object, instead the common approach of updating the position. This method can handle temporal occlusions by stopping the update process when an occlusion is detected. Is difficult to detect the end of the occlusion since the remembered model can be very different (due to orientation changes i.e.), so sometimes the duration of the occlusion is limited to L frames.

Finally, [72] is based on a layer model. A motion layer is an image that performs a coherent motion. Motion layers can be foreground (free movements) or background (if a back- ground moves, all background share this motion). Each layer is represented by the motion (usually restricted to foreground layers), shape and appearance. The behaviour of layers is modeled as a Hidden Markov Model. The state of the objects in the scene is estimated by maximizing the posterior probability of this state given the previous state. Since the state space is very large, a sub-problem optimization is performed step-by-step. Other system for solving self-occlusions is explained in [53].

As we described before, the problems formulated 15 years ago are solved (low resolution, low frame rate, finger occlusions and immaturity of computer vision techniques), or can be avoided with different approaches. So, we will continue with the analysis of the related work describing how previous systems has solved the problems related before.

Concerning the input devices, the gloves used 15 years ago has been widely exchanged by cameras. As we said, it is non desirable at all that each time a user want to teach a robot something to do, he would have to wear a glove and carry the cables while performing the demonstration. This is the main advantage of camera devices against glove-based devices.

Nowadays, the most common device used in this kind of systems is a monocular color camera [16, 5, 12, 11, 8, 7, 68, 67, 56, 49, 20, 21, 70, 52, 24, 46, 50, 31, 33, 69, 38, 14, 55, 48, 29, 41, 61, 64, 62, 63, 13, 39]. There are many reasons to choose a single camera system: it is cheaper (half the price!) have one camera than two, it is lighter in computing terms (half image processing and no 3D-matching process), and we can extract 3D information from a monocular camera in constraint environments like manual manipulation of objects (fixed length of hand segments). If this is the case, why during the last years more and more systems [40, 6, 9, 10, 30, 35, 34, 23] are using stereo cameras? One reason can be explained with other question: why humans have an "embedded" stereo-system? Humans can not apply constraints to extract 3D information because a usual human environment is not constrained. Humans have to be able to solve 3D extraction in any environment formed by different hands with different shapes, different objects and different backgrounds. And a system designed for general purpose hand pose estimation should have this ability also. But this is not the only advantage of binocular systems. The monocular 3D extraction is based on a comparison of the image to a model with fixed lengths. It was said before that the hand is a very complex object, highly deformable. This presents two disadvantages of monocular systems: the 3D extraction is less exact than in a well calibrated stereo system due to model imperfections; and, if model is updated, all the calculations and 3D extraction should be reprogrammed.

Cameras have also drawbacks. The price of the non-invasive interface is the complexity of the hand pose extractor. The relation between a hand image (or multiple hand images) and the hand pose is not trivial at all. This means that the systems developed have to constrain either the environment or the actions in order to apply some constraints that make the necessary steps easier. One constraint can be to forbid gestures out of a predefined set [12, 11]. This set would be an alphabet like American Sign Language alphabet [68]. Some systems can not work in cluttered environments [68, 46, 50, 34]. Other schemes model the hand with simple models that do not allow

some general position or rotation [68] or some finger movements [30, 18]. Sometimes a very specific environment is needed to work with as seen in the Figure 3.2, from [14, 55, 48]. There are approaches designed for very specific

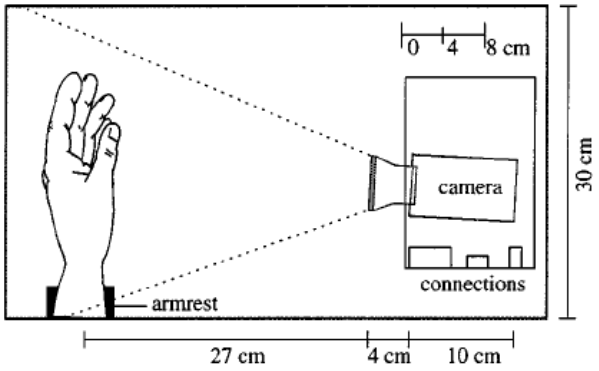


Figure 3.2: System used in [14, 55, 48]: hand is placed facing the camera, with an easy background

purposes like virtual mouse which has highly constraint models [70, 20, 49].

Despite color monocular and binocular cameras are the most used devices nowadays, there are more possibilities about the information acquisition. It is strange to find systems with more than 2 cameras. As we said before, the 3-camera system showed in [18] has very different points of view, which makes it capable of avoid occlusions very well. The possibility of extending the system to multicamera is taken into account in [35]. Color is not the only feature we can extract from the cameras. In fact there are many systems that use gray level images instead the color component [12, 21, 52, 14, 55, 48]. In [49] infrared camera is used in order to perform an easy hand extraction without significant computing costs. Very accurate depth maps are extracted in [27] from the shadow analysis with a multiflash camera. More complex devices, as the one used in [43] can be compulsory if accurate 3D hand shapes are needed. Finally, glove-based devices are still in use, mainly due to the accuracy they provide. They can be used for obtaining ground-truth data [40, 16], for improving the knowledge of hand motion and the hand model [61], for building a database of motion and joint angles [56] or as a final device [59, 19].

Referring to the features extracted from images (we will assume we are recording either monocular or binocular images), there are few different approaches. One of the most extracted feature are the edges [12, 11, 27, 29, 38, 41, 14, 55, 48, 61, 64, 63, 70, 50, 46, 34, 35]. Setting the threshold of edge extraction is a difficult task: if it is too low, a very cluttered scene

will be extracted; but if it is too high, important details of the scene would be lost. Since it is important to get as much information as possible, the thresholds are usually low and a lot of noise appears in the edge maps. For this reason, the edges need usually a refinement process [12] or are only the base for extracting another feature, as silhouettes [38, 50, 46], contours [12, 11, 33, 61, 64, 63], fingertips [5, 34, 35, 14, 55, 48, 70]. Theory of edge extraction can be found in [42], and an image segmentation system is explained in [26].

The other "feature" that is normally used is the result of a color segmentation [16, 40, 6, 8, 7, 9, 10, 5, 68, 12, 11, 67, 56], usually trying to locate the hand by color. Color segmentation can be used as a base for "higher level" features as edges are. Silhouettes can be extracted from color segmentation [6, 8, 7, 9, 10, 68, 67]. Geometric features as hand centroid and principal axis would be valuable information also, and are easily extracted from the color segmentation [6, 8, 7, 9, 10, 68, 67]. Fingertips position can be estimated directly from color segmentation by a voting process [5] or by curvature measures from the silhouette [6, 8, 7, 9, 10].

Instead the popularity of this measure, it is quite difficult to model robustly the skin color due to both illumination changes and variety of skin color between persons. For this reason, sometimes color segmentation can be made easier if we put some marks on the whole hand [40, 31] or on the fingertips [21]. But markers destroy the best advantage of visual hand pose estimation against glove-based hand pose estimation: the non-invasiveness. Other possibility to facilitate the color segmentation is to set some constraints on the background and the elements that would appear in the scene [68, 46, 50, 34, 14, 55, 48]. This approach has other drawback: generality is lost.

Other features less popular can be used, as optical flow [41], shading information [41, 27], etc.

The 3D information about the hand is a higher level feature more complex to be extracted. If we apply constraints to the hand size, this information can be extracted from monocular images [5, 52, 50, 31, 69, 14, 55, 48, 41, 61, 63]. Some of them apply some closed form computation to extract the information, and others apply some optimization to fit a model into the image. We can extract also 3D information from monocular cameras by comparing the images to an image database with 3D information stored [12, 11, 68, 67, 56, 64, 62].

Recently, stereo systems have become more popular. The price of the cameras is much lower than before. The extra computing effort due to image processing in the second camera is affordable in modern computers. The advantages of having a stereo system are mainly related with the 3D extraction.

Extraction of depth from a stereo system is done mainly in two steps: identify points in both images and extract depth by geometric equations. In this process, no information about objects size or shape is needed; the extraction is done in a more general way. As a drawback, the identification of points in both images is difficult. Argyros [6, 9, 10] matched contour points based on approaching both contours by a rough estimation follow by ICP optimization. The results of this "marriage" problem are good, but the underlying assumptions in the ICP optimization are not always kept (5.4). If two cameras are not available, an stereo system would be emulated with one camera and a mirror system ([23]). In Delamarre system ([23]) another approach for matching points in stereo images is taken. If the position and orientation of both cameras, as well as internal parameters of them are known, the possible correspondences between points in the image are restricted too much. From the epipolar geometry model of the stereo system (more detailed information in 4.4), we know that the correspondent point lie on a line whose equation can be easily computed given the fundamental matrix (matrix which define the stereo system given the intrinsic and extrinsic parameters of the camera, see 4.4). The search along the epipolar line is done by correlation windows. The model used in our system is similar to this.

Another important feature to emphasize are the fingertip locations. For example, in a pinch grasp the contacts between the hand and the object are only the fingertips. For this reason, the position of fingertips in this kind of grasp is crutial in the stability of the grasp. Fingertips are also commonly used in natural language for explaining directions, routes, etc. Because of this, fingertip is normally used as a pointer device in computer interfaces based on vision [70, 20, 49, 10]. The fingertip extraction have been used also as a feature from which hand pose can be recovered [5, 34, 35, 14, 55, 48, 33]. The fingertips can be seen as end-effectors in a kinematic model of the hand, converting the problem of extracting the hand pose in a inverse kinematics problem. Many different ways of detecting fingertips have been used. The extraction of fingertips would be simple if we apply some constraints. This is the case of application-specific systems, as computer interfaces based on vision [70, 20, 49, 10]. The assumption in these systems is usually that fingers are stretched out, allowing measures based on curvature [10], specific pattern matching [20], or simply select the farthest skin pixel from the centroid [70]. In more general systems, fingertip extraction is more complex. Gabor filtering and Neural Networks are used in [14, 55, 48] to extract fingertips. In [49] Hough transform is used to search a circular pattern in the edge map extracted from the image.

Once the features have been extracted, the hand pose should be extracted from them. This problem can be raised as an inverse function search: given

the forward function (extract the features given the internal parameters and the model) we should estimate what is the inverse function which returns the internal parameters given the features. This inverse function problem has been widely studied. There are two main ways of solving it:

- If we do not have any knowledge about the function, we should use training data to extract the inverse function. Training data (pairs of input and output of the forward function) can be used as a database. Approximate functions with training data is one of the common uses for learning procedures, such as neural networks. The provided data can be used for training a learning method which approximates the inverse function.

- If we know something about the underlying function, we can constrain the search and apply numeric optimization for local fitting. For example, we can make a rough approach and then optimize the results.

- If the forward function is simple enough (or if it is simplified for this purpose), it would be an invertible function with closed form inverse solution.

The complexity of hand appearance makes the database approach common in hand pose extractors [12, 11, 68]. This database approach would be very useful in a constrained environment, as a system for Sign Language detection, where the amount of poses to recognize is small. In a general pose estimator, the database should be very large to extract the wide range of poses a hand would have, with different viewpoints, illuminations, etc. In these systems, the database is usually generated by 3D rendering programs.

A learning approach was used by Lopes in [40, 15, 16] for extracting the hand pose in a general environment. He used a Multilayer Perceptron to link hand appearance with motor representation of the hand (joint angles). Ritter [14, 55, 48] used a also a Neural Network to extract the kinematics of each finger. Neural Networks were also used in [19] to extract joint angles from some fingertip positions provided by magnetic sensors on the fingertips.

The second approach is the most common. Instead the complexity of the hands, good approaches of the output can be guessed. The most used approach is based on temporal tracking. There are different methods to perform this tracking: different kinds of Kalman filtering [34, 31, 64, 16, 49, 61, 64], particle filtering [43, 60] or simple linear predictor [6, 8, 7, 9, 10]. Particle filtering has very good results, but it's very heavy in computing terms. Linear predictor is very light, but insufficient sometimes. Kalman filters are a good compromise between them.

Due to the hand complexity, the third approach is less common. The hand model should be simplified considerably in order to have a closed form solution of the inverse kinematics. On the other hands, the natural hand movements are also very constrained [69]. Systems like [33, 34] are examples of these solutions. These solutions are very useful, since optimization iterations are avoided in this way. If temporal tracking is not possible, the tracking estimation can be replaced by a closed form solution.

Chapter 4

Modelling

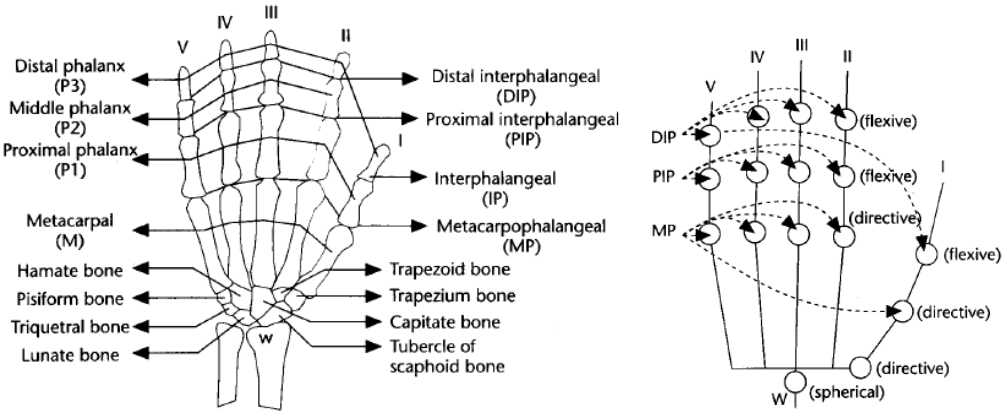
This chapter present different models used in the system. Basically, three models will be explained: the hand model, skin color model and vision model.

4.1 Hand Model

A hand model is represented by a set of internal parameters. Some of these internal parameters are fixed, as the phalanges lengths. Others are variable, as joint angles. Finally there are variable parameters that are usually modeled as fixed, as palm shape. The hand model represent some features (skeleton configuration in a kinematic model, appearance in an appearance model) given the internal parameters. In this thesis our job will be the inverse: given some features, extract the internal parameters of the model. Since we will use a kinematic model, this problem is called inverse kinematics. Hand is a very complex body part shown in Figure 4.1. It has about 30 degrees of freedom and its shape is highly deformable [37]. From the Computer Vision point of view, it has not any texture to be tracked. Finally, the shape and size is very heterogeneous.

All these characteristics make the task of developing a kinematic hand model very difficult. The intended model we are trying to draw should have two main features:

- It has to be as simple as possible. The simpler the model, the simpler and faster the inverse kinematics.
- It has to reflect as well as possible the human hand structure, in order to compare the model to hand images
- If possible, the joint angles should be extracted as a closed form solution



(a) Hand skeleton (b) Hand schema

Figure 4.1: Different representations of a human hands

of an equation whose variable are key points positions (fingertips, wrist, centroid, etc.)

In order to accomplish this task, a complex model will be taken as a starting point, and simplifications will be done in order to fulfill the requirements. The model shown in Figure 4.1(b) is one of the most complete model of the hand used in robotics. All the fingers haves three links. The thumb joints are 2 dof, 2 dof and 1 dof respectively. The rest of the fingers joints are 2, 1 and 1 dof respectively. 6 more dof are added to model the rotation (that would be seen as a spherical joint in the wrist) and position. This makes a total of 27 degrees of freedom.

But the movement of these joints is not autonomous at all. For example, it is almost impossible to move the last link of a finger (change the angle of last joint) without moving the previous link (change the previous joint angle). The tendon that runs through the finger cause this dependency [54]. Different measures reveal that the dependency can be approximated by $\theta_4 = 2/3 * \theta_3$. At this point, we can check the third prerequisite: is it possible to extract joint angles from some key points locations? As a starting point, the second joint angle $\theta 2$ on Metacarpophalangeal joint(the one perpendicular to the rest of joint angles in the finger) can be easily extracted by simple trigonometrics after solving the other angles in Figure 4.2. Since points lie on a plane, we can raise the problem as the following: we should extract angles θ_4 and θ_3 from the distance from fingerbase to fingertip and the link lengths 4.2.

The solution of this problem can be found in [33]. Joint angle θ_3 is the solution of a 5th grade polynomial, and $\theta_4 = 2/3 * \theta_3$. The final value should

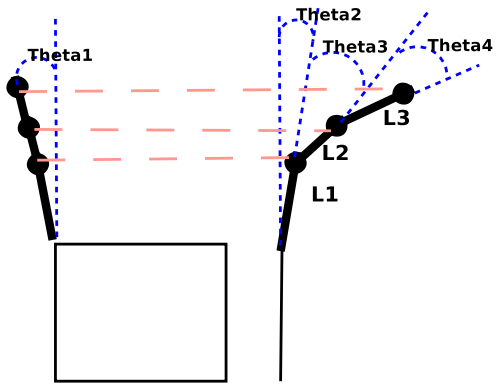


Figure 4.2: Model of finger with coplanar joints

be chosen among the real solutions of the equations system:

$$ax^5 + bx^3 + cx^2 + dx + e; \tag{4.1}$$
$$a = 32l_1l_3; \tag{4.2}$$
$$b = -40l_1l_3 + 8l_1l_2; \tag{4.3}$$
$$c = 4l_2l_3; \tag{4.4}$$
$$d = 10l_1l_3 - 6l1_l2; \tag{4.5}$$
$$e = l_1^2 + l_2^2 + l_3^2 - r^2 - 2l_2l_3; \tag{4.6}$$
$$x = cos(\theta 3/3); \tag{4.7}$$

In general, this polynomial will not have an analytic solution given any set of lengths. But, if we change slightly the constraints, we will obtain an analytic solution for $\theta 3$. Following the nomenclature in Figure 4.3:

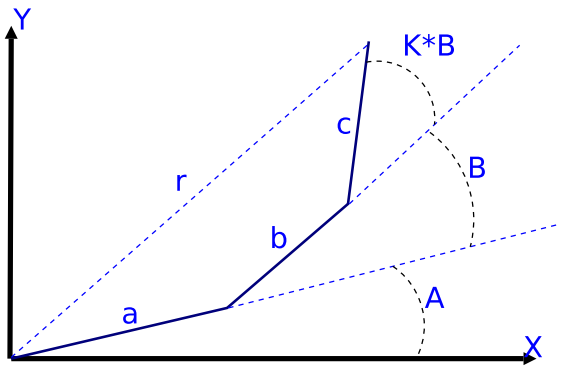


Figure 4.3: Model for general constraint

$$x_{ft} = a * cos(A) + b * cos(A + B) + c * cos(A + B + K * B); \tag{4.8}$$

$$y_{ft} = a * sin(A) + b * sin(A + B) + c * sin(A + B + K * B); \quad (4.9)$$

$$r^2 = y^2 + x^2 = (a * cos(A) + b * cos(A + B) + c * cos(A + B + K * B))^2 + \\ +(a * sin(A) + b * sin(A + B) + c * sin(A + B + K * B))^2; \quad (4.10)$$

$$((r^2 - (a^2 + b^2 + c^2))/2) = a * c * cos((1 + K) * B) + \\ +a * b * cos(B) + b * c * cos(K * B); \quad (4.11)$$

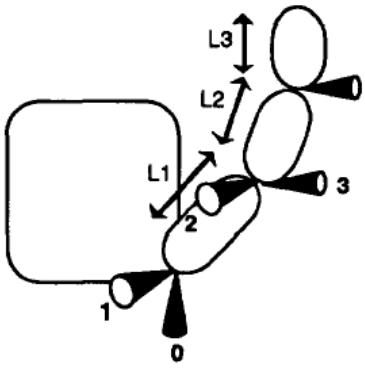
A polynomial is obtained from this equation. The polynomial order is related with the sum of denominator and numerator of the fraction constraint. If we set the constraint $\theta_4 = 1/2 * \theta_3$, we obtain a third order polynomial with 3 closed form solutions (with the constraint $\theta_4 = \theta_3$ we obtain second order polynomial, with solutions

$\left\{B = \pi - \arccos\left(1/4\ \frac{ab+bc\pm\sqrt{b^2a^2-2\,b^2ac+b^2c^2+4\,acr^2-4\,ca^3-4\,c^3a+8\,c^2a^2}}{ac}\right)\right\}$

for simplicity the solution when constraint $\theta_4 = 1/2 * \theta_3$ is not shown).

The drawback of this solution is that the constraint is less realistic than the previous one. On the other hand, this solution does not need neither special computation for each set of lengths nor numeric resolutions of polynomials.

The model of 4 degrees of freedom with one coupled joint is very used for all fingers but thumb. In [33] the thumb is just forgotten and other 4 fingers are modeled. Thumb has a very complex kinematic model [28]. Thumb can be modelled with five links and five non-perpendicular joint angles. The thumb model shown in Figure 4.4 was used in [54]. It has 5 joint angles, but the degrees of freedom are 3 due to the following constraints:



	θ	a	d	α
1	0	0	0	90
2	90	l_1	0	0
3	0	0	0	90
4	0	l_2	0	0
5	0	l_3	0	0

Figure 4.4: Thumb model

$$q3 = 2 * (q2 - \pi/6); \quad (4.12)$$

$$q5 = 7/5 * q4; \quad (4.13)$$

The drawback of this model is that a closed form solution were not found. But, for many applications , different constraints can be applied:

$$q5 = q4; \tag{4.14}$$

$$q3 = CONSTANT; \tag{4.15}$$

Additionally, some changes can be introduced in the axis orientation, in order to apply the fingertip model to the thumb also. The first assumption 4.14 is just an approximation of the second assumption in Rijpkema model [54](Figure 4.13) that makes suitable the closed form solution. The second just removes this possibility of movement. When the metacarpal is kept static, the finger movement is planar, so this constraint is not too far from reality. The model can be seen in Figure 4.5. The axis orientation should be chosen carefully since a degree of freedom was removed, in order to allow natural movements. For this reason, the orientation of axis of joint 0 is parallel to the line between thumb base and opposite corner of the palm.

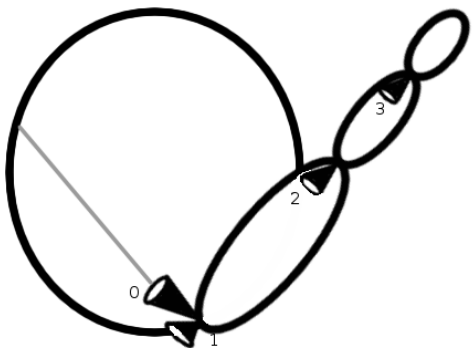


Figure 4.5: Simplified thumb model with closed form solution

4.2 Hand Model for simple matching

Although the previous model is very useful to provide the hand pose from few points locations, it is not the best for lower level recognition tasks as:

- Temporal matching: Does the current hand pose correspond to the one in the previous frame?

- Stereo hand matching: Does the current hand pose correspond to the one in the other camera?

In Figure 4.6 we can see both types of matching. For these purpose, there is is no need for a model which appearance can be directly related with hand appearance. The simple elliptic model shown in Figure 4.6 it is enough. This model has five parameters: x and y centroid position, principal and secondary axis lengths and orientation of axis. This parameters can be easily extracted from the second order momentums of skin color pixels.

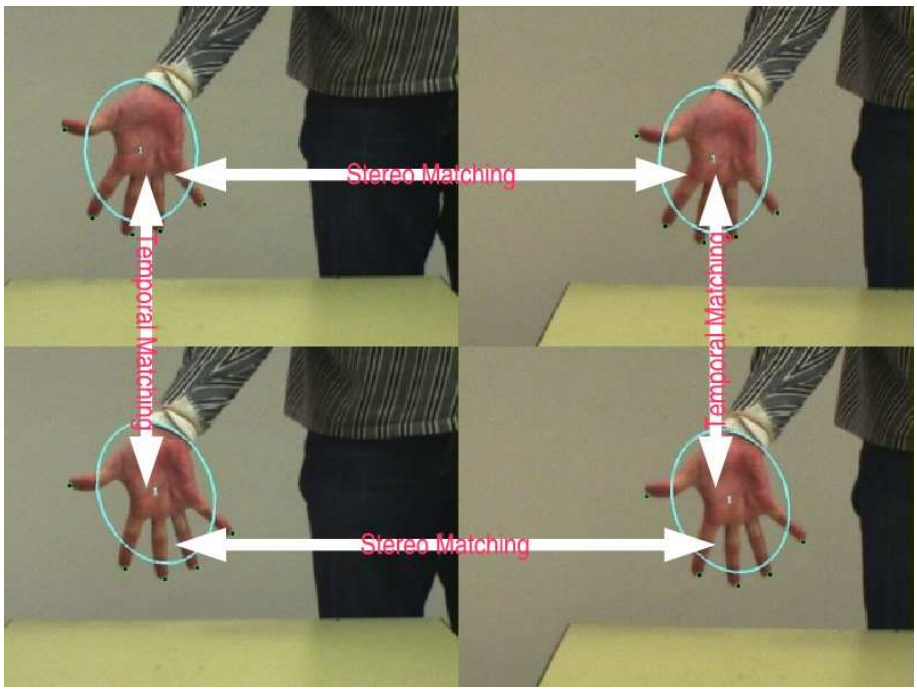


Figure 4.6: Temporal and Stereo Hand Matching

4.3 Skin color Model

The color model used for the segmentation was taken directly from the system proposed in [6, 8, 7, 9, 10]. A crucial part of a color segmentation is the color model that is going to be segmented. There are two main approaches about the color model:

- Parametric Model. The color is described by the parameters of a probabilistic density function.

- Non-parametric Model. The color is described by histograms of colors in training data.

For the system, a non-parametric model was chosen. Instead the parametric model are easier to be described, usually they are very simple to model a complex object color like human skin. Parametric models are usually based on simple Gaussian or mixture of Gaussians. We can see in [17] that the mixture of Gaussians models gets better performance in skin segmentation than the single Gaussian one. Performance of a single Gaussian model is usually insufficient, while complexity of mixtures of Gaussians increases fast. In mixtures of Gaussians models the problem of choosing the number of kernels is also difficult to solve in a dynamic environment like the system we are building.

Non-parametric models gives better performance and lower cost than mixture of Gaussians, furthermore when the amount of training data is high [36]. Since training data can be obtained off-line, it is preferable to have the best model although it needs more training data.

Since illumination variations change drastically the color segmentation performance, an adaptation procedure was developed in [6, 7] to take into account the actual illumination. There are two thresholds in the system, T_{min} and T_{max}. Based on Bayesian rule, if a pixel has a probability $P(s/c) > T_{max}$, it is automatically labelled as skin. If $T_{max} > P(s/c) > T_{min}$, the pixel is label as skin if it is a neighbor of a skin pixel of the previous type. After labeling completely an image, the histogram of this image is computed and added to a set of temporal probabilities, based on the last frames. When applying the Bayes rule, the final probability is a weighted average of the probabilities based on off-line and online set.

4.4 Vision Model

As we said before, the extraction of 3D information in the system is almost indispensable. In a stereo vision system, the extraction can be done without any constraint about sizes, or shapes. Basically, the depth computation is done in two main steps:

1. Find the location of the same point in left and right images.

2. Compute the depth of this point.

While the second step is only geometrical and easy to do, the first one is more complex.

From the calibration step performed in the setup (5.1), we obtained both intrinsic and extrinsic parameters of the camera. With this information, an epipolar geometry model of the stereo system can be built. This model provides constraints about the location of coupled points, decreasing in this way the number of possible candidates during a search. The relation between coupled points in world coordinates is given by the essential matrix, which can be computed directly from extrinsic camera parameters. To translate world coordinates to camera coordinates in pixels, intrinsic parameters are used. The "translation" of essential matrix to pixel coordinates is the well known fundamental matrix. The reader can see this mathematical development below:

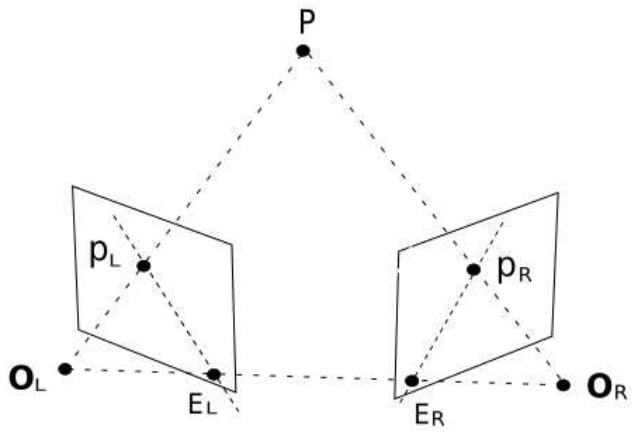


Figure 4.7: Epipolar geometry model

From coplanarity of the 3D point in left coordinates P_L, P_L displaced to right frame $(P_L - t)$ and translation vector t from extrinsic parameters, we obtain: Coplanarity:

$$a \cdot (b \times c) = 0; \tag{4.16}$$

$$(P_L - t)^{\mathrm{T}} \cdot t \times P_L = 0 \Leftrightarrow (R^{\mathrm{T}} P_R)^{\mathrm{T}} \cdot t \times P_L = 0; \tag{4.17}$$

This cross product can be written as a matrix product

$$P_R^{\mathrm{T}} E P_L = 0; \tag{4.18}$$

$$E = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix}; \tag{4.19}$$

It is also applicable to the projections of 3D points p_L, p_R into 2D,

$$p_R^{\mathrm{T}} E p_L = 0 \tag{4.20}$$

$$(4.21)$$

To translate into camera coordinates, intrinsic parameters matrices K_L, K_R are used:

$$\overline{p}_L = K_L p_L; \tag{4.22}$$
$$\overline{p}_R = K_R p_R; \tag{4.23}$$

So, finally:

$$\overline{p}_R^{\mathrm{T}} K_R^{-\mathrm{T}} E K_L^{-1} \overline{p}_L = 0; \tag{4.24}$$
$$\overline{p}_R^{\mathrm{T}} F \overline{p}_L = 0; \tag{4.25}$$
$$F = K_R^{-\mathrm{T}} E K_L^{-1}; \tag{4.26}$$

Epipolar lines are written as

$$\overline{u}_R = F\overline{p}_L \tag{4.27}$$

With this model, it is known that the point $\overline{p}_R$ which corresponds to point $\overline{p}_L$in left camera lies on the line

$$\overline{u}_R = F\overline{p}_L; \tag{4.28}$$

For this reason, the distance from a point to an epipolar line can be used as a similarity measure between points in different cameras. This measure should be completed with more information like tracking, 3D extraction or color in order to choose among the different points lying in the epipolar line.

In a typical scenario, hundreds or thousand of points should be matched. For this reason, a hierarchical matching procedure is desirable in order to decrease the computing effort needed to match all these points. A possible division would be the following:

1. Match the hands in stereo

2. Given hands correspondences, match contour points that belong to the same hand in different cameras.

The first problem is solved with the simple model explained in 4.2. The "cost" that it is minimized when performing the matching is the distance between centroid and epipolar line corresponding to centroid in the other camera. Since it is not a high reliable measure (if two centroids are in the same epipolar line, the matching is random), this stereo matching is only done when a new hand appears for the first time. After this match, the

identities will be tracked in monocular temporal tracking 5.2, which is more reliable than Stereo Matching.

The second step is more difficult. The amount of points to be matched is much higher, and usually more than one point lie on the same epipolar line. In this thesis three different approaches has been taken into account:

- Compute a motion that approximates the hand contour in one image to the contour in the other one. After that, optimization methods such as Iterative Closest Point (ICP [57, 71]) would improve the results. This approach is taken from [9].
- Assume that there are characteristics (curvature, color, etc.) which allow us to know a priori the correspondence between points.
- Match a starting point and perform a sequential correspondence search based on minimizing the sum of the matching costs.

The first approach takes the advantage of the model in 4.2 to make a fast first approximation for the point correspondences. One hand is scaled, and the centroid, principal and secondary axis are aligned. All these parameters are taken from the elliptic model in 4.2. After this alignment process, a matrix of distances between all the points is computed. These distances are simple Euclidean distances. For a given left camera point, the first right camera point whose square distance is lower than a criterion (i.e. 90% of square distances) is selected as the match. This procedure is not optimal, but it is fast. If the quality of matches is still not good enough, an optimization method based on robust ICP variant improves the matching. The ICP optimization tries to find the best affine transformation which transforms one hand contour to the other. Affine transformations preserve collinearity and ratios of distances; basically is a composition of rotations, translations, dilations, and shears [1]. This conditions are not really kept if points of the contour do not lie on the same plane, and in this conditions the optimization would fail (see Figure 4.8). For these reasons, other approaches where taken into account.

The second approach tries to improve performance when contours do not fit the affine transformation. If we would know a priori the point correspondences for interesting points, the computing effort would be much lower. Unfortunately, there is not any feature that can characterize clearly the points. When fingertips are not occluding the palm, the curvature of contour has usually a maximum on them. Unluckily, this measure is very sensible to the noise, and would change a lot when the viewpoints are very different. In order to make it work, it can be mixed with shape or color correlation.

Figure 4.8: Example of non-Affine transformation between left and right images

The third approach also tries to solve the problem of affine transformation. Let us think about hand contours just as two uni-dimensional sequences of points $r[i], l[j]$. We will assume that we can match one point with considerably good accuracy. Considering this point as an origin $r[0], l[0]$, the "marriage" of points can be done in a increasing order of index. If we consider the last match $r[n], l[m]$ (i.e. the first one, starting point), we can do three actions for the next match:

- Match the following 2 points : $r[n+1], l[m+1]$
- Match the following right point with the current left point (shift one point in right image): $r[n+1], l[m]$
- Match the current right point with the following left point (shift one point in left image): $r[n], l[m+1]$

The first action should be accomplished when the following points are similar enough to be matched. The other actions are performed when some misalignment has occurred. These misalignments are mainly caused by two reasons:

- The starting point selection may be imperfect. Misalignments can be solved by repeating the 2^{nd} or 3^{rd} actions until a good starting point is reached.
- The sequences have different "speeds", since the number of border points is not the same (see Figure 4.9).

The election of the best set of actions (match, shift right, shift left) in each iteration returns finally the correspondences between the points. When

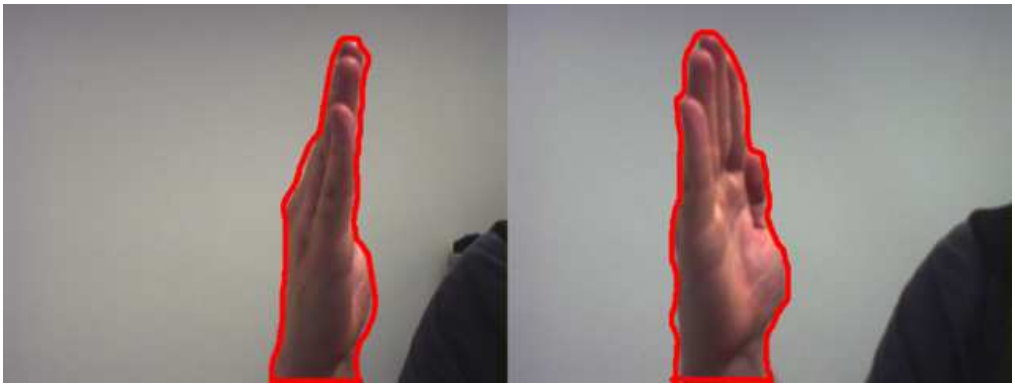


Figure 4.9: Stereo images with different amount of border points

a point is matched to more than one (shifted ones) in the other image, the middle point is chosen as the correspondence. More details about the implementation will be given in section 5.4

The last problem in 3D reconstruction is still remaining: we should extract 3D position given the intrinsic and extrinsic parameters of stereo cameras and point location in right and left camera. This would be done quite easily by calculating the intersection in 3D of lines from the optic center.

Chapter 5

System implementation

In this chapter, the final implementation of the system will be explained in a more detailed way. Also an explanation about the system setup is given here. A flow diagram of the system can be seen in Figure 5.1

In Figure 5.1 we can see 5 levels or varieties of modules. Some of them were mainly solved by other projects, and here are briefly referred to, while others were solved or planned here and explained in more detail.

5.1 System Setup

Some time of this master thesis was spent setting up the system where the hand pose estimator will run. Basically the setup can be divided into hardware and software setup.

Hardware Setup

The hardware used in this system is basically a stereo head with two firewire cameras and a computer. The stereo head model is a Yorick Platform [4]. The head has 4 degrees of freedom. It was not designed to be equipped with the firewire cameras, so an adaptation was done to keep them stable. Due to this adaptation, cameras alignment is not perfect. In order to get a good 3D reconstruction of the scene, the cameras alignment should be as good as possible or the misalignment should be known precisely.

In order to know exactly the parameters of the camera, both extrinsic and intrinsic, a matlab calibration toolbox was used [65]. This tool is easy to use and provide very accurate results. For the future, an specific tool for Yorick head, that use the four degrees of freedom in order to calibrate and align the head would be very useful. In the actual configuration, the head is kept static and the parameters are constant.

The cameras are two Allied Vision Marlin F-080C [3]. The maximum resolution is 1024x728, but a lower resolution is used for various reasons. First, the maximum resolution does not work at high frame rates, due to the bandwidth of firewire card. Second, since the algorithm perform a color segmentation and other pixel by pixel image process, the time of processing the images increase with the resolution. A resolution of 800x600 at a frame rate of 30 msec was chosen for this system. The cameras are fully-configurable. There are important parameters to be checked:

- White balance: it should be set to manual; if not, the illumination will be unstable.
- Shutter: it should be set as low (fast) as possible. The faster, the less motion blur will be in the image
- Auto Exposure and Gain: should be as high as possible in order to compensate the darkness due to the fast shutter.

Motion blur should be avoided when possible. Color segmentation and fingertip extraction are features highly sensible to motion blur. We can see in Figure 5.2the differences between a fast 5.2(a) and a slow 5.2(b) shutter. Camera synchronization was another problem to deal with while working with sequences. Due to frame dropping, the sequences were not synchronized in the end, and the matches between left and right cameras were done between frames delayed more than half a second. In order to avoid this, a program was written to drop frames in the camera less delayed until the time difference between frames was less than a threshold. In this way, we avoid to match delayed frames, but some abrupt jumps occur during the sequences. The computer is an AMD Opteron Dual Core running at 2x2.6 GHz. The operative system is GNU/Linux

Software Setup

In order to begin to work, some software were installed and adapted. In the beginning, the following software pieces had to be connected:

- A firewire camera grabber (based on libdc1394 package example grabcolor [2])
- Tracker based on Fast Hand Tracker proposed in [6, 8, 7, 9, 10]
- A program for visualizing images in linux
- Robotic Grasping simulator GraspIt [44]

The connection between the firewire grabber and the tracker was done by shared memory. This implementation allows any program or any number of trackers to use the frames that are being grabbed from the cameras. This would be very useful during the integration of a hand tracker and one (or many) object trackers. The program for visualizing on-line images from the tracker is a simple code in C++ that use directly Xlib. More complex programs were not taken into account since the main purpose was simplicity and speed. A C++ interface called graspif was used to connect the tracker to GraspIt. This way is cleaner than the usual socket control of GraspIt. The last two point were solved by C++ code. Since the original tracker was written in C, some adaptations were done in the tracker to be compiled by g++ compiler.

5.2 Hand Detector

The module of hand detection is taken directly from the system proposed in [6, 8, 7, 9, 10]. As it can be seen in the Figure 5.1, it is basically divided into three components:

- Color segmentation.
- Hypothesis generation.
- Temporal hypothesis matching.

Color segmentation is based on the model explained in 4.3. When the image is segmentated, hands should be modeled as described in 4.2; this is called hypothesis generation. But the generation of hypothesis from pixels is only needed at the beginning or when a new hypothesis (new skin pixels blob) appear in the image; usually pixels are associated to predicted hypothesis from the previous frame. This is the procedure for performing temporal matching. The hypothesis predictor used here is just a simple linear predictor. The performance is good when the assumption of linear motion is kept. But when this assumption is broke, temporal matching fails 5.3. This can be improved with more complex predictors, but the improvement in performance depends highly in the motion model, that is difficult to build. Another solution can be integrate 3D information feedback extracted in the output in order to solve occlusions.

5.3 Fingertip Detector

The main feature extracted in our system will be the fingertips. One reason for choosing fingertips is that, for gestures we are interested in, the position of the fingertips (together with another point position on the palm, to determine palm orientation) usually determine univocally the full hand pose (the reader can realize it by himself trying to change hand pose while keeping fingertips and a point in the palm static).

The positions of fingertips reduce the huge dimension of a hand image (three real values per pixel) to 3D position of 6 points. This is a drastic reduction of the space, which makes much easier the extraction of the pose (if the space of fingertips position is really complete enough to determine univocally the pose). Another reason for choosing fingertips positions is that they are directly related with a kinematic model of the hand; as we saw in 4.1, a closed form solution of finger joint angles can be extracted known the locations of fingertips, fingerbases locations and palm orientation.

Fingertips have some features which make possible to search for them in an image. Some of them are the following:

- Contour shape: The shape of a fingertip contour is semicircular, from almost any point of view. This is one of the main characteristics.
- Silhouette curvature: Since the contour shape is semicircular, the curvature of a fingertip is higher than in any other point in the hand.
- Colour: The nail colour is different than hand skin colour. Although this difference is not too big, it can be used for discarding false positives.

The easiest feature to search for are curvature maxima in the silhouette. It is light in computing terms and reliable in most the situation when fingers are not occluding the palm 5.4(a). Also errors can appear when fingertips touch each other, as seen in Figure 5.4(b).

These failures can be solved if curvature maxima have been searched for in hand contour from an edge map. The problem of edge maps is that these are very noisy, and a lot of points with high curvature can appear in them. In this scenario it is better to search for fingertip shapes, since there are not too many semicircular shapes in an edge map from the hand. This search can be done by a hough circular transform recursively for different fingertip sizes. In the other hand, this procedure is much more expensive computationally. Finally, color can be useful in order to improve fingertip detection in both of the methods described before (if nails are visible). A fingertip extractor based on Hough transform is being developed with good initial results, but

it is still unfinished. For this reason, fingertip extractor proposed in [10] was used. It is based on silhouette curvature measuring.

The silhouette is extracted from the color segmentation by just checking the amount of skin pixels in the neighborhood. The curvature is computed between the lines from different equidistant points to the current point. This is done in various scales (different distances). In order to be a fingertip candidate, a point should be a curvature local maxima and have a curvature above a threshold. From these candidates, the best 5 are selected as fingertips. In Figure 5.5 there are 4 points selected as fingertips (marked as dark squares), since they are curvature local maxima and their curvature is above the threshold. The points marked with a dark circumference are local maxima, but they are below the threshold. One of these points is a local maxima due to the noise, but the other one is a real fingertip with low curvature since pinky finger it occludes partially the pinky finger. The middle finger has also drawn two of the different angles used to compute the curvature. The minimal curvature along the different angles between equidistant points is selected as the curvature measure. If the equidistant points are to close to the reference point, local minima due to noise can become false positives.

5.4 3D Reconstruction

Depth information is very important in the system. The input to this module consist of two stereo images with segmented hands. The contours of the hands are extracted in a separated structure, with the fingertip locations identified. During the first appearance of a hand, the stereo matching of the hand is done following the model in 4.4. After the first appearance, hands are identified by the tracker in the hand detector module.

When hands are already identified, points belonging to each hand contour should be matched to points from the other image. As it was said in 4.4, we considered three ways for matching the hand contour points. The first one was the method used in the original system. The method is simple and fast, but accurate if the constraints are kept. The method consist basically of two steps: a first step when a rough approach is done, and a second one when this approach is improved by iterative optimization. In the first step the contour from the left hand is scaled, translated and rotated in order to be as similar as possible to the right one. This operations are based on the simple elliptic model of the hand used for temporal and stereo hand matching. The principal axis are scaled until they are equal; the right contour centroid is translated to the location of the left one, and the axis are rotated until the ellipses orientation is the same. The elliptic model is sometimes not enough

to model the hands; principal axis can be different in left and right cameras due to their different viewpoint, and the centroid can be different also. The second step was designed for this reason. A new alignment procedure is done by iterative optimization. The optimization method used here is ICP, iterative-closest-point. This method tries to find the best transformation according to a given model between two sets of points. In this case, ICP tries to find the best affine transformation between the two sets of contour points. This process is iterated several times until the quality of the match is good enough (or until the number of iterations exceeds a threshold). In this system, the measure of quality is based on simple square distances between matched contour points.

The drawback of this method is that an affine transformation sometimes can not model the relation between the contour points set (see Figure 4.8). The Affine transformation of one point consist of a linear transformation followed by a translation:

$$x \Rightarrow Ax + b \tag{5.1}$$

This transformation does not preserve angles or lengths between points, but it preserve collinearity and ratios of distance [1]. Basically, if we are considering 2D projections of a 3D environment, this assumptions are kept when sets of points lie on a plane. But contour points of a hand does not keep this constraint in general. This model is also violated when there are objects occluding the hand, and distorting the hand contour. For this reason, other methods were explored.

The second method tries to solve this problem. We are interested only in few points in the contour hand, and these points have specific characteristics which make them easy to be tracked. If we can know the exact position of these points in both images, the matching problem is simplified to match sets of 6 or 7 points that are usually distanced. On the other hand, the location of fingertips should be accurate in order to perform a good estimation of the depth. Since the method used for fingertip detection was not accurate enough, this method was discarded early.

Finally, the third method tries to avoid the affine model of transformation in a different way. The constraints applied to the matching problem are geometric constraints from the epipolar model geometry (more information in 4.4) instead of constraints in the transformation model between stereo images. The method described in Section 4.4 was implemented as a dynamic time warping algorithm [45]. This algorithm solves very well matching problem with misalignments and different "speeds" (see Figure 4.9). Dynamic time warping algorithm is more general than the method described before.

Basically, a cost of each match is computed based on the distance to the previous match and a measure of the match quality itself. The distance between matches is used to favour some type of matches. For example, if the last pair was (n, m) and the intrinsic quality of matches $(n+1, m+1)$ and $(n+2, m+2)$ is the same, we would prefer $(n+1, m+1)$ as the best next match in order to have a larger set of matched points. In order to select this kind of small jumps preferably than the long jumps, the distance from one match to other one would be based on the amount of points between them. In this system jumps of only one point were allowed (it is explained in Section 4.4), and all had the same distance-weight. When shorter jumps were favoured ($(n+1, m)$ and $(n, m+1)$ having less cost than $(n+1, m+1)$) performance was worse. This has sense since the most common action in this system should be "matching" ($(n+1, m+1)$) since contours usually have not misalignments (or small ones) and the number of contour points is usually similar.

There are two important points in this algorithm that have to be fixed in order to have a good performance:

- The intrinsic measure of quality for each match
- The selection of the starting point

The performance of the algorithm depends drastically on these points. If the measure of match quality is not good enough, it is very difficult to get accurate results. One characteristic this quality measure should have is that the matchings should be globally ordered by this quality measure: if pair a is better than pair b, the measure of quality in a should be equal or bigger than in b, never lower. It is desirable also that the local order of quality measure (order in a neighborhood) would be stricter: if pair a is better than pair b, the measure of quality in a should be bigger than in b. Another important characteristic of the measure is monotony: if pair b is the best match after current pair a, the intermediate steps (usually matches that shift points in one image to fix misalignments) should have increasing quality measures while the best match is closer and closer. For example, orientation of the tangent to the contour does not keep this constraint, since noise in the contour extraction imply fast changes in the tangent orientation. If the measure have not this characteristic, the alignment procedure would fail (another pair c would have better quality than the intermediate step with non-increasing quality). This kind of errors would be avoided allowing longer jumps; that is, considering point combinations further than one point $(n+2, m), (n+2, m+1) \cdots$. The computing effort is increased considerably with the number of points considered. We can consider the inverse of the

distance from one point to the epipolar line corresponding to a point in the other image as a measure of quality. This measure orders strictly the points in a neighborhood if contours is non-convex (if it is convex, is not strictly ordered, the equal rule is not kept since an epipolar line can cross twice a contour in the neighborhood). It orders the contour points globally also while points lying in the same epipolar line have the same quality measure.

Unfortunately, this measure does not fulfill the monotonicity characteristic always. This would lead to errors. The best advantage of epipolar lines quality measure is that it does not depend on any model. Coming back to the tangent orientation measure, this measure is only applicable when viewpoints are similar enough to keep angles between points. Epipolar lines quality measure lie only on the position and orientation of the cameras. On the other hand, the dependency on the camera calibration is so strong that any error on this calibration means a considerably loss in matching performance . The epipolar line measure was the measure used in this system.

The other important part in the algorithm is the starting point. Instead of the ability of auto-alignment of this algorithm, if the starting point choice is very bad it is difficult to align the points properly. One reason for this is that the epipolar line orders strictly the points within a non-convex neighborhood; if misalignment is bigger than this neighborhood, another good match (another cross between the epipolar line and the contour) would be found before the correct one. For this reason, it is important to find a good starting point. It is difficult to apply general rules for finding good starting points. For this reason, the system tries random points and evaluate how good they are as starting points. Basically, the system searches for points lying on an epipolar line with two only cross points with the contour line. If there are only two crosses, and they are far enough, it would be easy to choose between them by any of the following methods: orientation of tangent contour (if there are only two crossings, one should have almost the opposite direction of the other one; it is easy to differentiate between them), angle in polar coordinates, see Figure 5.6, etc. If this characteristics are similar between both crossing points, probably the starting point is not good (see Figure 5.7).

The main problem of this approach for choosing the starting point is that it is difficult to know how many times an epipolar line crosses the contour. Since the distance to an epipolar line is a real value, a threshold should be set in order to determinate how many crossings there are. In Figure 5.8 we can see the consequences of a inappropiate threshold.

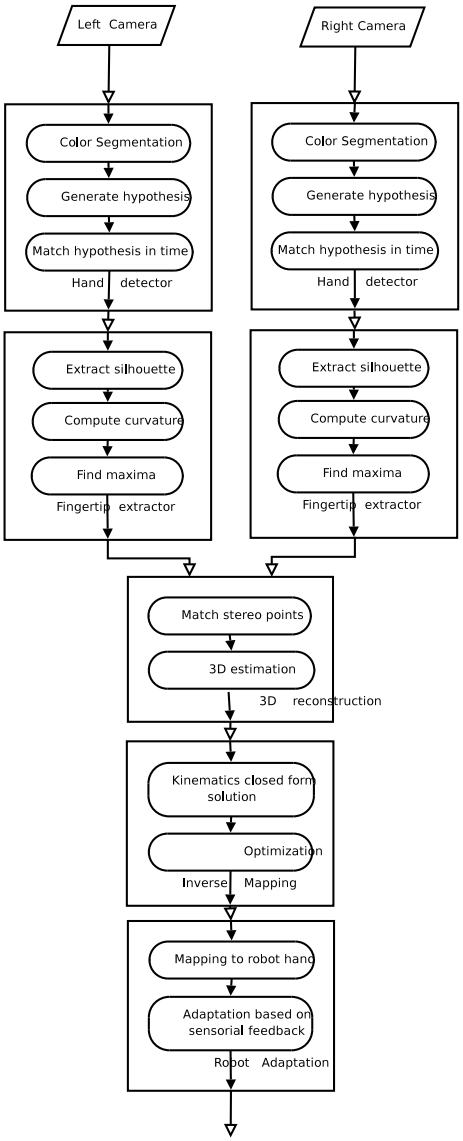


Figure 5.1: System

(a) Fast Shutter (b) Slow Shutter

Figure 5.2: Comparison between fast and slow shutter

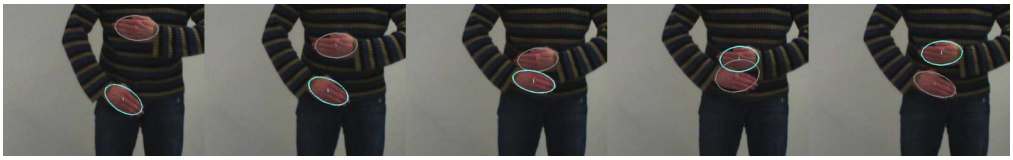
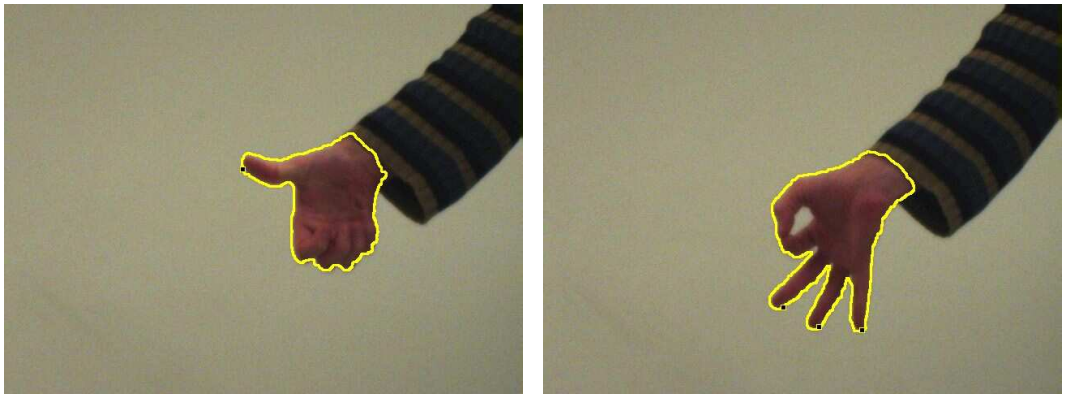


Figure 5.3: Sequence demonstrating a failure of linear predictor



(a) Occlusion Fail (b) Fingertip contact Fail

Figure 5.4: Failures in fingertip detection by curvature measure

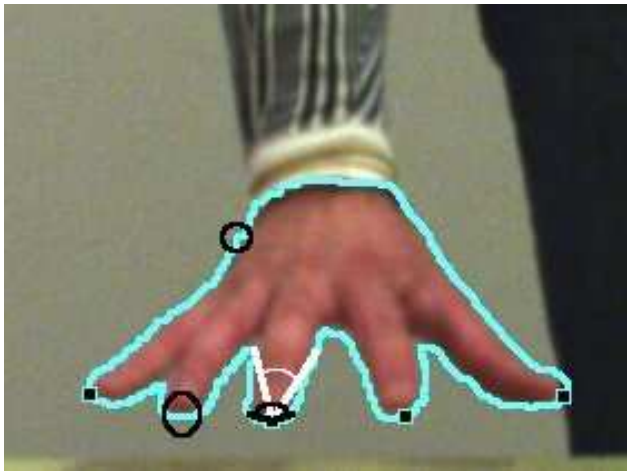


Figure 5.5: Curvature measures for fingertip extraction

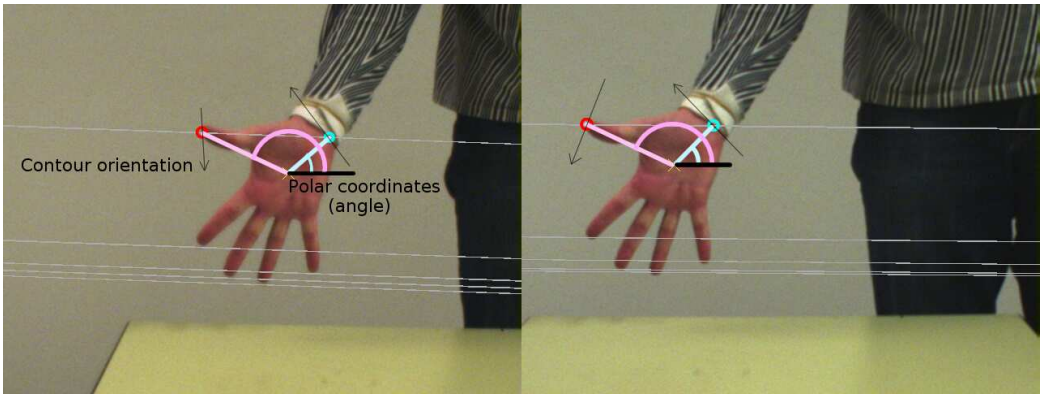


Figure 5.6: Characteristics for rejecting bad starting points: Contour orientation and Polar angle

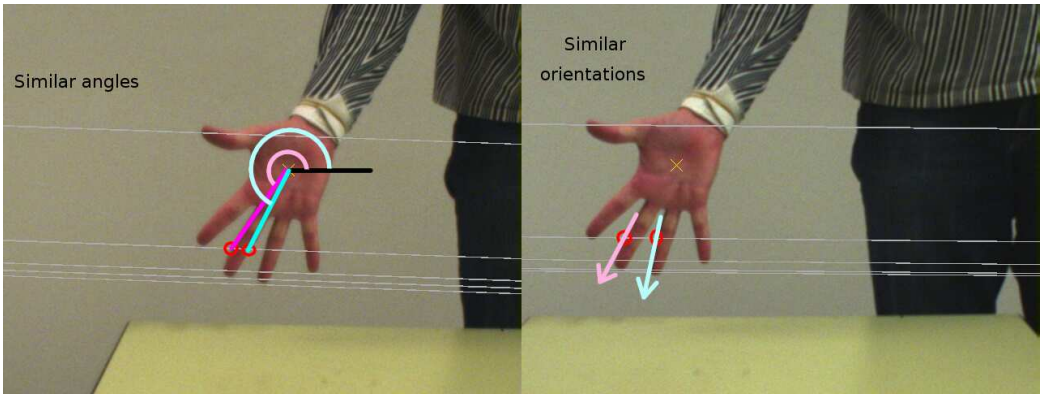


Figure 5.7: Rejected starting points since they have similar characteristics

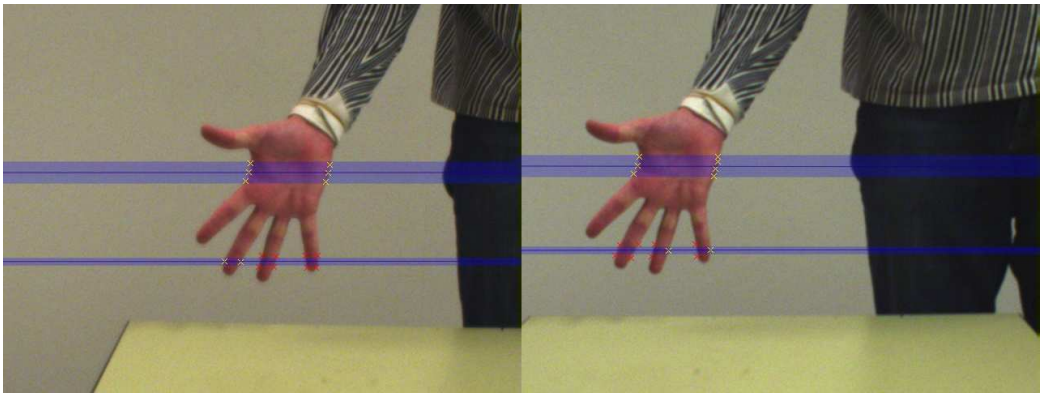


Figure 5.8: Main problem in selecting first point: Threshold selection

Chapter 6

Evaluation

In this chapter the performance of the system will be evaluated. The results about image segmentation and hand contour extraction can be seen in the original publications of the system, [6, 8, 7, 9, 10]. The results that will be related here are about 3D extraction. The inverse kinematics extractor is still in a developmental phase. The ground truth data was generated by manually selecting the position of the fingertips in some pictures, and performing the 3D extraction with the same procedure as it is done in the algorithm.

6.1 Depth estimator

In the evaluation, a set of sequences with different levels of difficulty have been used for testing the algorithms. This set is basically composed of 6 different actions: waving, rotating, pushing, pinch grasp, power grasp, and moving an object. Those actions are executed with and without object (waving is only executed without object), in order to evaluate the effects of object occlusions in the depth estimator. Some samples of each action can be seen in Figure 6.1. All the sequences were recorded in stereo, with a frame rate of 15 frames per second. This section begins with a qualitative evaluation of the two basic implementations of the deph estimator. Basically, three different comparisons were performed: in the original system, depth extraction with and without ICP optimization was compared through different sequences; a comparison between the original system with ICP optimization and depth extraction based on dynamic time warping; finally, a comparison of depth extraction in sequences with and without objects. Also a short comparison between two different time warping algorithms was included. For some of these comparisons, figures with the original picture and the depth extraction computed with ground truth data are provided.

As it was said in Section 5.4, the 3D depth estimation in the original system is based on a rough approach and ICP optimization. The first question presented here is if the optimization is really needed, and how it improves the depth estimation. It can be seen that the depth estimation without ICP optimization is considerably good in simple sequences shown in Figure6.2. Nevertheless, in more complex sequences as power grasp (Figure 6.6), the performance improvement is considerable.

The comparison between ICP and dynamic time warping can be seen in Figures 6.8 and 6.10. In these figures we can see the behaviour of both algorithms during a simple sequence with the fingertips always visible (waving) and during a complex one (power Grasp). The performance of the algorithm ICP is slightly better during the waving sequence, while the DTW achieves a better accuracy in the power Grasp sequences. This is because of the assumptions in the ICP algorithm: this algorithm assumes that the fingertips lie on a plane. This assumption is almost true in the waving sequence, but not in the power grasp sequence.

The last figures try to compare the performance when the extraction is done with and without objects. In general, the performance in sequences without objects is better than in those with objects (see Figure 6.14). But sometimes the fingertips seems "embedded" in the hand due to the object absence. Since the object is usually non-skin colored, it makes the fingertip extraction easier in these cases (see Figures 6.28).

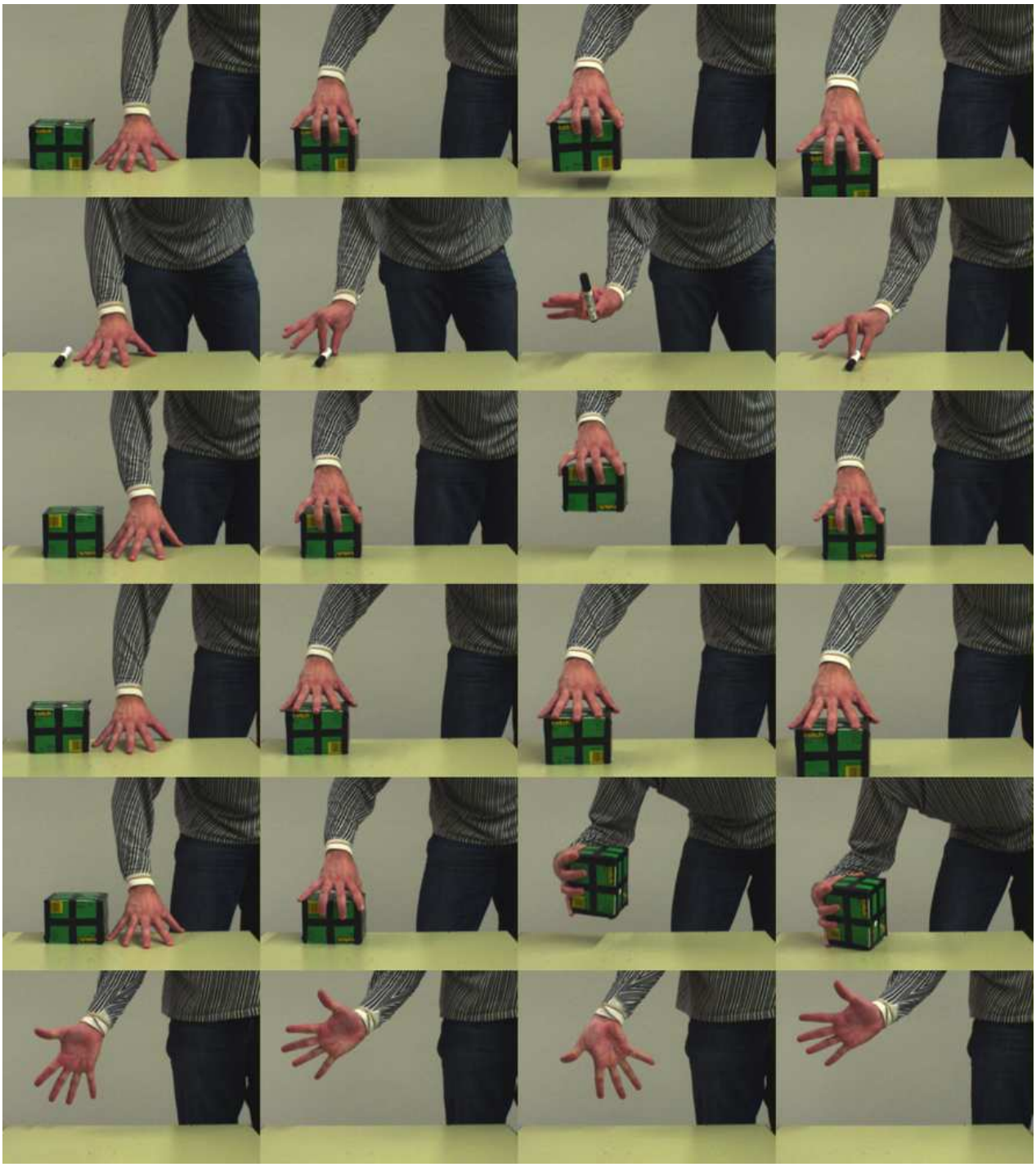
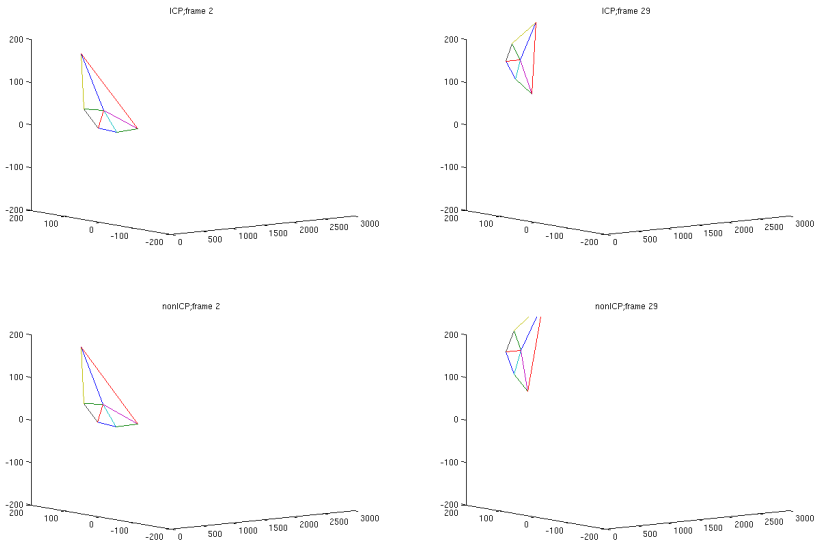
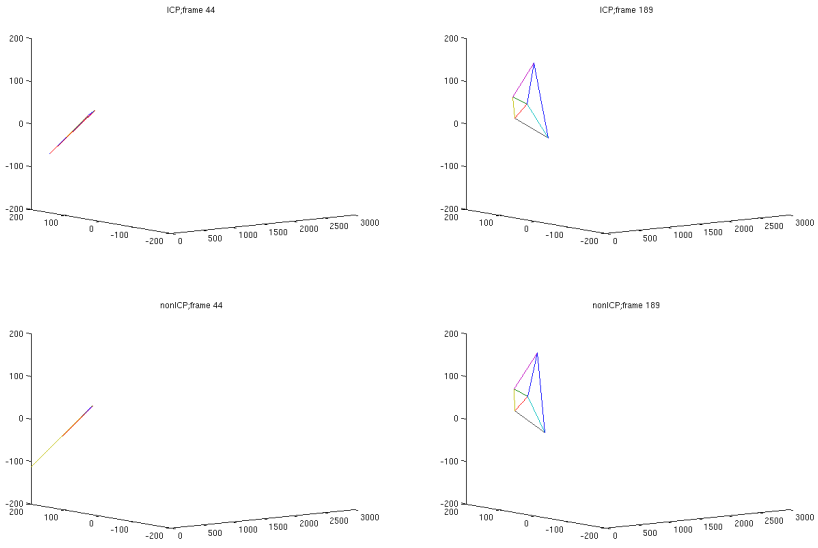


Figure 6.1: Sequences used in the evaluation. From Above: moving, pinch Grasp, power Grasp, pushing, rotating and waving

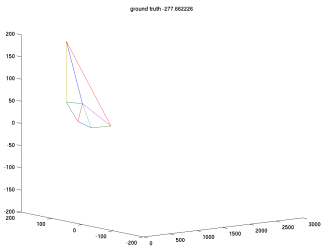


(a) Capture 1 (b) Capture 2

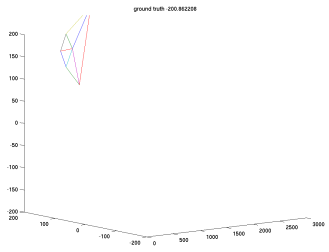


(c) Capture 3 (d) Capture 4

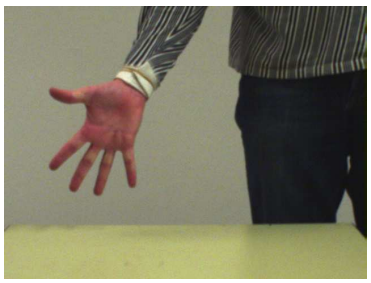
Figure 6.2: Fingertip extraction with and without ICP performing the waving sequence, four captures



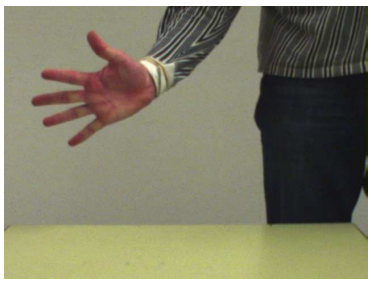
(a) Extraction 1



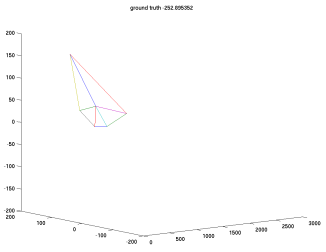
(b) Extraction 2



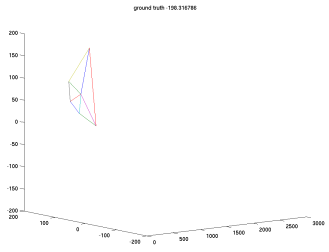
(c) Image 1



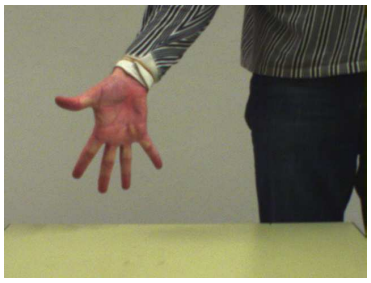
(d) Image 2



(e) Extraction 3



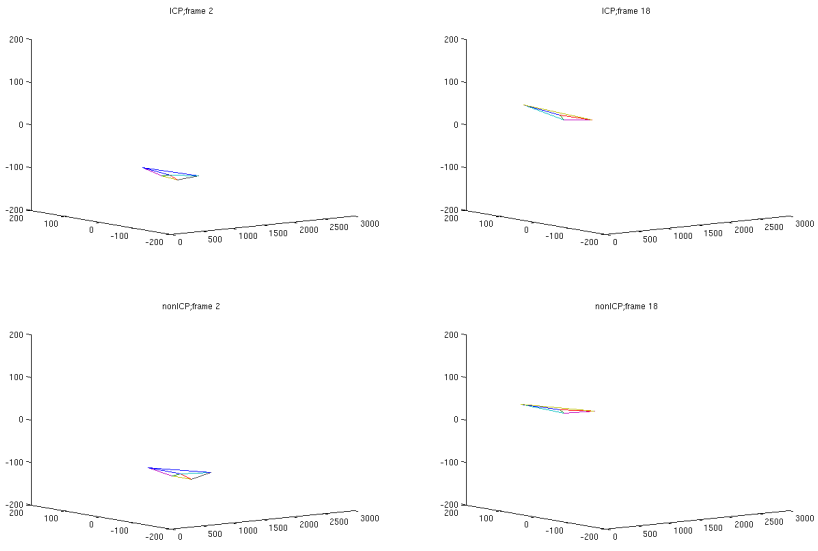
(f) Extraction 4



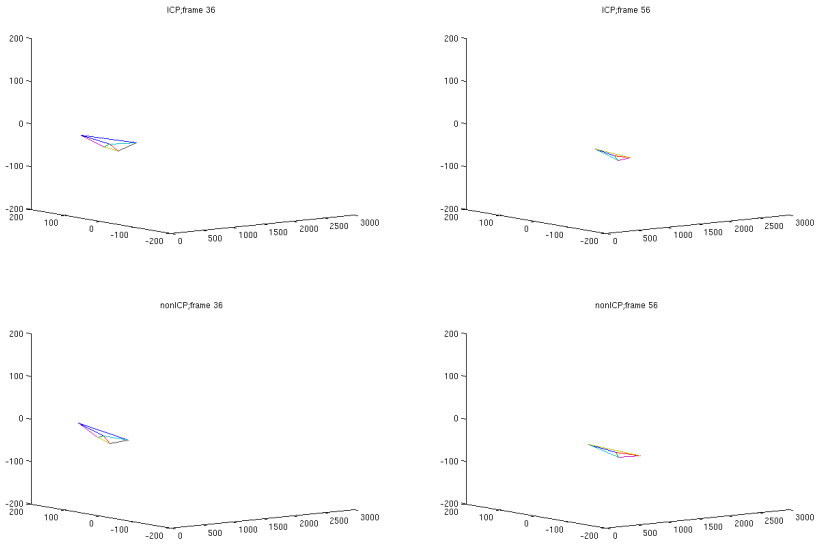
(g) Image 3

(h) Image 4

Figure 6.3: Ground Truth fingertip extraction and images correspondent to the previous Figure (Figure 6.2)

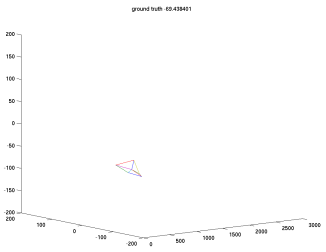


(a) Capture 1 (b) Capture 2

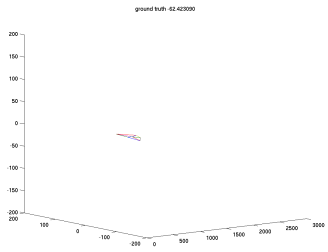


(c) Capture 3 (d) Capture 4

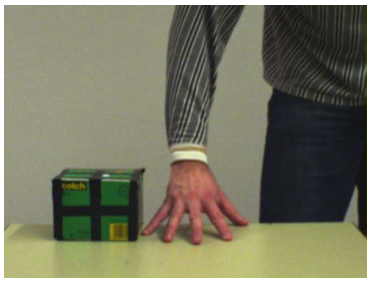
Figure 6.4: Fingertip extraction with and without ICP performing the moving sequence, four captures



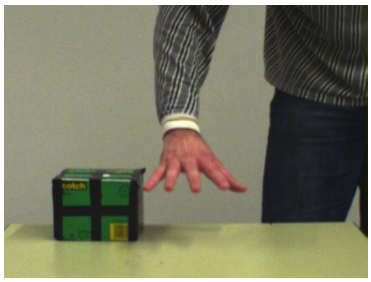
(a) Extraction 1



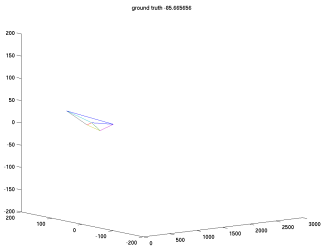
(b) Extraction 2



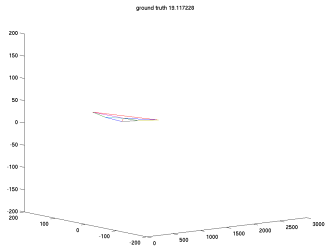
(c) Image 1



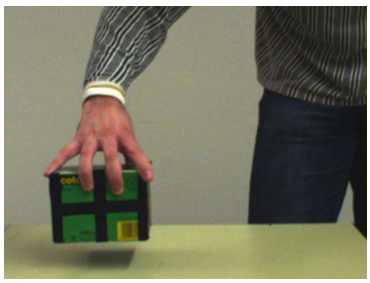
(d) Image 2



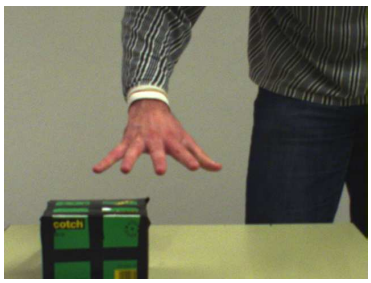
(e) Extraction 3



(f) Extraction 4

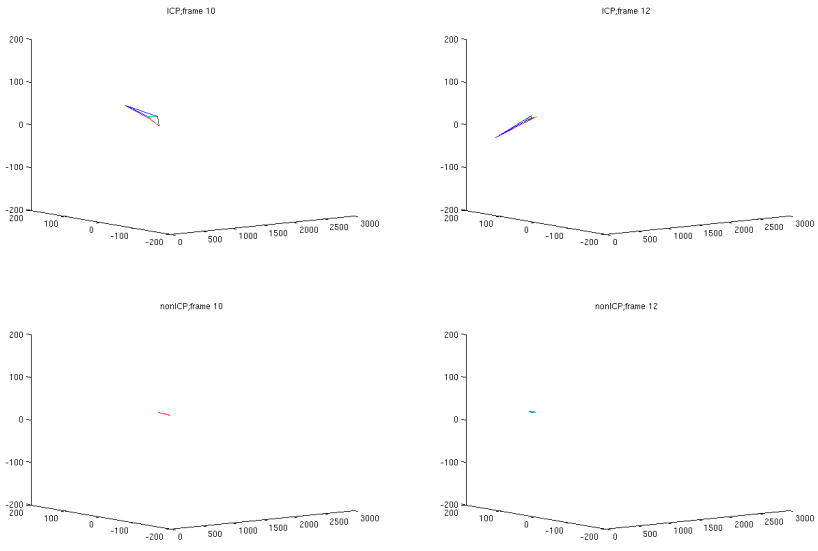


(g) Image 3

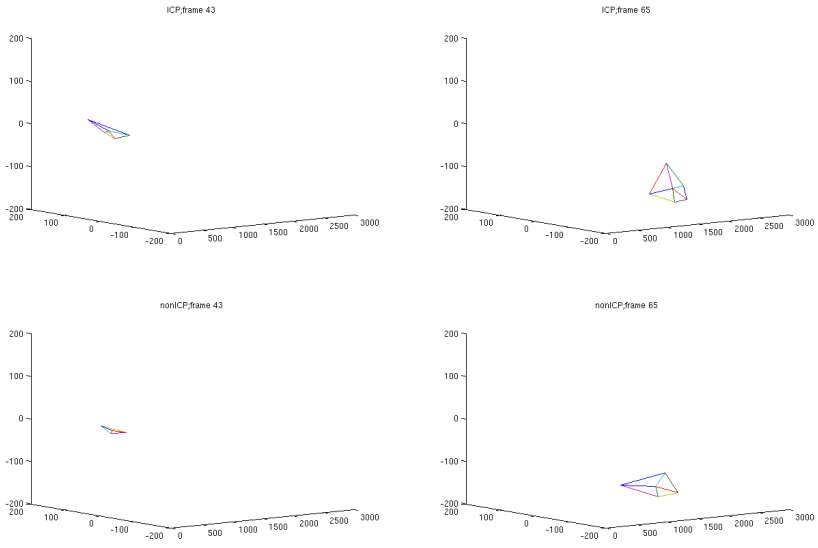


(h) Image 4

Figure 6.5: Ground Truth fingertip extraction and images correspondent to Figure 6.4

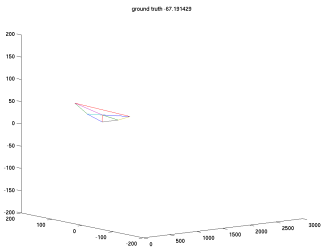


(a) Capture 1 (b) Capture 2

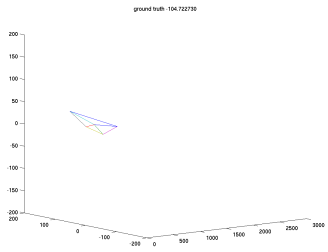


(c) Capture 3 (d) Capture 4

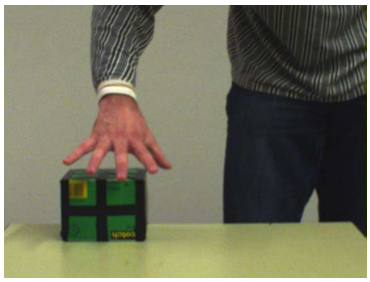
Figure 6.6: Fingertip extraction with and without ICP performing the power Grasp sequence, four captures



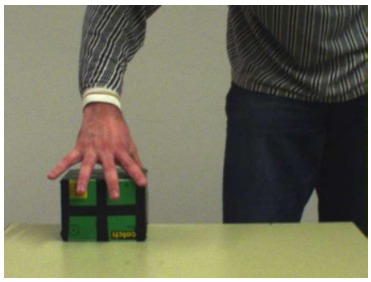
(a) Extraction 1



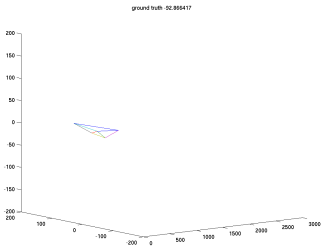
(b) Extraction 2



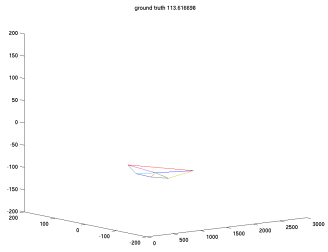
(c) Image 1



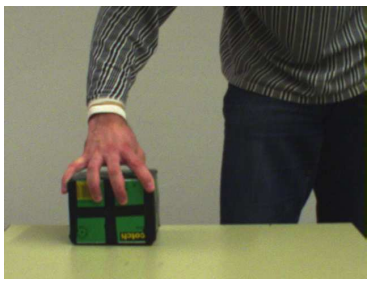
(d) Image 2



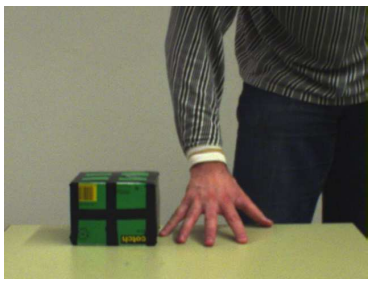
(e) Extraction 3



(f) Extraction 4

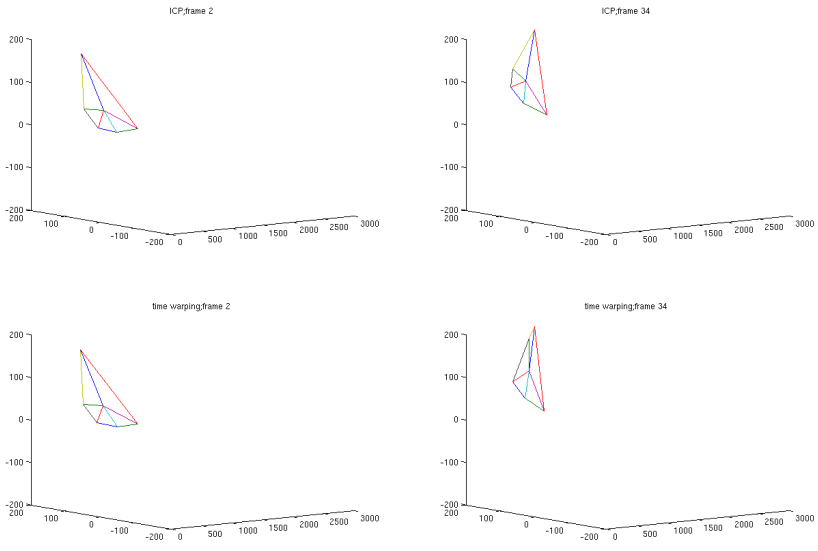


(g) Image 3

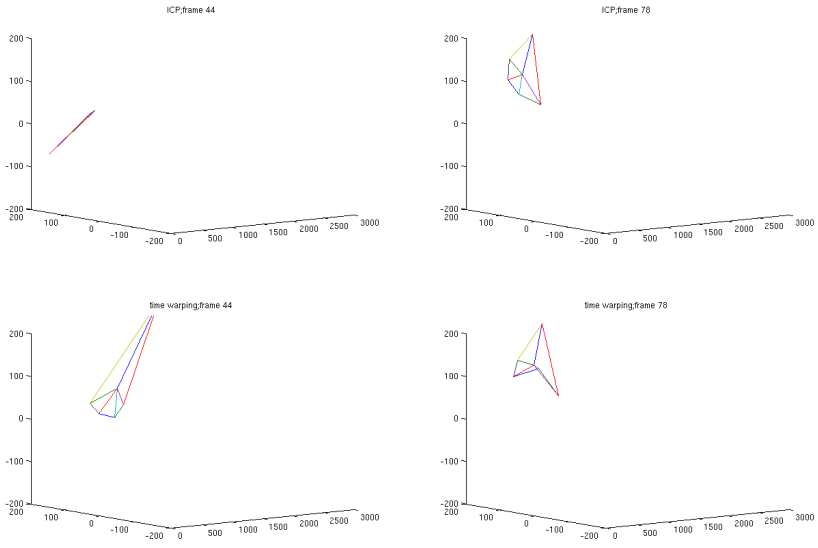


(h) Image 4

Figure 6.7: Ground Truth fingertip extraction and images correspondent to Figure 6.6

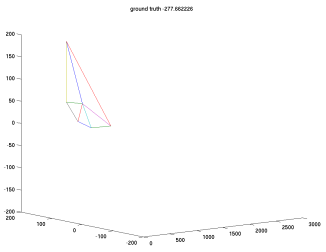


(a) Capture 1 (b) Capture 2

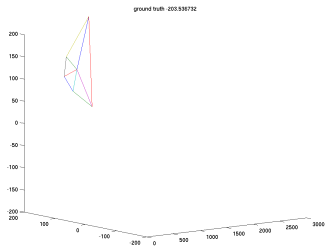


(c) Capture 3 (d) Capture 4

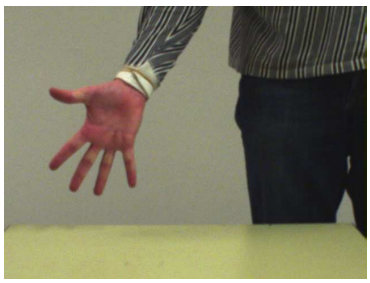
Figure 6.8: Fingertip extraction performing waving sequence with ICP and TW estimators, four captures



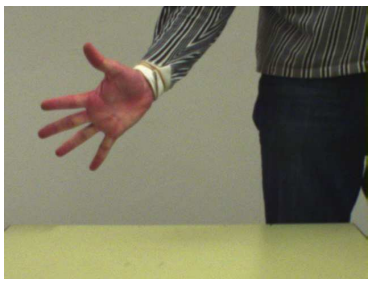
(a) Extraction 1



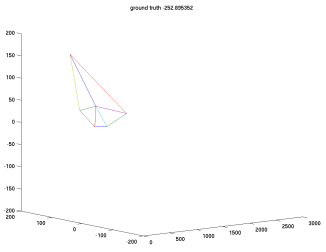
(b) Extraction 2



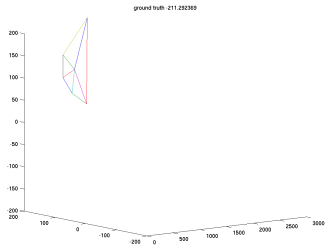
(c) Image 1



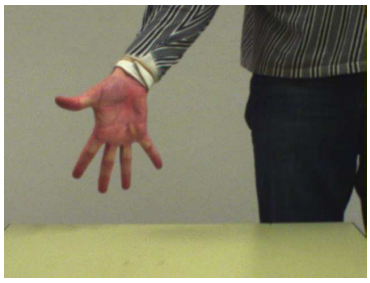
(d) Image 2



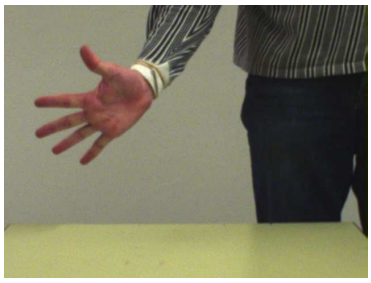
(e) Extraction 3



(f) Extraction 4

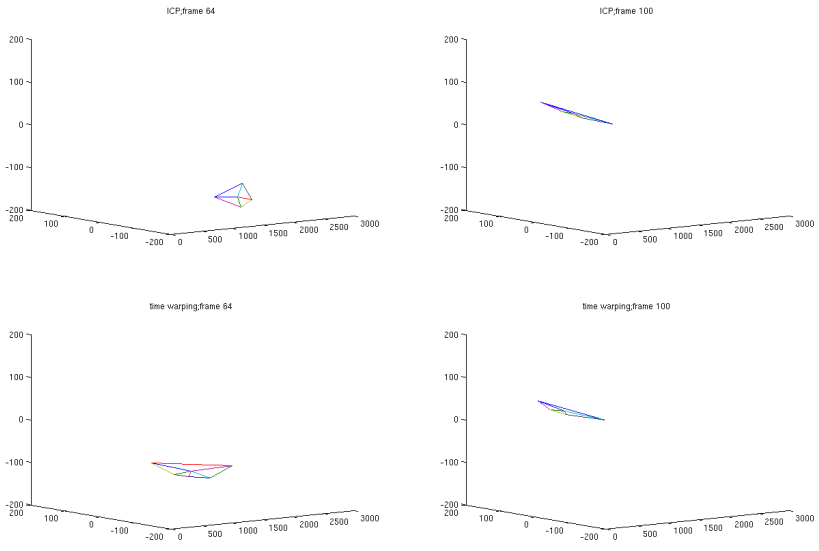


(g) Image 3

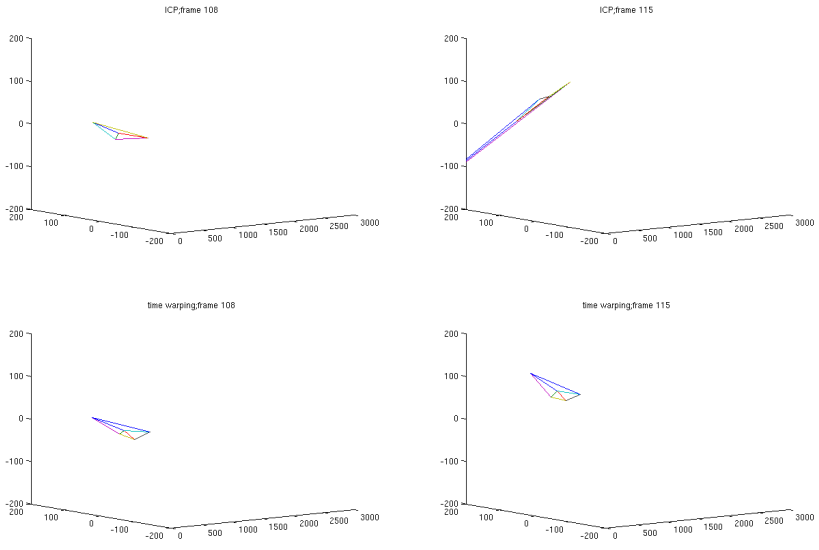


(h) Image 4

Figure 6.9: Ground Truth fingertip extraction and images correspondent to the previous Figure (Figure 6.8)

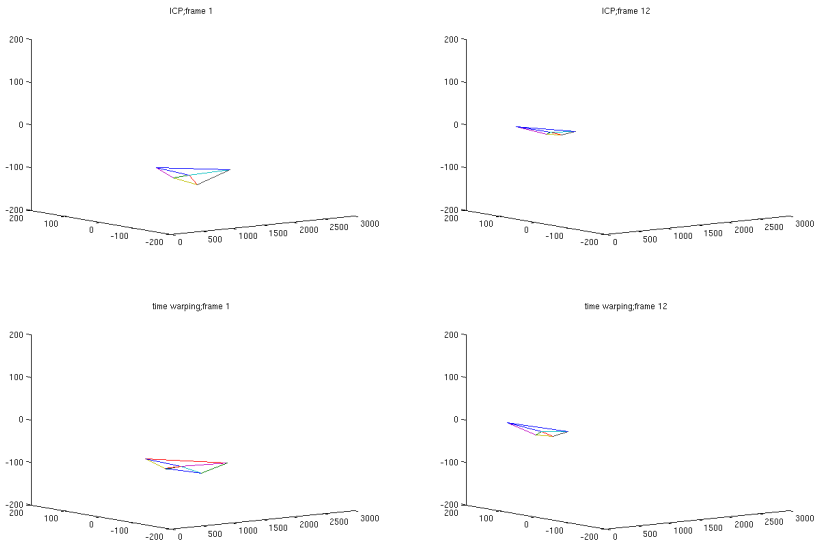


(a) Capture 1 (b) Capture 2

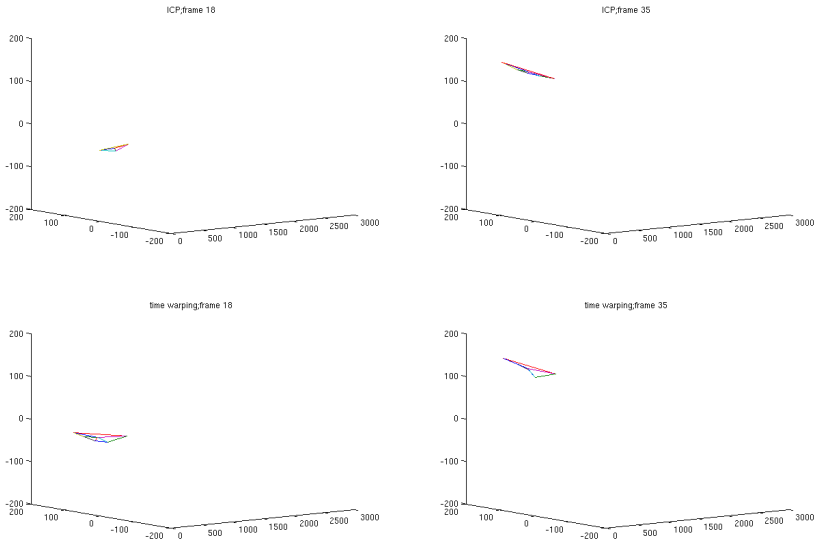


(c) Capture 3 (d) Capture 4

Figure 6.10: Fingertip extraction performing power Grasp sequence with ICP and TW estimators, four captures

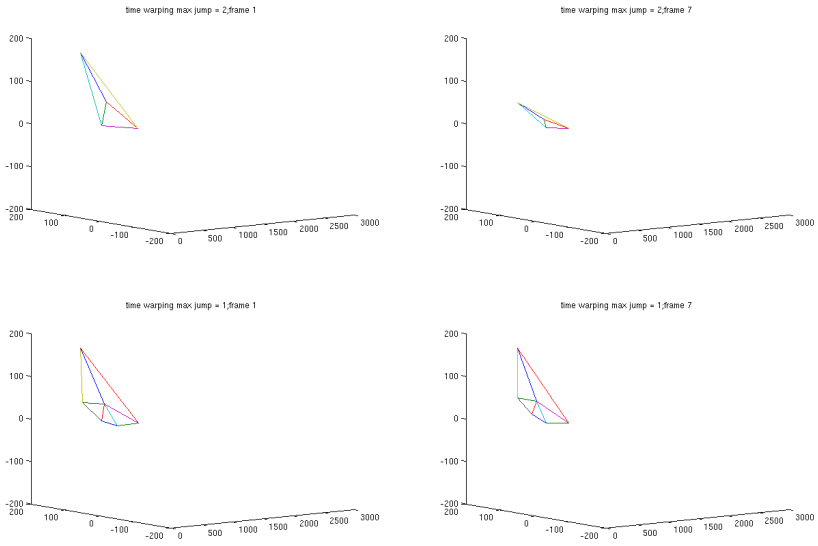


(a) Capture 1 (b) Capture 2



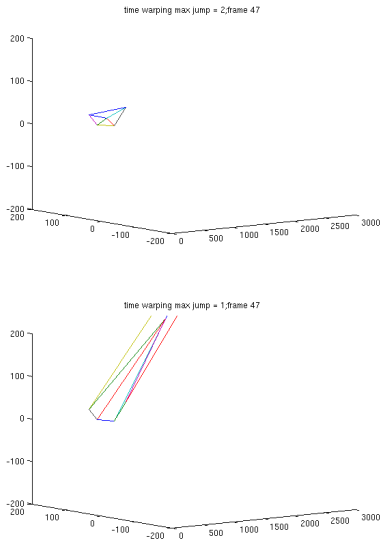
(c) Capture 3 (d) Capture 4

Figure 6.11: Fingertip extraction performing power Grasp sequence with ICP and TW estimators, four captures



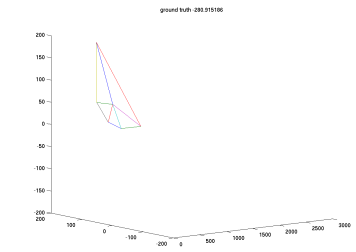
(a) Capture 1

(b) Capture 2

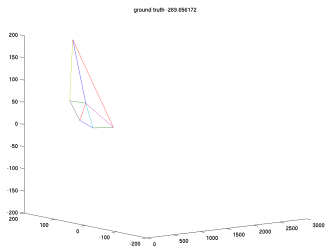


(c) Capture 3

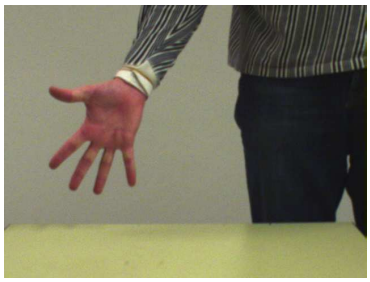
Figure 6.12: Fingertip extraction performing waving sequence with TW1 and TW2 estimators, four captures



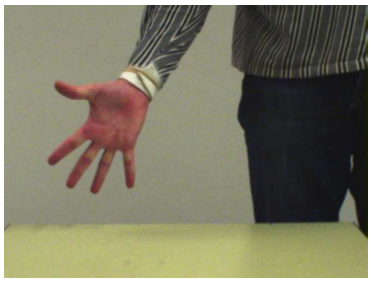
(a) Extraction 1



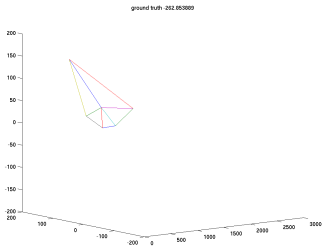
(b) Extraction 2



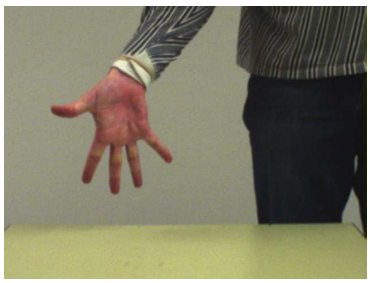
(c) Image 1



(d) Image 2

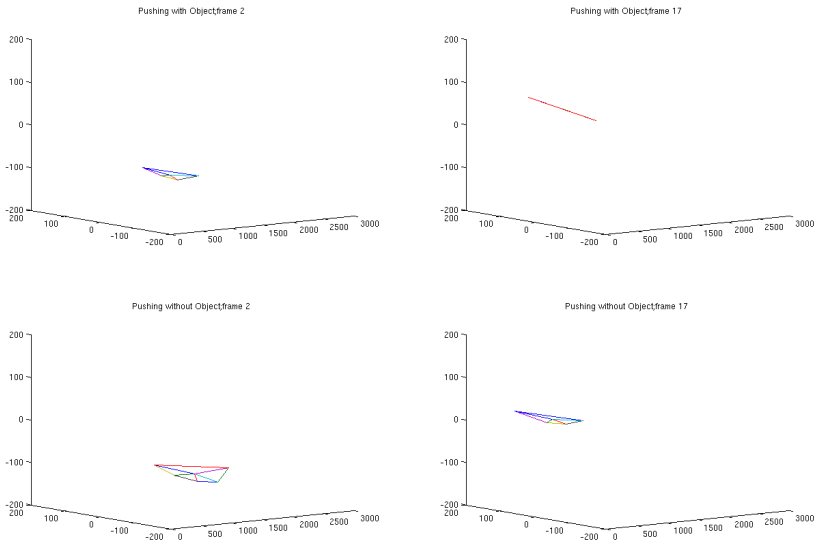


(e) Extraction 3

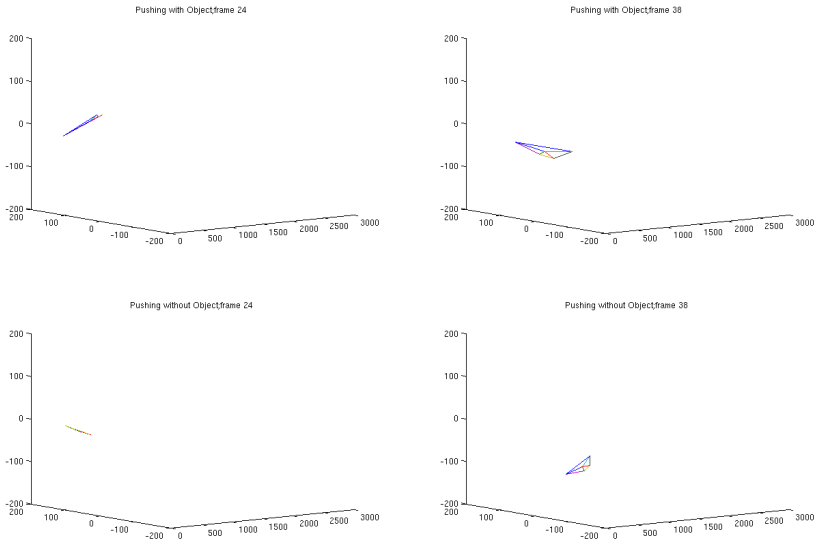


(f) Image 3

Figure 6.13: Ground Truth fingertip extraction and images correspondent to the previous Figure (Figure 6.12)

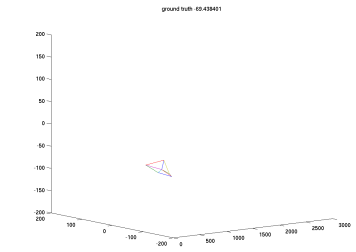


(a) Capture 1 (b) Capture 2

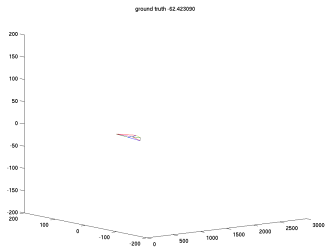


(c) Capture 3 (d) Capture 4

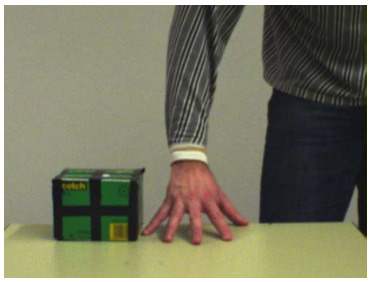
Figure 6.14: Fingertip extraction performing the moving sequence with and without object, four captures. ICP estimator



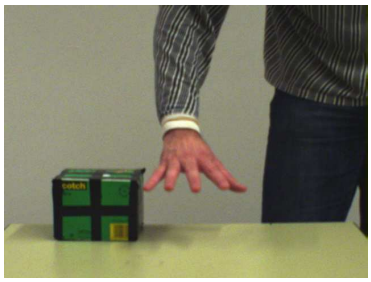
(a) Extraction 1



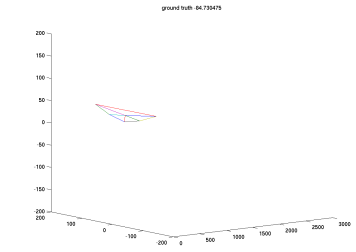
(b) Extraction 2



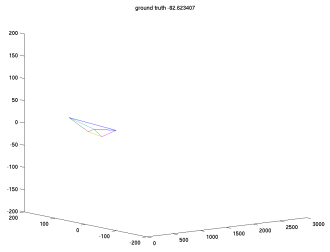
(c) Image 1



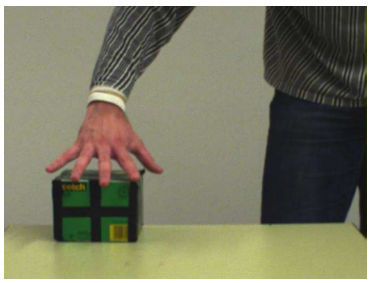
(d) Image 2



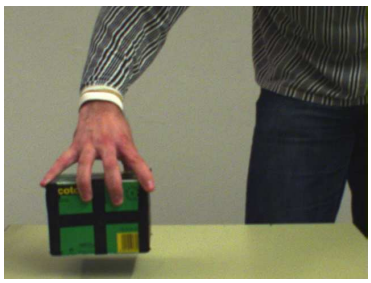
(e) Extraction 3



(f) Extraction 4

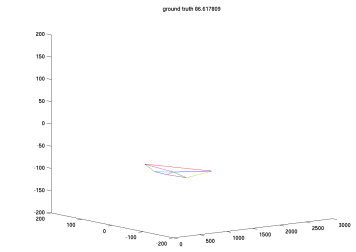


(g) Image 3

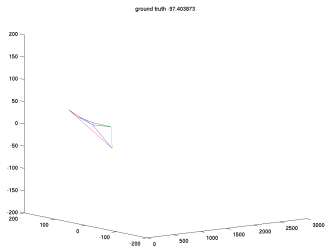


(h) Image 4

Figure 6.15: Ground Truth fingertip extraction and images correspondent to Figure 6.14, sequence with object

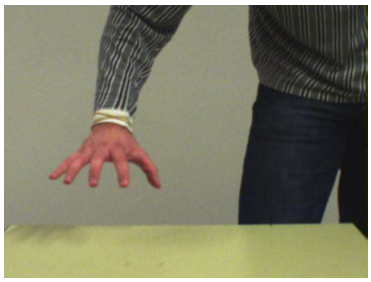


(a) Extraction 1

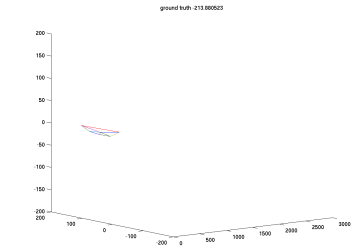


(b) Extraction 2

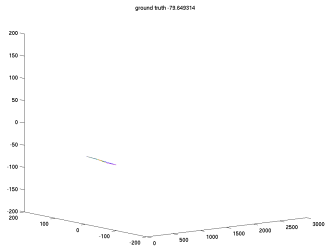
(c) Image 1



(d) Image 2



(e) Extraction 3

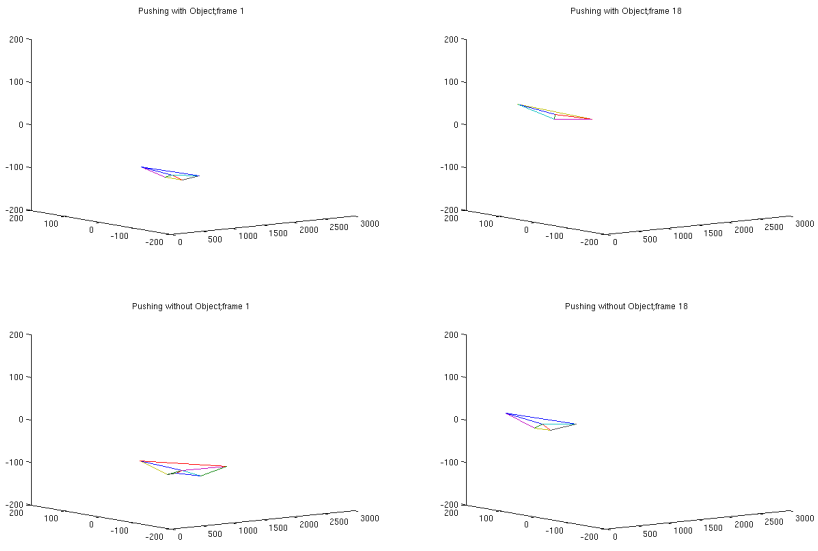


(f) Extraction 4

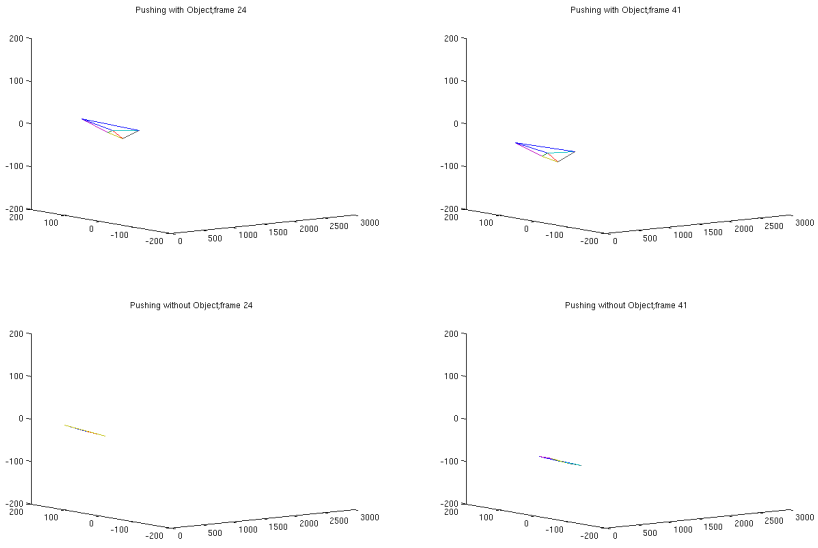
(g) Image 3

(h) Image 4

Figure 6.16: Ground Truth fingertip extraction and images correspondent to Figure 6.14, sequence without object

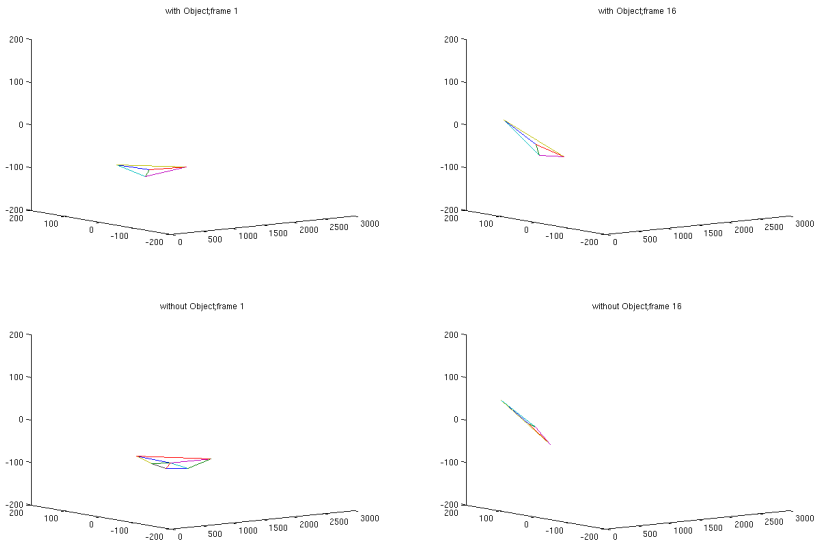


(a) Capture :1 (b) Capture 2

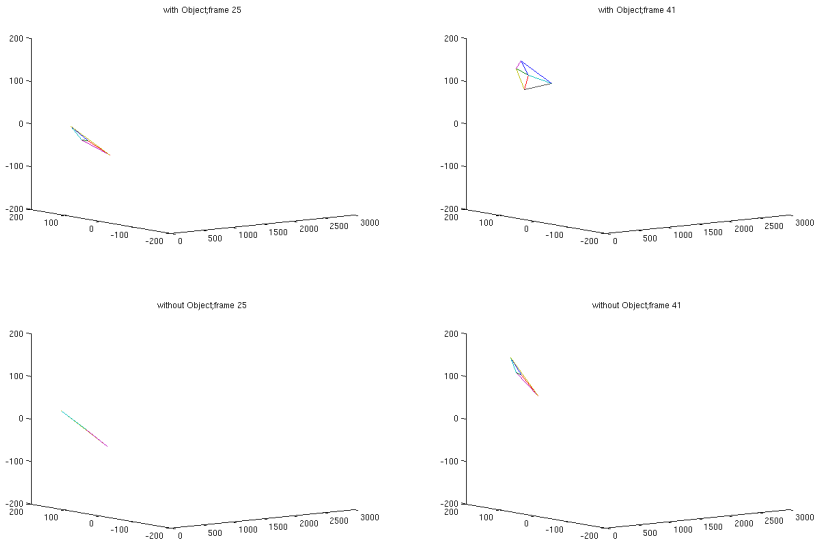


(c) Capture 3 (d) Capture 4

Figure 6.17: Fingertip extraction performing the moving sequence with and without object, four captures. TW estimator

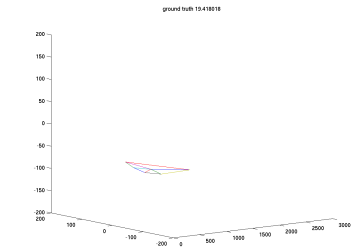


(a) Capture 1 (b) Capture 2

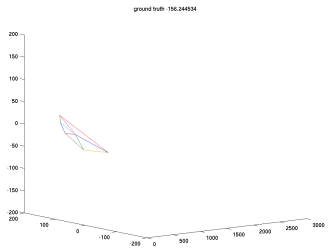


(c) Capture 3 (d) Capture 4

Figure 6.18: Fingertip extraction performing the pinch Grasp sequence with and without object, four captures. ICP estimator



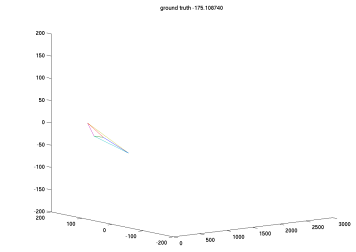
(a) Extraction 1



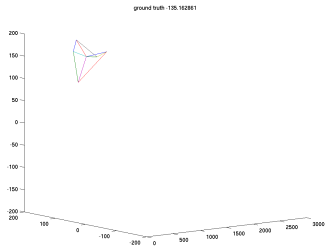
(b) Extraction 2

(c) Image 1

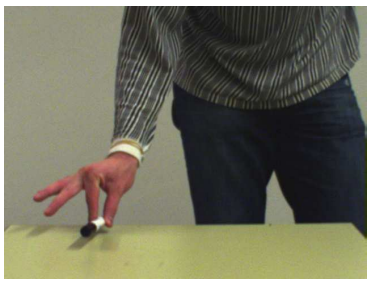
(d) Image 2



(e) Extraction 3



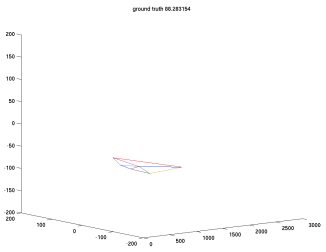
(f) Extraction 4



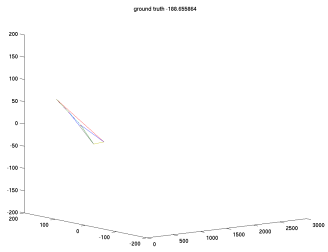
(g) Image 3

(h) Image 4

Figure 6.19: Ground Truth fingertip extraction and images correspondent to Figure 6.18, sequence with object

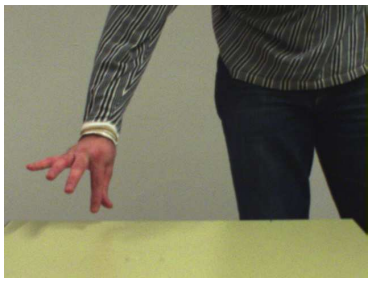


(a) Extraction 1

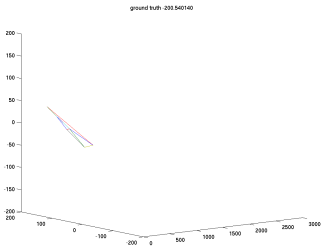


(b) Extraction 2

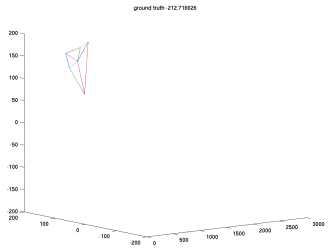
(c) Image 1



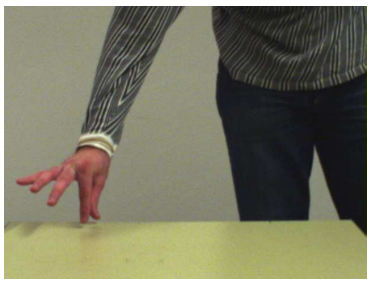
(d) Image 2



(e) Extraction 3



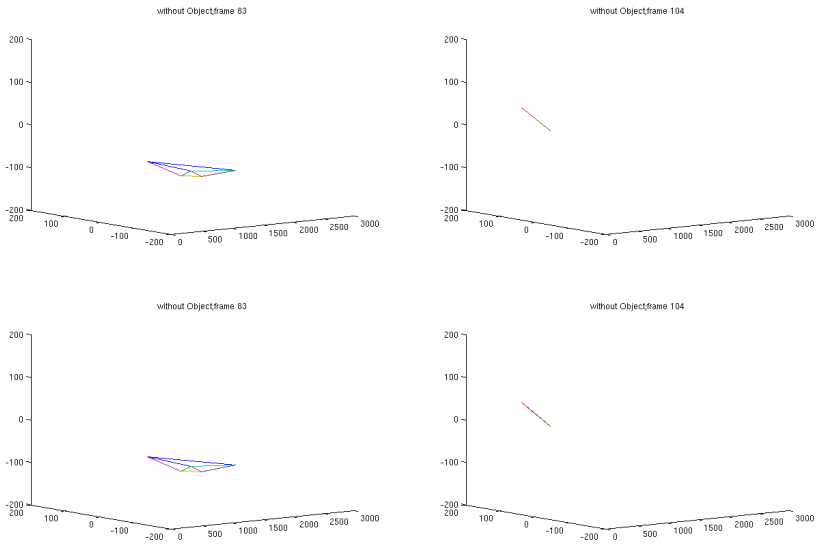
(f) Extraction 4



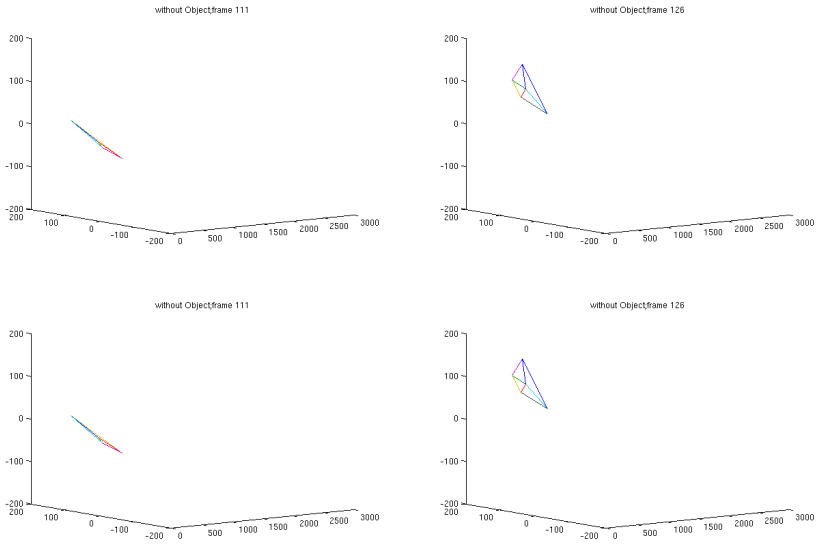
(g) Image 3

(h) Image 4

Figure 6.20: Ground Truth fingertip extraction and images correspondent to Figure 6.18, sequence without object

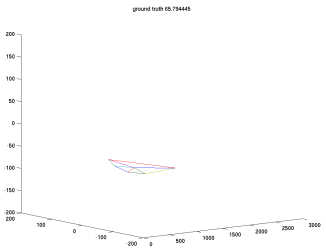


(a) Capture :1 (b) Capture 2

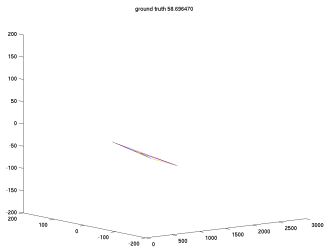


(c) Capture 3 (d) Capture 4

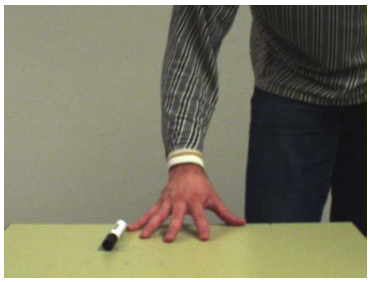
Figure 6.21: Fingertip extraction performing the pinch Grasp sequence with and without object, four captures. TW estimator



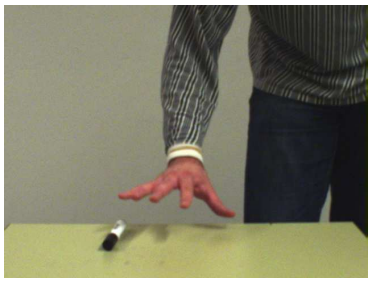
(a) Extraction 1



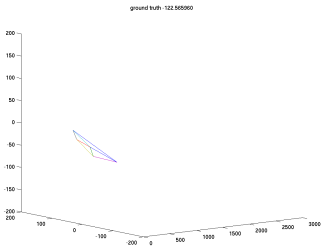
(b) Extraction 2



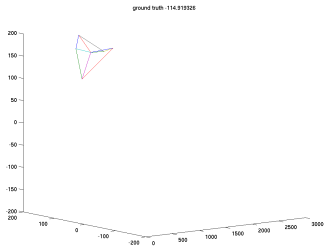
(c) Image 1



(d) Image 2



(e) Extraction 3

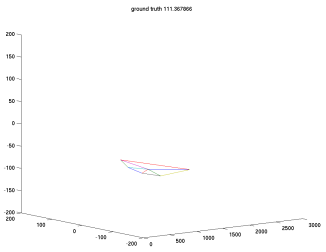


(f) Extraction 4

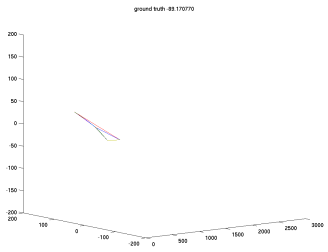
(g) Image 3

(h) Image 4

Figure 6.22: Ground Truth fingertip extraction and images correspondent to Figure 6.21,sequence with object

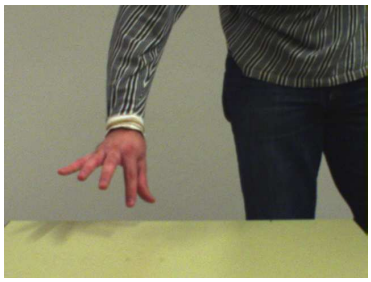


(a) Extraction 1

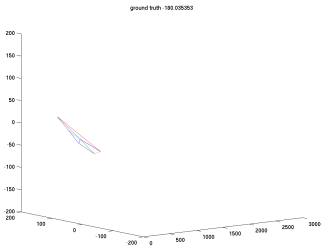


(b) Extraction 2

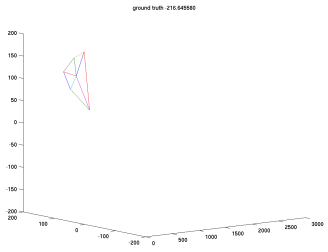
(c) Image 1



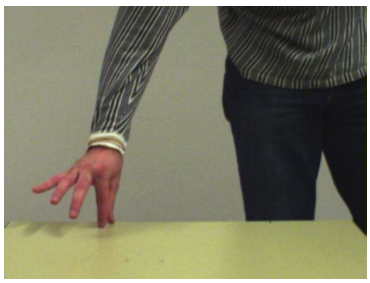
(d) Image 2



(e) Extraction 3



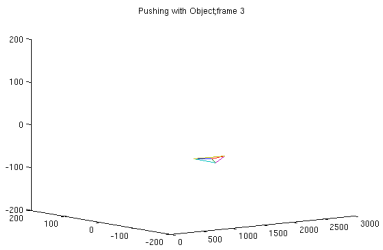
(f) Extraction 4

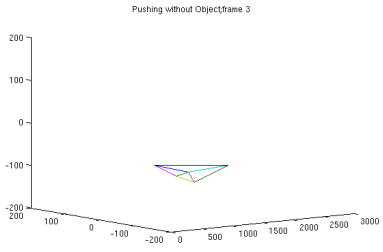


(g) Image 3

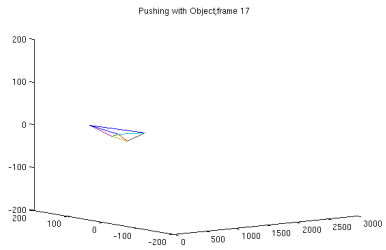
(h) Image 4

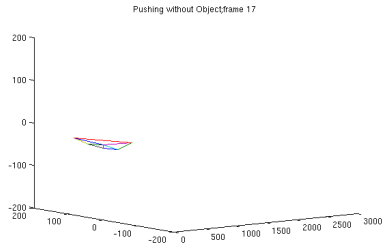
Figure 6.23: Ground Truth fingertip extraction and images correspondent to Figure 6.21, sequence without object



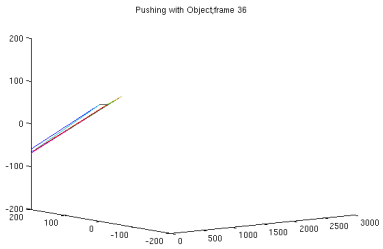


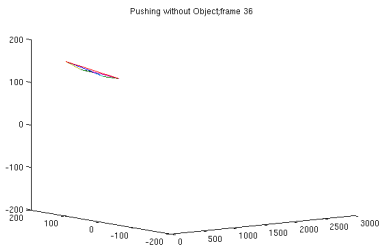
(a) Capture 1



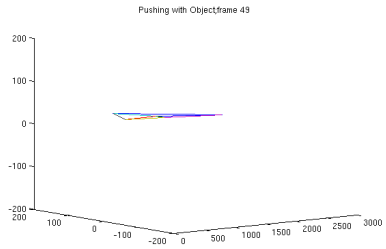


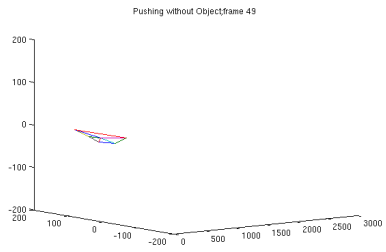
(b) Capture 2





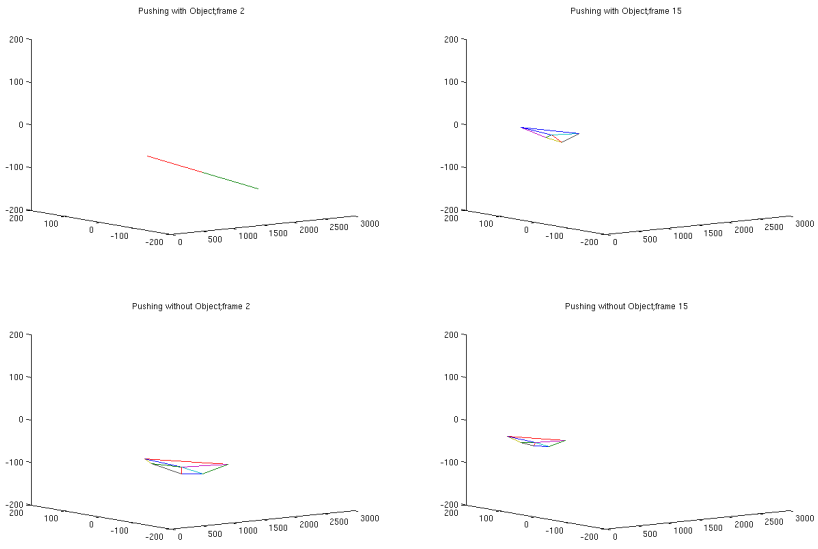
(c) Capture 3



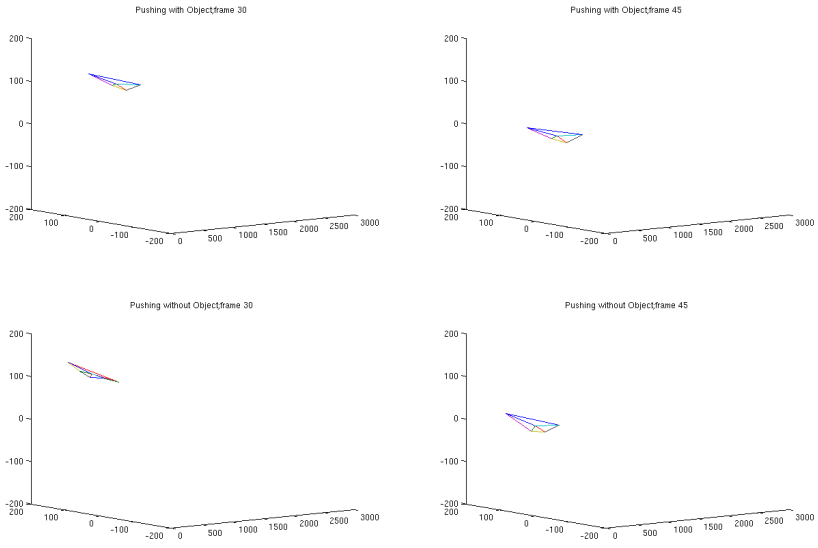


(d) Capture 4

Figure 6.24: Fingertip extraction performing the power Grasp sequence with and without object, four captures. ICP estimator

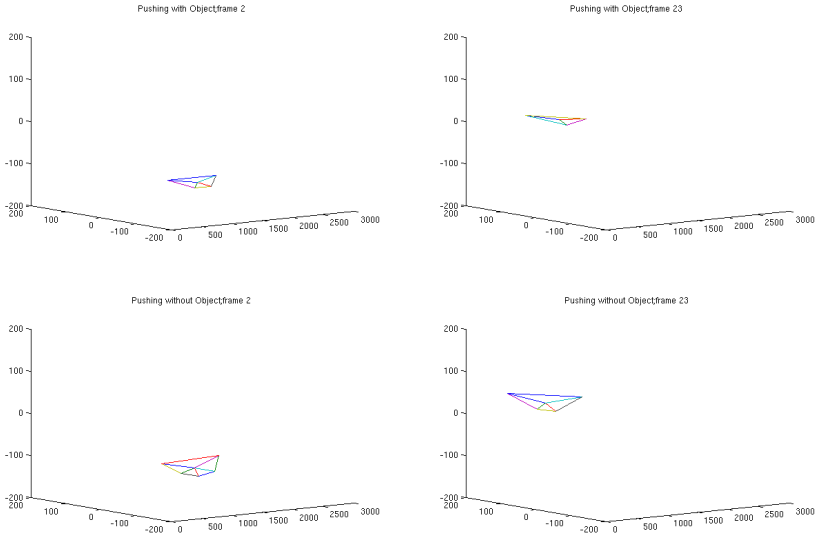


(a) Capture :1 (b) Capture 2

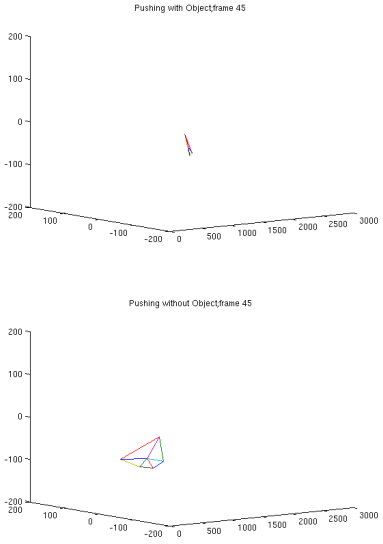


(c) Capture 3 (d) Capture 4

Figure 6.25: Fingertip extraction performing the power Grasp sequence with and without object, four captures. TW estimator

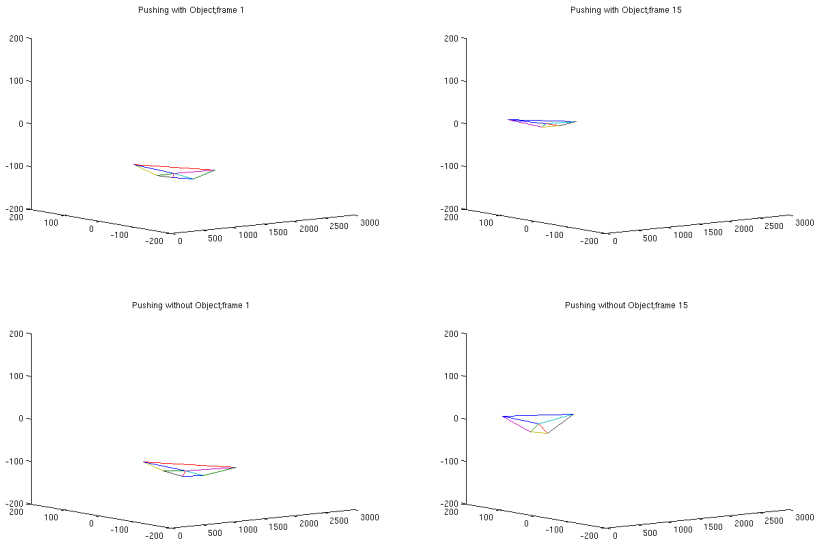


(a) Capture 1 (b) Capture 2

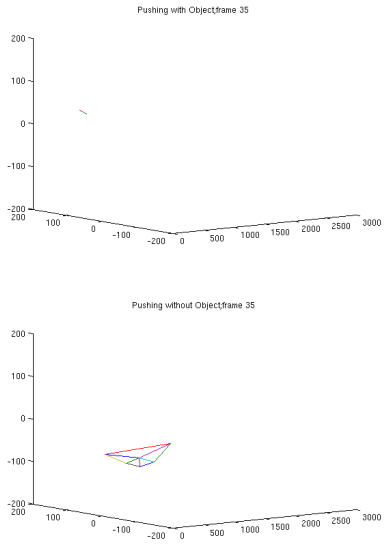


(c) Capture 3

Figure 6.26: Fingertip extraction performing the push sequence with and without object, four captures. ICP estimator

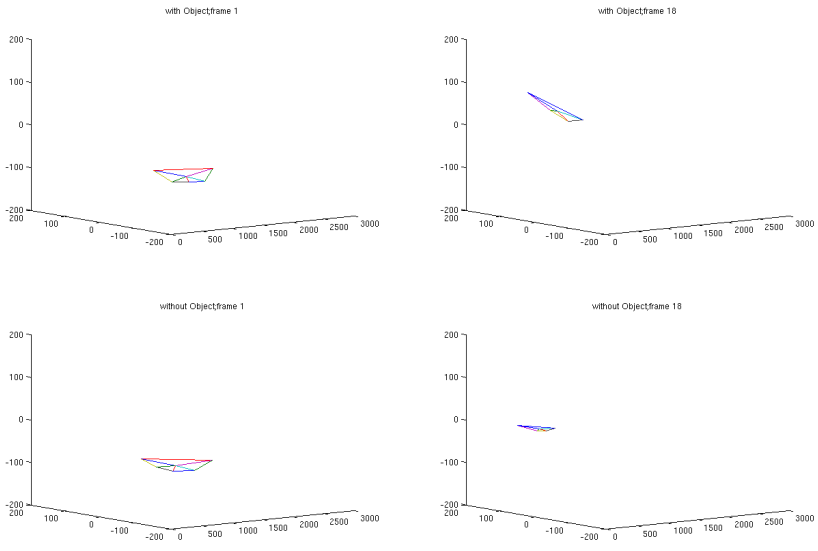


(a) Capture :1 (b) Capture 2

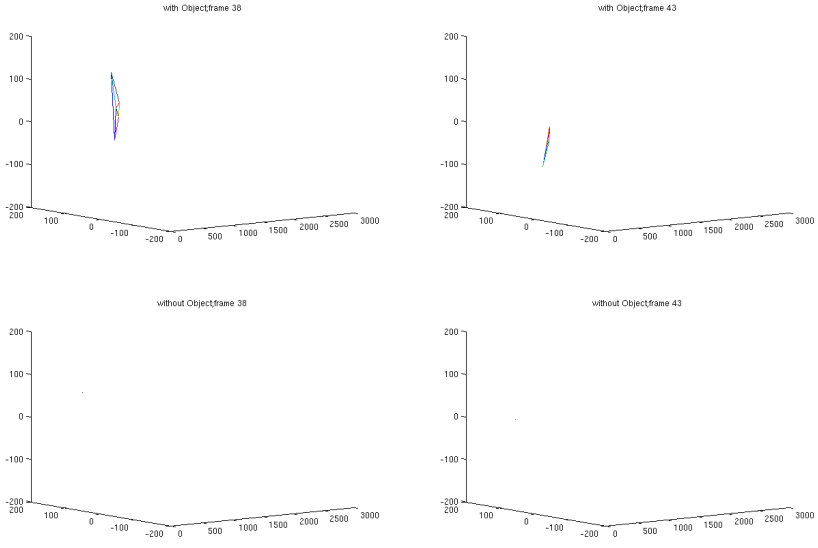


(c) Capture 3

Figure 6.27: Fingertip extraction performing the push sequence with and without object, four captures. TW estimator

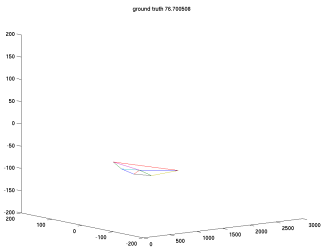


(a) Capture 1 (b) Capture 2

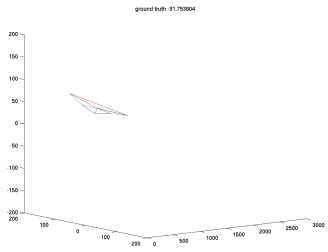


(c) Capture 3 (d) Capture 4

Figure 6.28: Fingertip extraction performing the rotating sequence with and without object, four captures. ICP estimator

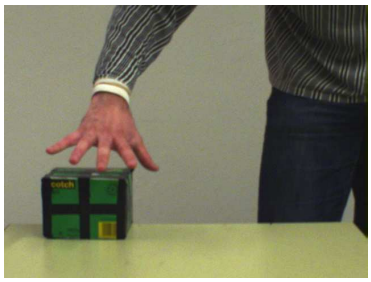


(a) Extraction 1

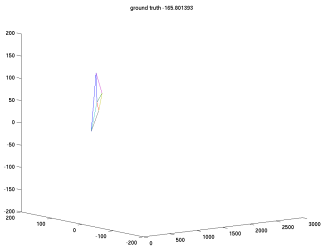


(b) Extraction 2

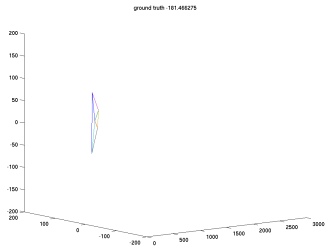
(c) Image 1



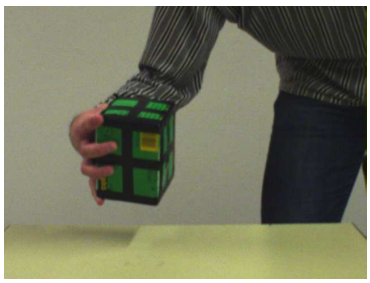
(d) Image 2



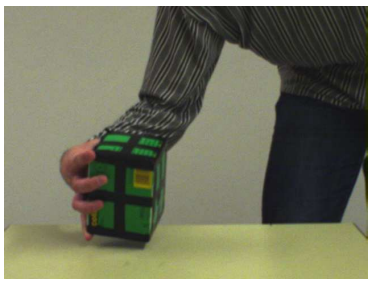
(e) Extraction 3



(f) Extraction 4

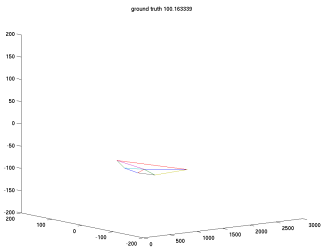


(g) Image 3

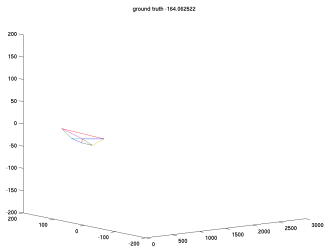


(h) Image 4

Figure 6.29: Ground Truth fingertip extraction and images correspondent to Figure 6.28,sequence with object



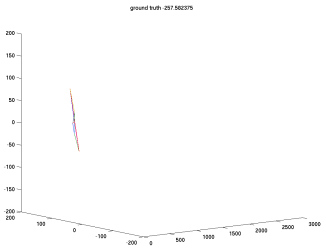
(a) Extraction 1



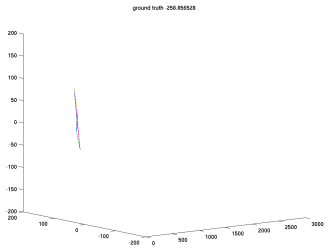
(b) Extraction 2

(c) Image 1

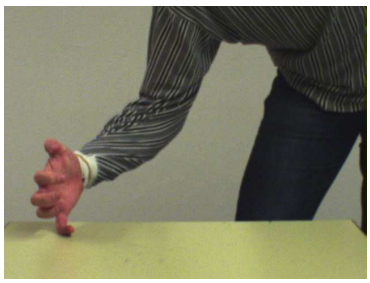
(d) Image 2



(e) Extraction 3



(f) Extraction 4



(g) Image 3

(h) Image 4

Figure 6.30: Ground Truth fingertip extraction and images correspondent to Figure 6.28, sequence without object

In order to get quantitative data, glove based devices with magnetic sensors would be a good approach [19, 59]. Another possibility is to mark manually the fingertips in the pictures and perform the 3D estimation based on triangulation. The last one is the method used in this thesis. The results presented here are not totally complete, since not all the pictures were manually marked. In the previous figures the 3D fingertip position can be compared visually with the extraction based on manually marked fingertips. Some error probability density graphics have been included also.

The accuracy of the depth estimator is definitely dependent on the axis; while x and y accuracy is similar, in z the accuracy is much worse (see Figure 6.31).This is logical: x and y are just computed by scaling and translating the fingertips, while z depends strongly in the calibration of the cameras and in the exactitude of x and y location of the fingertips. Small errors in the fingertip location on the picture are propagated and amplified to the depth coordinate.

In Figure 6.32 it can be seen a comparison between depth estimators based on ICP and TW algorithms. It can be seen that the performance of both methods is similar. TW algorithm seems to achieve a slightly better depth extraction. It has to be taken into account that some fingertips are not detected in the course of the tracking: the ICP algorithm lost a fingertip 162 times while TW algorithm lost 204 fingertips.

As it was said before, there are sequences easier to be tracked than other ones. It can be seen in Figure 6.33. Waving and moving have the best accuracy among the different actions. During these actions, the behaviour of the hand is almost planar. For this reason the accuracy of ICP algorithm is much lower in the other cases. The behaviour of TW algorithm in the rest of the sequences is slightly better (except in pinch Grasp sequence).

Finally, a comparison between actions with and without objects can be seen in Figure 6.34. As it was said before, accuracy in sequences without objects is better than in those with objects, due to the lower number of occlusions.

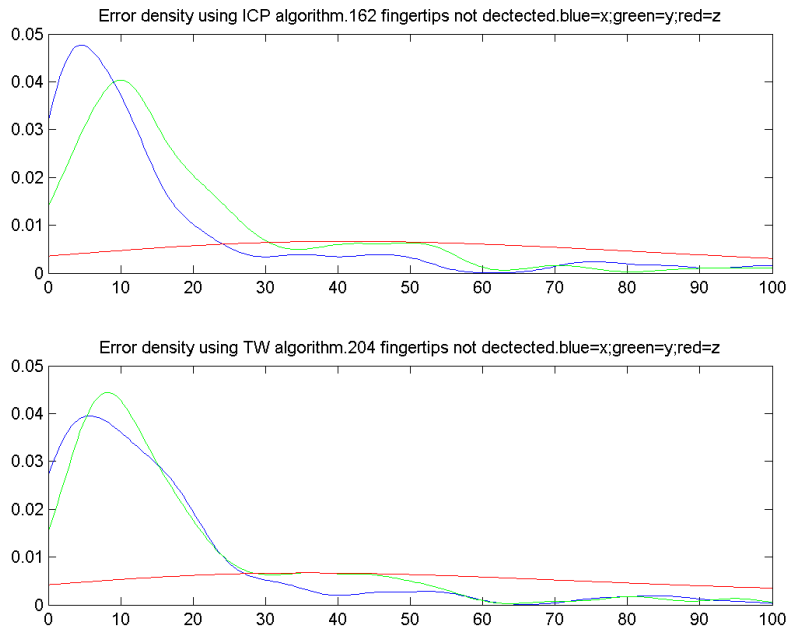


Figure 6.31: Comparison between accuracy of depth estimation in x, y and z axis

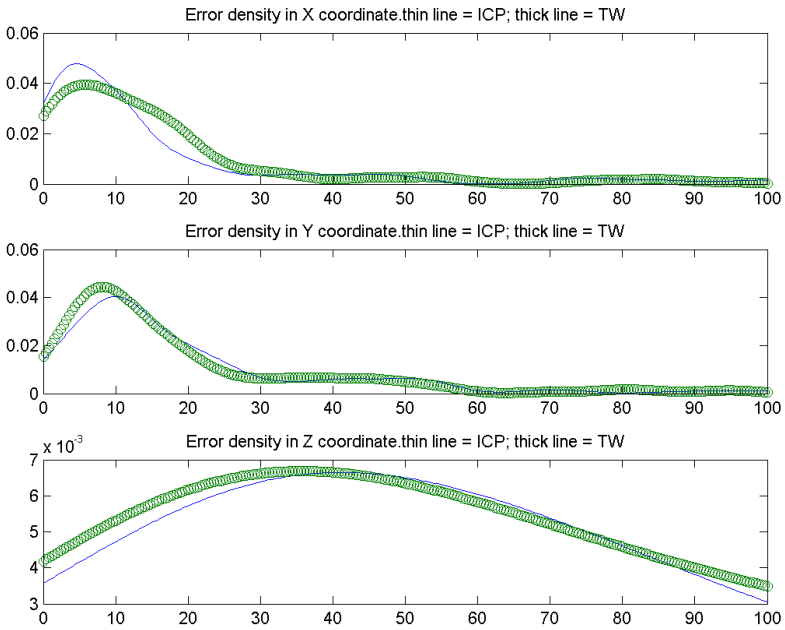


Figure 6.32: Comparison between accuracy of depth estimation with ICP and TW algorithm

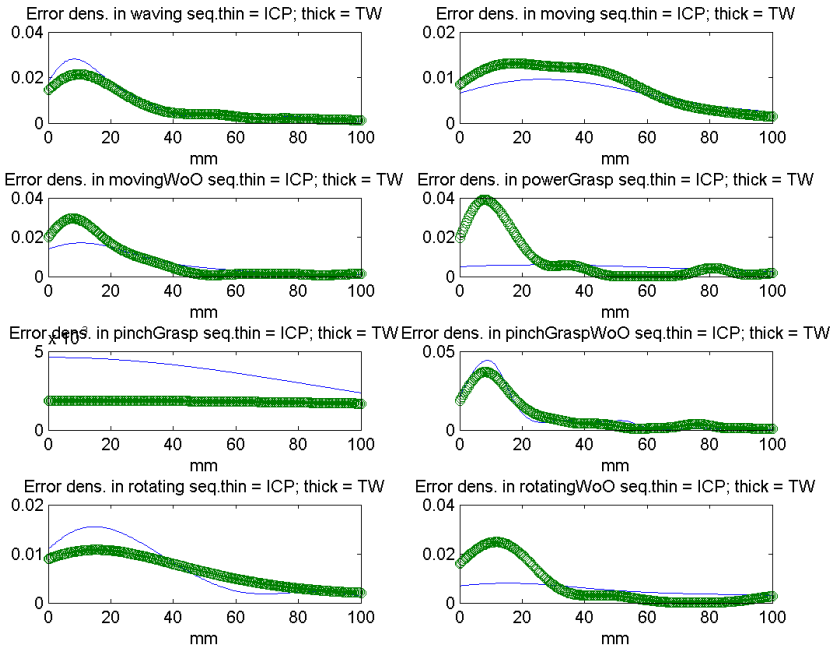


Figure 6.33: Error probability density of different sequences

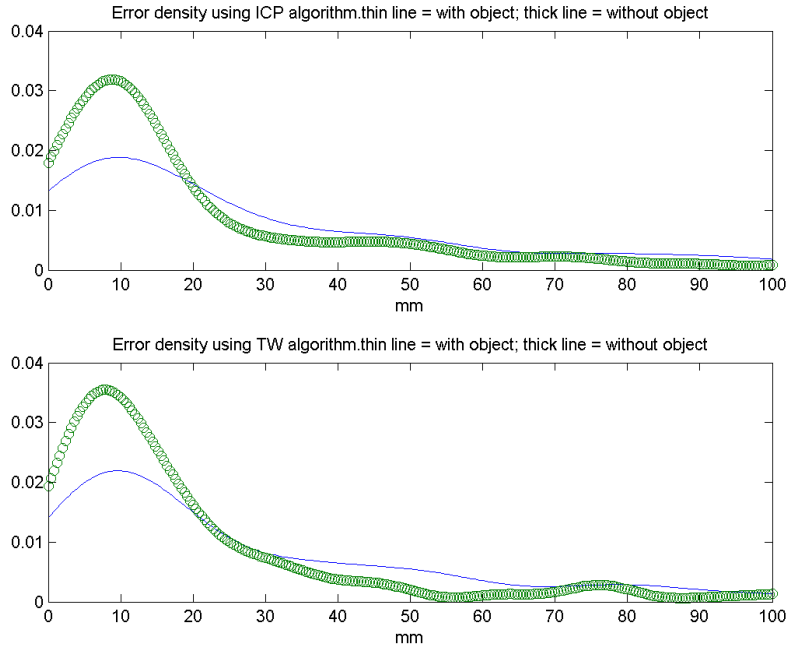


Figure 6.34: Comparison between accuracy of depth estimation in sequences with and without object

Chapter 7

Conclusions

This thesis tried to face the problems related with the user interface in a Programming By Demonstration scenario. Due to the complexity of the task and the limited time of development, the results achieved do not compose a total solution.

A visual system for object recognition is almost compulsory for these kind of systems. Since the goal is to have a system general enough to perform different kind of tasks in a regular environment, it is desirable avoid constraints on the methods used as far as possible. For this reason a stereo head was chosen as the input for the device: monocular systems need strict constraints to get depth information.

The thesis is focused on an interface based on hand gestures that can be used either as command generator or as examples for learning by demonstration. The first problem to be faced when a stereo system is used is to match the images that come from each camera. Two algorithms are proposed as a solution for this problem: iterative closest point and dynamic time warping. The first method is fast but strictly constrained about the position of the points to be matched. The second one is slower (the actual implementation is not optimized) but improves the results achieved by the first one during complex hand positions.

The other main problem faced in this thesis is how to extract the hand pose from the stereo-matched images of the hand. A database solution is one of the most used approaches nowadays. However, the size of a database capable of recognizing any kind of regular hand movement is quite big. They have also the problem of recognizing hands with different sizes, shapes and colors. For these reasons, the chosen solution is based on a general kinematic model, applicable to any hand external appearance. The hand configuration is extracted from a closed form solution of a simplified inverse kinematics model. If the approximated model is not accurate enough, optimization

methods would be applied to improve the performance.

In conclusion, the speed of a closed form inverse kinematics solution would be used for alleviating the slowness of a dynamic time warping matching method in a recognition system close to real-time behaviour.

Bibliography

[1] Affine transformation. "http://mathworld.wolfram.com/AffineTransformation.html".

[2] libdc1394 v1 examples. "http://www.ptgrey.com/support/kb/data/libdc-1_imaging.tgz".

[3] Marlin f-080b/c xga camera ieee1394. "http://www.alliedvisiontec.com/produktinfos.html?t=produktinfos&o=17&a=selectid".

[4] Yorick robot head series. "http://www.isrg.reading.ac.uk/yorick/index.htm".

[5] S. Ahmad. A usable real-time 3D hand tracker. In *28th Asilomar Conference on Signals, Systems and Computers*, pages 1257–1261. IEEE Computer Society Press, 1995.

[6] A. A. Argyros. 3d tracking of multiple skin-colored regions by moving stereoscopic observer. *Applied Optics, Information Processing Journal, Special Issue on Target Detection*, 43(2):366–378, January 30 2004.

[7] A. A. Argyros and M. I. A. Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *European Conference on Computer Vision*, pages Vol III: 368–379, 2004.

[8] A. A. Argyros and M. I. A. Lourakis. Tracking skin-colored objects in real-time. In *Cutting Edges Robotics Book*. Advanced Robotic Systems International, 2005.

[9] A. A. Argyros and M. I. A. Lourakis. Binocular hand-tracker and reconstruction based on 2d-shape matching. In *ICPR*, 2006.

[10] A. A. Argyros and M. I. A. Lourakis. Vision-based interpretation of hand gestures for remote control of a computer mouse. In *Computer Vision in Human-Computer Interaction*, pages 40–51, 2006.

[11] V. Athitsos and S. Sclaroff. An appearance-based framework for 3d hand shape classification and camera viewpoint estimation. In *International Conference on Automatic Face and Gesture Recognition*, pages 40–45, 2002.

[12] V. Athitsos and S. Sclaroff. Estimating 3D hand pose from a cluttered image. In *CVPR*, pages 432–442. IEEE Computer Society, 2003.

[13] L. Bretzner, I. Laptev, and T. Lindeberg. Hand gesture recognition using multi-scale colour features, hierarchical models and particle filtering. In *International Conference on Automatic Face and Gesture Recognition*, pages 405–410, 2002.

[14] H. Ritter C. Nölker. Visual recognition of continuous hand postures, 1999.

[15] M. Cabido-Lopes and J. Santos-Victor. Visual transformations in gesture imitation: what you see is what you do. In *ICRA*, pages 2375–2381. IEEE, 2003.

[16] M. Cabido-Lopes and J. Santos-Victor. Visual learning by imitation with motor representations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(3):438–449, 2005.

[17] T. S. Caetano, S. D. Olabarriaga, and D. A. Couto Barone. Performance evaluation of single and multiple-gaussian models for skin color modeling. In *SIBGRAPI*, pages 275–282. IEEE Computer Society, 2002.

[18] T. Campos. Art tracker demos. Website. "`http://www.robots.ox.ac.uk/~teo/art/`".

[19] J. Comas. Intelligent task-level grasp mapping for robot control. Master's thesis, CVAP/KTH, 2006.

[20] J. L. Crowley. Finger tracking as an input device for augmented reality, June 15 1995.

[21] J. Davis and M. Shah. Visual gesture recognition, August 11 1994.

[22] T. E. de Campos, B. Tordoff, and D. W. Murray. Recovering articulated pose: A comparison of two pre and postimposed constraint methods. *IEEE Trans. Pattern Anal. Mach. Intell*, 28(1):163–168, 2006.

[23] Q. Delamarre and O. D. Faugeras. Finding pose of hand in video images: A stereo-based approach. In *FG*, pages 585–590. IEEE Computer Society, 1998.

[24] D. E. DiFranco, T. J. Cham, and J. M. Rehg. Reconstruction of 3D figure motion from 2D correspondences. In *IEEE Computer Vision and Pattern Recognition or CVPR*, pages I:307–314, 2001.

[25] A. E., R. Duraiswami, D. Harwood, and L. S. Davis. Background and foreground modeling using nonparametric kernel density for visual surveillance, September 21 2002.

[26] J. Fan, D. Yau, A. K. Elmagarmid, and W. G. Aref. Automatic image segmentation by integrating color-edge extraction and seeded region growing. *IEEE Transactions on Image Processing*, 10(10):1454–1466, 2001.

[27] R. Feris, R. Raskar, L. Chen, K. Tan, and M. Turk. Discontinuity preserving stereo with small baseline multi-flash illumination. In *ICCV*, pages 412–419. IEEE Computer Society, 2005.

[28] D.J. Giurintano, A.M. Hollister, W.L. BUford, D.E. Thompson, and L.M. Myers. A virtual five-link model of the thumb.

[29] D. O. Gorodnichy and A. Yogeswaran. Detection and tracking of pianist hands and fingers. In *Computer and Robot Vision*, pages 63–63, 2006.

[30] T. Gumpp, P. Azad, K. Welke, E. Oztop, R. Dillmann, and G. Cheng. Unconstrained real-time markerless hand tracking for humanoid interaction.

[31] E. Holden. Visual recognition of hand motion, 1997.

[32] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *FRAME-RATE: Frame-rate Applications, Methods and Experiences with Regularly Available Technology and Equipment*, 1999.

[33] Y.K. Ho H.Y. Guan, C.S. Chua. Hand posture estimation from 2d monocular image. In *3DIM99*, pages 424–429, 1999.

[34] H. Dzindo I. Infantino, A. Chella and I. Macaluso. A cognitive system for human interaction with a robotic hand.

[35] C. Jennings. Thesis proposal: 3d tracking & recognition of natural objects.

[36] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection. In *CVPR*, pages 1274–1280. IEEE Computer Society, 1999.

[37] J. Lee and T. L. Kunii. Model-Based analysis of hand posture. *IEEE Computer Graphics and Applications*, 15(5):77–86, September 1995.

[38] J. Y. Lin, Y. Wu, and T. S. Huang. 3D model-based hand tracking using stochastic direct search method. In *FGR*, pages 693–698. IEEE Computer Society, 2004.

[39] T. Lindeberg and I. Laptev. Tracking of multi-state hand models using particle filtering and a hierarchy of multi-scale image features. In *Scale Space*, pages xx–yy, 2001.

[40] M. Cabido Lopes, J. Santos-Victor, and Escola Superior De Tecnologia. Motor representations for hand gesture recognition and imitation, November 06 2003.

[41] S. Lu, D. N. Metaxas, D. Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *CVPR*, pages 443–450. IEEE Computer Society, 2003.

[42] D. Marr and E. Hildreth. Theory of edge detection. Technical Report AIM-518, MIT Artificial Intelligence Laboratory, April 6 1979.

[43] M.Bray, E.Koller, and L.Van Gool. Smart particle filtering for 3D hand tracking. In *FGR*, pages 675–680. IEEE Computer Society, 2004.

[44] A. Miller. Graspit. `"http://www1.cs.columbia.edu/~amiller/graspit/"`.

[45] C. S. Myers. A comparative study of several dynamic time warping algorithms for speech recognition. Master's thesis, M.I.T., Dept. of Electrical Engineering and Computer Science, 1980.

[46] Y. Kuno N. Shimada, Y. Shirai and J. Miura. Hand gesture estimation and model refinement using monocular camera - ambiguity limitation by inequality constraints. In *International Conference on Automatic Face and Gesture Recognition*, pages 268–273, 1998.

[47] H. T. Nguyen, M. Worring, and R. van den Boomgaard. Occlusion robust adaptive template tracking. In *ICCV*, pages 678–683, 2001.

[48] C. Nölker and H. Ritter. Detection of fingertips in human hand movement sequences. In Ipke Wachsmuth and Martin Fröhlich, editors, *Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction (GESTURE-97)*, volume 1371 of *LNAI*, pages 209–218, Berlin, September 17–19 1998. Springer.

[49] K. Oka, Y. Sato, and H. Koike. Real-time fingertip tracking and gesture recognition. *IEEE Computer Graphics and Applications*, 22(6):64–71, 2002.

[50] H. Ouhaddi, P. Horain, and R. C. Fourier. Hand tracking by 3D model registration, June 29 1999.

[51] V. Pavlovic, R. Sharma, and T. S. Huang. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Trans. Pattern Anal. Mach. Intell*, 19(7):677–695, 1997.

[52] J. M. Rehg and T. Kanade. Digiteyes: Vision-based hand tracking for human-computer interaction, July 30 1994.

[53] J. M. Rehg and T. Kanade. Model-based tracking of self-occluding articulated objects. In *ICCV*, pages 612–617, 1995.

[54] H. Rijpkema and M. Girard. Computer animation of knowledge-based human grasping. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 339–348, July 1991.

[55] H. Ritter, J. J. Steil, C. Nölker, Frank Röthling, and Patrick C. McGuire. Neural architectures for robot intelligence. *CoRR*, cs.RO/0410042, 2004.

[56] R. Rosales, V. Athitsos, L. Sigal, and S. Sclaroff. 3D hand pose reconstruction using specialized mappings. In *ICCV*, pages 378–387, 2001.

[57] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. pages 145–152, 2001.

[58] A. Senior, A. Hampapur, Y. L. Tian, L. Brown, S. Pankanti, and R. Bolle. Appearance models for occlusion handling. In *International Workshop on Performance Evaluation of Tracking and Surveillance*, pages xx–yy, 2001.

[59] I. Serrano. Human action recognition based on linear and non-linear dimensionality reduction using pca and isomap. Master's thesis, Center for Autonomous Systems, Royal Institute of Technology, 2006.

[60] C. Shan, Y. Wei, T. Tan, and F. Ojardias. Real time hand tracking by combining particle filtering and mean shift. In *FGR*, pages 669–674. IEEE Computer Society, 2004.

[61] B. Stenger. *Model-Based Hand Tracking Using A Hierarchical Bayesian Filter*. PhD thesis, University of Cambridge, Cambridge, UK, March 2004.

[62] B. Stenger, P. R. S. Mendonça, and R. Cipolla. Model-based 3D tracking of an articulated hand. In *CVPR*, pages 310–315. IEEE Computer Society, 2001.

[63] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Filtering using a tree-based estimator. In *ICCV*, pages 1063–1070. IEEE Computer Society, 2003.

[64] B. D. R. Stenger, P. R. S. Mendonca, and R. Cipolla. Model-based hand tracking using an unscented kalman filter. In *British Machine Vision Conference*, page Session 2: Tracking &. Sequences, 2001.

[65] K. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. Camera calibration toolbox for matlab. `"http://www.vision.caltech.edu/bouguetj/calib_doc/"`.

[66] D. Sturman and D. Zeltzer. A survey of glove-based input. *Computer Graphics and Applications*, 14(1):30–39, 1994.

[67] E. B. Sudderth, M. I. Mandel, W. T. Freeman, and A. S. Willsky. Visual hand tracking using nonparametric belief propagation. In *Workshop on Generative Model Based Vision*, page 189, 2004.

[68] C. Tomasi, S. Petrov, and A. Sastry. 3D tracking = classification + interpolation. In *International Conference on Computer Vision*, pages 1441–1448, 2003.

[69] Y. Wu and T. Huang. Capturing articulated human hand motion: A divide-and-conquer approach. In *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, volume I, pages 606–611, Los Alamitos, CA, September 20–27 1999. IEEE.

[70] Z. Zhang, Y. Wu, Y. Shan, and S. Shafer. Visual panel: Virtual mouse, keyboard and 3D controller with an ordinary piece of paper, December 01 2001.

[71] Z. Y. Zhang. Iterative point matching for registration of free-form curves. In *INRIA*, 1992.

[72] Y. Zhou and H. Tao. A background layer model for object tracking through occlusion. In *International Conference on Computer Vision*, pages 1079–1085, 2003.

Seguimiento y estimación de la postura de la mano mediante un sistema de visión en estéreo

Autor: Javier Romero González-Nicolás
Tutor: Danica Kragic
Institución: CVAP-CAS KTH
Estocolmo, 19/02/2007

Introducción

El objetivo de este proyecto consiste en estimar la postura de la mano mediante el tratamiento de imágenes provinientes de un sistema de visión estereoscópico.

Dicha estimación es muy valiosa en el contexto de "Programación por Demostración", en el que el programador-usuario ejecuta una tarea que será más tarde imitada por el robot.

Planteamiento del Problema

Dada la complejidad de la tarea planteada anteriormente, únicamente ciertas partes de la misma fueron abordadas en este proyecto. En la Figura 5.1 podemos ver en un esquema general de un sistema de estimación de gestos manuales. Como punto de partida utilizamos el sistema explicado en [6, 9, 10], que comprende todos los módulos salvo los dos últimos, "inverse mapping" y "robot adaptation". Los módulos diseñados o modificados en este proyecto son básicamente el emparejado de puntos comunes provinientes de las imágenes en estéreo ("Match Stereo Points") y el diseño de un sistema que obtiene los ángulos de las articulaciones de la mano dada la posición y orientación de la palma, y las posiciones de las puntas de los dedos ("Kinematic closed Form solution"). También se ha trabajado en la adaptación del sistema al nuevo hardware.

Trabajo previo

La estimación de la postura de la mano ha sido un campo de investigación muy activo en los ultimos 20 años. En los primeros sistemas se usaron, principalmente, dispositivos basados en guantes con distintos sensores: magnéticos, acústicos u ópticos. Una buena comparativa de los sistemas de la época puede verse en [66]. El problema de dichos dispositivos

estriba en la necesidad por parte del usuario de modificar su comportamiento habitual para utilizar el sistema (tiene que "ponerse" los guantes para que sus gestos manuales sean reconocidos). El uso de dichos guantes se ha decrementado en favor de los sistemas basados en cámaras, dada la resolución parcial o total de los problemas de estos sistemas relatados en [66].

Se pueden hacer distintas clasificaciones entre los sistemas actuales basados en cámaras, dependiendo del dispositivo de adquisición de imágenes, el método de estimación de postura de la mano, etc. Una buena introducción al problema y las distintas implementaciones puede encontrarse en [51]. Más detalles acerca de las diferentes implementaciones de los distintos módulos que forman un sistema de estimación de la postura de la mano pueden encontrarse en la sección 3.

Modelos teóricos

En este capítulo se desarrollan los modelos usados en el proyecto. En primer lugar, se desarrolló un modelo cinemático de la mano, mediante el cual podemos extraer la postura de la mano a partir de la localización de ciertos puntos clave en la misma. La configuración de los dedos (salvo el pulgar) es similar a los modelos utilizados anteriormente [37, 54, 33]. Estos modelan el dedo como un cuerpo rígido compuesto por tres segmentos (las falanges) y con un total de 4 ejes de rotación (ver Figura 4.2). Mientras que al dedo pulgar se le suele asignar un modelo más complejo (ver Figura 4.4) sin solución analítica, nosotros optamos por utilizar un modelo similar al de los otros dedos, para así utilizar el mismo desarrollo trigonométrico. El desarrollo de la solución puede verse en la sección 4.1, y un desarrollo similar es expuesto en [33]. El resultado de todo este desarrollo es que, dada la posición de las puntas de los dedos y la posición y orientación de la palma de la mano, podemos extraer los distintos ángulos que definen por completo la postura de la mano en 3D.

En este capítulo también se explican modelos tomados de otros sistemas, como el modelo simplificado de la mano usado para encontrar correspondencias entre manos provinientes de imágenes en distintos instantes (correspondencias temporales) y en las distintas cámaras (correspondencia en estéreo). Dicho modelo proviene del sistema expuesto en [6, 9, 10].

Asimismo se expone el modelo de color de piel, proviniente del mismo sistema [6, 9, 10] y una introducción a la geometría epipolar, elemento clave en la mejora del procedimiento de extracción en 3D del contorno de la mano, necesario para la posterior extracción de la postura completa de la mano.

Implementación

En este capítulo se exponen distintos detalles de la implementación final de los distintos módulos. Para empezar, se explican detalles acerca de la adaptación del sistema original ([6, 9, 10]) al nuevo hardware (software de sincronización de las cámaras y ajuste de los parámetros de adquisición de las imágenes), asi como la adaptación del sistema al sistema operativo GNU/Linux. Podemos encontrar también ciertos detalles de la implementación original del localizador de manos y puntas de los dedos. Finalmente se explica más extensamente la extracción de coordenadas en 3D del contorno de la mano. Recordemos que estas coordenadas (más concretamente las coordenadas en 3D de las puntas de los dedos y un punto en la palma de la mano) seran, junto con la orientación de la palma, los parámetros necesarios a la hora de calcular la postura de la mano. Para la extracción de las coordenadas 3D de un conjunto de puntos necesitaremos, básicamente, las coordenadas en 2D de dichos puntos, los parámetros intrínsecos y extrínsecos de las cámaras y la correspondencia entre los puntos de ambas imágenes. Estas necesidades estan cubiertas por el sistema original, pero la precisión del algoritmo de correspondencia entre puntos no era suficiente para nuestros objetivos. Pasemos a comentar ese algoritmo y el algoritmo desarrollado que lo sustituye.

En el sistema original, la extracción 3D se utiliza para situar aproximadamente la silueta de la mano en el espacio. Sin embargo, en nuestro sistema esas coordenadas seran utilizadas por un módulo posterior para extraer la postura completa de la mano, luego se requiere una precisión mayor. En el sistema original estas correspondencias son fijadas mediante un algoritmo iterativo (Iterative Closest Point, ICP [57]) que relaciona ambos contornos mediante una serie de transformaciones afines [1]. Estas transformaciones sirven, básicamente, para transformar conjuntos de puntos coplanares. Sin embargo los puntos del contorno de una mano distan normalmente de pertener al mismo plano. Nuestra propuesta es utilizar la calibración de las cámaras, más en concreto la geometría epipolar, para hallar las correspondencias entre los puntos. Para discernir entre posibles correspondencias se utilizó programación dinámica o "Dynamic Time Warping" [45]. Hay más información acerca del algoritmo usado en la sección 5.4.

Evaluación

El capítulo de evaluación relata los experimentos realizados con el sistema. Desgraciadamente, no se consiguió efectuar una estimación adecuada de la orientación de la palma de la mano, por lo que el sistema no pudo

evaluarse en conjunto.

El módulo de extracción de los ángulos que conforman el modelo de la mano fue evaluado con fines de depurado de código mediante un módulo de extracción de coordenadas de los puntos clave dados los ángulos de las articulaciones. Los resultados fueron satisfactorios, con pequeñas inexactitudes cuando los ángulos se acercaban a 0 y 90 grados.

El módulo de correspondencia entre puntos de las imágenes en estéreo fue evaluado con más profundidad. Para dicha evaluación se utilizaron distintas secuencias de complejidad variada, ejecutadas con y sin objetos (ver Figura 6.1). La primera evaluación consiste en una comparación visual de los resultados obtenidos mediante los distintos algoritmos (ICP y programación dinámica) y los resultados provinientes del marcado manual de dichos puntos. Tras esta evaluación, se procedió a estimar la densidad de probabilidad de error en la estimación de dichos puntos. Mediante esta medición, se comparo la exactitud de los distintos métodos en varias secuencias, así como la precisión en distintas coordenadas y otras mediciones.

El resultado principal de esta evaluación muestra que el algoritmo original, ICP, tiene una precisión mayor en secuencias donde los puntos del contorno de la mano son coplanares o casi coplanares, mientras que el algoritmo desarrollado muestra un comportamiento mejor en secuencias más complejas. Otros resultados adicionales se muestran en la sección 6 Cabe mencionar que el algoritmo basado en programación dinámica no fue optimizado para su ejecución en tiempo real, y por tanto es más lento que el algoritmo basado en ICP.

Conclusiones

Las aportaciones de este proyecto son básicamente dos: la aplicación de un nuevo algoritmo para la busqueda de correspondencias entre puntos del contorno de la mano provinientes de imágenes en estéreo, y el desarrollo de un modelo cinemático de la mano con una solución cerrada para los ángulos de las articulaciones dadas las posiciones de las puntas de los dedos y la posición y orientación de la palma de la mano.

El algoritmo de correspondencia entre puntos resulta en una mejora considerable en la extracción de coordenadas en 3D, dado que no depende de ningún modelo de transformación entre contornos. No obstante, este algoritmo debe ser optimizado para acortar su tiempo de ejecución.

El modelo cinemático de la mano nos proporciona un modo de extraer la configuración de la mano rápida. Al contrario que las soluciones basadas en bases de datos de posturas manuales, esta solución nos permite extraer cualquier tipo de configuración manual acorde al modelo de la mano utilizado.