Point Localization in Surfaces

Ricardo Jorge Pires da Rosa Fernandes
ricardo.r.fernandes@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal
November 2015

Abstract

The aim of this thesis is to locate a point or several points in a surface, using a Microsoft Kinect camera. Our research is focused on articulated objects, such a human body. The motivation for this thesis is simple but very useful. It is tracking skin lesions (points) in human body (surface). We propose a novel method called “Trilateration” of geodesics. Representing the points which we want to track by its distance of “anchor” points (easy points to track), we trilaterate those in order to get the location of the points, using a technique similar to GPS. At the end, we have estimated the location of the points in the surface in different time stamps. This method uses the concept of geodesic, which is invariant whatever pose of the object is. With this approach we solve the localization of point/points in a surface either for rigid or articulated objects. This is a powerful tool, since tracking in objects articulated objects is extremely difficult.

Keywords: Geodesics, Trilateration, Fast Marching, Tracking, Articulated objects.

1. Introduction

Articulated objects are being widely studied in computer vision because they are very common on the everyday life. This kind of objects are in general difficult to track, model, recognize, detect its pose and almost all the interesting things in computer vision.

The goal of this thesis is to locate a point in a surface. The surface is the boundary of an articulated object, the human body, and the point or points which we want to recover are skin lesions (moles). The skin has three layers (epidermis, dermis and subcutis) and in the top layer(epidermis) exists a type of cells called Melanocytes that undergo a malignant transformation produce a less common but far more deadly and aggressive cancer: malignant melanoma. So, becomes important to track the lesion’s location because either the doctors need to know which analysis belongs to each lesion, when the patient have more than one, or they need to know which mole has been tracked considering that the patient have several moles that are potentials candidates to melanoma.

Other interesting applications can also be considered e.g., hand tracking. The goal is to simultaneously track the fingertips and hand joints linked by an articulated model.

We propose a novel method to locate points in a surface called Geodesic Trilateration. This method is based on the fact that the distance (geodesic) between points, measured over the object’s surface is invariant to pose. So, representing each point by its distance to some anchor points, we can uniquely represent them. Our proposal can be divided in two main points: model learning where the output are geodesics and Trilateration where we locate the target point.

Since we are trying to identify a point in two different images (2D registration) or in two different surfaces (3D point set registration), the problem could be represent as a registration problem. There are several methods that try to solve either 2D or 3D alignment, which we will classify depending of the type of transformation: rigid or non-rigid.

Beginning with 2D rigid approaches, correlation-based methods compute cross-correlation to measure the degree of similarity between the reference image and sensed image [4]. Fourier methods as the name suggests, take advantage of Fourier transform properties. These methods differ from the previous because they search the optimal match using information in frequency domain.

In [4] is also explained Point Mapping based approaches. Until now, Point Mapping based are the first where type of misalignment is unknown. Those methods can be resumed by 3 steps: feature extraction from sensed image, correspondence between control points and features points of the reference image and 2D polynomial function determination.
using the matched features.

There are also methods that try to solve image registration when the transformation between images is non-rigid. Elastic model based methods are examples of methods where the image is modeled as a distortion of the deformation of an elastic material. The advantage of these methods is that they do not need a preliminary step in which features are matched.

Point set registration consists in finding the correspondence between two point sets and to recover the transformation that maps one point set to the second [2].

There are several developed methods that solve 3D rigid alignments. Iterative closest point (ICP) is an algorithm which minimizes the difference between two sets of points, estimating the rigid transformation, which aligns them. Since the correspondence between points in the two sets is unknown a recursive procedure is used [3]. After ICP has been proposed in 1992, other variants were proposed to improve the method [7]. ICP needs that initial pose of the two point sets are substantially close what is not possible always especially when transformation is non-rigid. So, improvements of ICP based on probabilistic approaches were developed [1] [6]. Robust Point Matching (RPM) was one of the methods, performing local search and assigning correspondences between two sets accordingly with some probability [8].

To overcome the limitations of the previous rigid methods, several non-rigid registration methods are introduced [5] [1]. Putting together, RPM and spline-based technique, Non-Rigid Point Matching (TPS-RPM) was proposed by Chui and Rangarajan [5]. It use thin-plate spline (TPS) as the parameterization of the non-rigid spatial mapping and the softassign for the correspondence [5]. In [2] is introduced a probabilistic method for point set registration called Coherent Point Drift (CPD) method. This method finds non-rigid transformation and the correspondence between two point sets without any prior assumption of transformation model [2]. Tsin and Kanade in [9] proposed an effective way to align intensity images: correlation-based approach. It uses a kernel correlation and a function of the point set entropy to estimate the transformation between the sets.

2. Point Localization Methods in Surfaces

2.1. “Trilateration” Approach

Trilateration is the name given to a process of estimation either the absolute or relative position, relying upon distance measurements only (normally associated to GPS). In GPS, when we want to estimate the location of a point on the Earth’s surface we use at least three static satellites (Fig. 1(a)) to measure the Euclidean distance from the point for each satellite. Then, GPS computes a sphere with center point in each satellite and radius the distance between the point and the satellite. With those distances, it finds the point that intersects all the spheres.

![Figure 1: Illustration of the Trilateration approach using in GPS.](image)

We use similar approach as in GPS but in surfaces (Fig. 1(b)), where the measured distances are not the Euclidean distances but geodesics instead. Geodesic is a minimum length curve between two points in a surface. It is a generalization of concept of “straight lines” to “curved spaces” and manifolds.

The localization method can be performed with the following steps:

1. Identify anchor points $y_i$ which are points that the system can easily track (playing the same role as satellites).
2. Choose set of target points $x_i$.
3. Compute geodesics from $x_i$ to $y_i$.
4. Estimate $x'_i$ in new image by intersecting geodesics.

Note that geodesics are computed by two distinct methods which we will explain later.

2.2. Computing Geodesics on Surfaces

Geodesics are minimum paths. Dijkstra algorithm (DM) and fast marching method (FMM) are two possible implementations to compute geodesic distances. These two algorithms were used to computed the distances between the mole and the anchors in two different approaches. Table 1 present a small overview of the two algorithms.

One of the differences between the methods are the input approximation of the surface. In Dijkstra's, the distances are computed along the graph which represents the surface. On the other hand, FMM uses a mesh, which represents better the surface, in order to compute geodesics. Other
<table>
<thead>
<tr>
<th>DM</th>
<th>FMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph</td>
<td>mesh</td>
</tr>
<tr>
<td>$O(</td>
<td>V</td>
</tr>
<tr>
<td>Restricted to nodes</td>
<td>Every point in the edges</td>
</tr>
<tr>
<td>discrete</td>
<td>continuous</td>
</tr>
<tr>
<td>$\mathbb{R}^2$</td>
<td>$\mathbb{R}^3$</td>
</tr>
</tbody>
</table>

Table 1: Comparison between Dijkstra’s method and Fast Marching method.

The difference is the complexity as well as the points were the algorithm can “walk”, i.e., Dijkstra’s is able to pass through the nodes, on the other hand, FMM can pass through every point in the edges of the mesh.

**Dijkstra’s Algorithm**

Dijkstra’s algorithm applies a recursive algorithm based on Dynamic programming. The input of the algorithm is a weighted undirected graph and the output is a distance map from the source point to all points, i.e., each node have the distance from the source to itself. It can compute the distance map updating successively, for each point $x$, the length $d(x)$ of the shortest path found so far between $x_0$ and $x$. At the beginning, it assign the source $x_0$ with $d(x_0) = 0$, all the other points with $d(x) = \infty$ and the set of vertices in $Q$ (set of unprocessed points). Then, the algorithm starts the iteration section until all nodes have been processed.

**Fast Marching**

FMM is a numerical procedure that solves a boundary value problem. Boundary value problem solution is a solution of differential equation which accomplish at the same time the boundary conditions.

The outline of the algorithm is the following. First, in initialization, we set all the distance from the source point $x_0$ to $\infty$ except from itself where the distance is set to zero. Beginning the loop:

1. Get the point with the smallest distance from the source;
2. Update all the triangles that share the point;
3. Tag as Close all neighbors of chosen point, that are not already Accepted; if the neighbor is in Far, remove it from that list and add it to the set Narrow Band;
4. Recompute the values of $T$ at all neighbors by a solving the quadratic equation, using only values of points that are Accepted.
5. Return to top of loop

2.3. Target Points Candidates

For each point $x_i'$ in the surface, we pick all the points whose distances $d_i$ to anchor points accomplish the following criteria:

$$|d_i - m_i| < \delta$$

wherein $m_i$ is the distance from target points to anchor $i$ (that was saved before). This will produce the shortest distance so far. If $x'$ has adjacent points, their distances to the source will be updated with the minimum distance between the distance $d(x')$ and $d(x) + w(x,x')$ which are the distance that had been saved in that node in early computations and the distance from the last path computed to $x'$, respectively. Processed $x'$, it must be removed from $Q$, and the process continues until the set $Q$ is empty. In this manner, each point is computed exactly once. In Figure 2 we present an example of Dijkstra’s algorithm.

![Dijkstra’s algorithm example](image)

Figure 2: Dijkstra’s algorithm example.
bands (Fig. 3), which we will be intersect (in analogy with GPS). This intersection is done using a vote scheme (constructing an histogram) where the output candidates are those which accomplish more times the criteria, i.e., the points which are present in more bands (red dots in Fig. 3).

To conclude, the estimated target point is estimated from the set of candidate points. We compute the mean of the 3D candidate points and pick the point which better approximate this mean, i.e., which the Euclidean distance between the mean and the point is smaller.

3. Dermatology Application

The system proposed for the thesis, is a very useful tool for dermatologists, whom want to monitor several skin lesions in a patient. If the appearance of the lesion changes this information is an important cue for the dermatologist. However, there are no automatic tools to monitor individual skin lesions along the time. We propose a method which automatically identifies the position on the body of either one mole or several ones which are being tracked by the dermatologist along time. So, let us explain all the work behind the proposed system.

3.1. Experimental Setup

The experimental setup proposed in this thesis is shown in Figure 4. This setup allows the acquisition of depth and rgb images of the patient which are registered by the system. The depth image allows the localization of points on the body surface while the rgb image provides an intuitive interface with the doctor.

The equipment considered is the following:

- Microsoft Kinect (1).
- Auxiliary rgb camera with better resolution than Kinect (2).
- Kinect support (3).
- Computer with graphical interface with the user (4).

Now let us explain the procedure we propose during the medical appointment. The procedure is different in the first session (model learning) and in the follow up sessions (tracking).

First session

- Medical doctor examines the patient and acquires dermoscopic images of suspicious lesions. Using his knowledge, the doctor inspects the patient’s moles in order to see if there are suspicious moles which could turn to melanomas.

Follow up sessions

- Medical doctor acquires images of the patient using the Kinect and RGB camera. The two images are aligned by the system. First, the patient should stand the arms in “psi” pose to calibrate the system, and then he can do other pose.
- Doctor identifies moles of interest on the RGB-D image and the system computes the geodesic from the moles to each joint.

At the end of the session, the system should have a 3D model of the patient’s body, with some labels on it which are the locations of the moles. The labels have attached to them rgb and dermoscopic images of the correspondent mole.

3.2. Implementation Details

Microsoft Kinect Sensor

Kinect is a device produced by Microsoft for xbox 360 gaming console. It has a image, audio and depth sensors attached, which detects movements, identify faces, and recognize speech of the players. It allows us to compute 3D point clouds of the scene using depth’s camera information.
Kinect’s depth camera can be modeled as a standard projective camera

$$\lambda \bar{x} = P \bar{X} \iff \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R] [t] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

(2)

where $X, Y, Z$ are the coordinates of a 3D point in the scene and $(u, v)$ are the coordinates of the projected point in the image. We assume that the scene points are measured in the Kinect coordinates system (depth camera reference frame) $(R = I, t = 0)$ and the matrix of intrinsic parameters $K$ is estimated by a calibration procedure. Since we know the depth $Z$ for each projected point $(u, v)$ we can retrieve the $X, Y$ coordinates as follows,

$$\begin{cases} \lambda u = f_x X + c_x Z \\ \lambda v = f_y Y + c_y Z \\ \lambda = Z \end{cases} \iff \begin{cases} X = \frac{Z(u - c_x)}{f_x} \\ Y = \frac{Z(v - c_y)}{f_y} \end{cases}$$

(3)

Registration Between RGB and Depth Images

The doctor should identify the position of the mole of interest in the RGB image and the system should determine its position in the 3D point cloud. This operation requires the calibration of two cameras. This means that we should estimate the matrices of intrinsic parameters $K_d, K_{rgb}$ and the rigid body transformation from the RGB coordinate system to the coordinate system associated to the depth camera

$$dX = R_{rgb} X + t$$

(4)

Anchor points

The anchor points are the body joints given by the Kinect software (OpenNI). A joint or articulation is the point where two or more bones meet. So, we determine the joints location in order to have well known points where we can apply the algorithm.

It should be stressed that joints are the union of two bone and hence OpenNI give us a point inside the body surface. So, what we did is project the point into body surface. We find the nearest point in the body surface and considered it to the computation.

Surface Representation

To estimate the initial position of the mole, we need to compute geodesics along the surface. Hence, we need to find an accurate representation of the surface in order to compute those paths. In this section, we will present two different representations based on graph and mesh, which lead us to different algorithms to compute geodesics.

A graph is a representation of a set of vertices where some pairs are connected by links. Depending on the direction of the links (edges) can be defined three types of graphs, directed, undirected or bi-directed. In this case, is convenient to ignore the direction of the link, so we choose an undirected graph.

![Figure 6: Undirected weighted graph construction.](image)

We picked the depth image and then, connect each pixel of the image $(i, j)$ to its eight immediate neighbors where the cost of each edge is the Euclidean distance between this points in the surface (Fig. 6).

The second representation considered was the mesh. This is a better approximation of the surface, comparing with the graph representation. A mesh is a representation of surfaces which has three fundamental components, vertices, edges and faces. The several vertices are connected by edges forming faces which can have different geometries and topologies. Normally, faces are triangles or quads. Here we will work with triangles.

Delaunay Triangulation was the name given of a method to triangulated points after Boris Delaunay did his work on this topic in 1934. We use this method to form the triangles of the mesh which is the input for the FMM. The aim of this method (Fig. 7) is that there are no vertices inside circumcircles (circle that passes through all three vertices) of any triangle in the triangulation. In this manner we avoid skinny triangles be maximizing

![Figure 5: Joint transformations definition (with number labeling).](image)
the minimum angle of all the angles of the triangles in the triangulation.

Target Points Candidates

Candidates estimation is done essentially applying two steps. First a voting scheme is applied, retrieving points that are consistent with the reference distance $d_0^i$. Then, the most voted points are determined.

1. Voting: for each node $x$ in the surface, decide if it is compatible or not with the reference distance $d_0^i$

   $v_i(x) = \begin{cases} 
   1, & \text{if } |d_i(x) - d_0^i| < \delta \\
   0, & \text{otherwise} 
   \end{cases}$

   wherein $d_0^i$ is the distance from the lesion to joint $i$, $d_i(x)$ is the distance from point $x$ to anchor $i$ and $\delta$ is a threshold.

2. Optimization: determine the most voted points

   $\hat{x} = \arg \max_x \sum_i v_i(x)$

To conclude the algorithm, we need to choose a point between all the candidate points. We organize the points in clusters and extract the bigger one. After that, we compute the mean of the 3D points and pick the closest point relatively this mean, i.e., which the Euclidean distance between the mean and the point is smaller.

3.3. Experimental Results

Geodesic paths

In this section we will present the output paths from the two methods (Fast Marching Method (FMM) and Dijkstra’s method (DM)) for a set of different cases. First consider a planar scene. We wish to estimate the geodesic curves between two points in a plane. The results are shown in Figures 8(a), synthetic plane produced in Matlab, and 8(c), real data acquired by kinect. To conclude, is shown in Figure 8(b) a sample of the patient’s body with the paths between two points.

The plane is the simplest experience that can be done in order to test the methods. Inherent to the definition of geodesics, in the plane it becomes straight lines, so Figure 8(a) shows that FMM is a better approximation than DM. Figure 8(a) gives us two important informations. First, the output path of FMM is shorter than Dijkstra’s method. Second, FMM achieves this output by “walking” not only through the vertices as the Dijkstra’s method does, but going also through the faces of the mesh.

Statistical Evaluation

In order to characterize the system performance we acquired a dataset of depth and RGB images of the human body. We have performed 30 pairs of images for two different poses (Fig. 9). This corresponds to 60 acquisition sessions of the same person. First we will present a study of the variation of the distances between joints and the target lesion for several time instants.

Figure 9 shows the geodesics from the 15 body joints to the mole in 3D patient’s body. We also show histograms of the error $e = d_i - \bar{d}_i$, where $d_i$ is the distance from the joint $i$ to the lesion and $\bar{d}_i$ the mean of the 30 distances of the joint $i$. Histograms are classified as bad, reasonable and good and they are represented with three different color (red, orange and green).

Figure 10 show the variance of the distances associated to all pairs of joints i.e., we computed the geodesic distances between all pairs of joints, for each of the 30 experiments and display the variance.

The last experience was to compare the performance of the two methods, using the same thirty
The results are actually better in the overall when we apply the fast marching method (Table 2). However, besides Dijkstra’s method does not give a good approximation of geodesic paths, the error is quite near to fast marching method.

Table 2: Comparison between the mean error of Dijkstra’s method and Fast Marching method.

<table>
<thead>
<tr>
<th>Pose</th>
<th>DM</th>
<th>FMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.023</td>
<td>0.019</td>
</tr>
<tr>
<td>2</td>
<td>0.036</td>
<td>0.031</td>
</tr>
</tbody>
</table>

Figure 11: Euclidean error between the estimation and the right location of the skin lesion for FMM (blue) and DM (red) for pose 1.

To evaluate the computational effort associated to both methods, we computed the CPU time required to estimate the distance matrix by both methods using 3D representation with 60000 points. We concluded that Dijkstra algorithm is fast, computing the distance in 0.065s and FMM in 1.75s which is about an order of magnitude, as expected.

4. Hand Tracking Application

Hand tracking is an application where we wish to track joints and fingertips. This is a very difficult task because joints and fingertips are featureless points, i.e., they are smooth points, without texture. We will do it by tracking an easy pattern and using it as anchors for trilateration.

4.1. Hand Modeling

Hand Model

The model in Figure 12, shows the marker (arUco) and the joints (circles) which will be tracked. It has two types of distances for different purposes. The distances from joints to anchors are need to compute the candidate points for each joint after trilateration. On the other hand, distances between joints are used to refine the location of the estimated points. So, in Figure 12 is also shown these two sets of distances:

1. $d_{A_{ij}}$ which is the distance from the starting point $i$ to the anchor point $j$.
2. $d_{P_{mn}}$ which is the distance from the target point $m$ to target point $n$. 

Figure 9: Histograms of the distance between the skin lesion to each joint (pose 1), computed by FMM.

Figure 10: Variance of the distances between joints for pose 1 computed by DM (left) and FMM (right).
Detecting Hand Configuration

Tracking hand features is a subject that have some issues. Dramatic changes in the model distances (mainly in $d_B^{mn}$ distances) due to different hand configurations (Fig. 13) can occur. So, we need multiple hand models, one for each configuration of the hand. The problem is predict the hand configuration in order to apply the correspondent model, i.e., with a new input image, we should know if the hand are either in configuration A, B, C or D (Fig. 13), for example. This problem is solved using an inductive inference method: decision tree learning.

Different hand configurations was the bigger problem to solve in this application. However, others problems were solved with the algorithm explained in Section 4.2. In sum, the tracker overcomes the following issues:

1. **Hand featureless points.** The joints and fingertips are points which cannot be extracted any type of features.

2. **Occlusions.** When some joints are occluded, the algorithm retrieve the joints that are visible.

3. **Several hand configurations.** The algorithm predicts the configuration, applying the correspondent hand model.

4. **Large hand displacements.** The tracker does not lose the hand, retrieving the 3D location of the joints and fingertips.

4.2. Tracking Multiple Hand Configuration

In this section we will present a novel approach called Learning hand configuration algorithm. It incorporates, the Trilateration of geodesics algorithm and a decision tree.

Learning Hand Configuration Algorithm

- **Hand configuration training:**
  - **Trilateration of geodesics** are computed in labeled images and the feature vector $v_i$ is extracted from that, forming by distances between joints.
  - Build a **decision tree** with the sample data, trying to use a diversity of configurations and different cases for different configurations (example in Fig. 14).
  - Compute the **models** for different configurations.

- **Estimation of point location:**
  - **Predict the configuration** of the new image based on the decision tree which was built (example in Fig. 14).
  - Perform the **Trilateration of geodesics** with the correspondent model of the configuration that was predicted.
  - Build an **histogram** and find the candidate points for each target point.
  - Perform the **refinement of point localization** with the model information, if confusion problem occurs (candidate points of a joint of one finger appears in other fingers).

4.3. Implementation Details

**Anchor Points**

In this application, the anchor points are the corners of the ArUco marker. The marker is pasted on the hand with glue in order to be smooth.
enough to not interfere in the computation of geodesic paths. ArUco is a library which uses black and white coded markers, which each of one has its unique code. It analyses rectangles, and decides whether the marker is reliable or not to be a marker. The code is detected, and if the code is a valid one, the rectangle is considered as a marker, otherwise the rectangle is thrown away.

Refinement of Point Localization

The refinement algorithm explained in this section is needed because of the problem named this candidate’s confusion problem, where the candidate points for one joint or fingertip of one finger are in other finger. This leads to wrong target points estimation. The aim of the algorithm is to minimize the distance between the current centroid and the others computing the cost function

\[ f = ||C'_t - d_B||_1 \] (7)

wherein \( d_B \) is the model distances from each target to all other target points and \( C'_t \) is the matrix of distances from all candidate points of the joint which we are trying to minimize to all centroids that are fixed.

4.4. Experimental Results

Large Displacements Data Set

The first results were achieved moving the hand along kinect’s covered area with the configuration A shown in Figure 13 and estimating the 3D location of joints and fingertips labeled in Figure 12.

In Figure 15, we have 4 sample images where we can see that the tracker retrieves the location of the joints (red dots) even when the hand has a large displacement between frames.

In Figure 16 we present a mesh with the error \( e_b = d^0_{mn} - d^i_{mn} \), where \( d^0_{mn} \) is the distance (geodesic) between joint \( m \) and \( n \) in the reference frame and \( d^i_{mn} \) is the distance (geodesic) between joint \( m \) and \( n \) in the frame \( i \) with

\[ i = 1, 2, 3, \ldots, 162. \]

The error \( e_b \) is small almost in every frames (between 6 and 8 mm) however, there are two frames where the error is quiet bigger (frames 58 and 113). These frames are outliers.

Occlusions

In Figure 17 we present images where some joints and fingertips are occluded. The images show that Trilateration of geodesics solves the occlusion problem.

Analysis of the Classifier

Decisions trees were the solution for the different hand’s configurations problem. The correction algorithm must be applied, using a decision tree to predict the input image configuration.

We built an example of decision tree with 3 different configurations. Each of those had 300 sample images where the features were the geodesics between all joints/fingertips. The configuration A in Figure 13 was used to save the model.
Then, geodesic trilateration was computed and the features extracted. Note that the output decision tree has in all edge the geodesic between fingertips and joints, showing that those distances are good attributes. The output confusion matrix for 118 sample images was built, showing that prediction only failed for one case.

**Manipulation**

In this section we show 4 sample images, of the manipulation of a bottle. In Figure 18, the red dots are the estimate points for joints and fingertips. We can see that tracker works quite well, although sometimes it does not recover some visible joints because the error for the model is higher than a threshold.

![Figure 18: Output images of manipulation, with estimated location of joints (red dots).](image)

5. Conclusions and Future Work

The work described in this dissertation aims to figure out a manner of developing a system which could locate skin lesions along time. The system must be able to identify the location of several lesion, in different time instants.

Dermatology application was important to perform a comparison between the methods and concluding about practical issues when a real system is implemented. First we conclude that DM cannot approximate geodesic paths and that FMM estimate the points in the surface (human body) with less error. However, besides Dijkstra’s method does not approximate well geodesics, the estimation error is not far from FMM. Moreover, DM is faster than FMM. This leads us to think in a trade-off between accuracy and computational time. Furthermore, we realize that there are better patient poses than others and the anchor points which will be used can be chosen carefully because there are more accurately tracked joints than others.

In hand tracking, some of the issues verified for the human body happens more frequently. Those problems were solved using learning, optimization and others techniques. ArUco marker used in the implementation is not good enough for this application. However, besides that, we had a very good results on problems as occlusions, different hand configurations and manipulation.

We conclude that the sensor have huge influence in the results. On the one hand, a good calibration is very important. On the other hand, is important that sensor has not problems estimating the depth map, which does not happen with kinect.

Regarding future work, the first obvious think is improve the computational time of the method in both applications. Then, would be important to test other sensor whose error in depth map is smaller than kinect, for example the next generation of kinect, kinect2.

References


