Fingerpointer - Pointing Gestures for Collaborative Robots

Sara Cordo, Alexandre Bernardino and Mário Rui Gomes
Universidade de Lisboa, Instituto Superior Técnico

Abstract. With the increase of electronic gadgets that allow more natural interactions, there is a growing need to improve the use of these capabilities. The iCub, is a widely used humanoid robot, and our aim is to train it to identify if a user is pointing to something. This is a very useful ability, since it allows gestural communication with the robot, enabling it to execute joint tasks, such as picking up objects, etc.

We divide it into 5 parts: 1) Video Synchronization, 2) Hand Detection, 3) Hand Segmentation, 4) Hand Gesture Classification, 5) Finger Direction and Orientation Detection.

We study, test and evaluate several techniques and settings for each part, and we present our proposed solution for the problem as a whole. The best solution found was not as robust as desired, specially in detecting the correct hand gesture, but achieved 74% accuracy in correctly detecting the fingertip in the images identified as containing the pointing gesture, and calculating the respective pointing vector.

Keywords: Gesture Recognition, Hand Detection, iCub Interaction, Kinect, Pointing Direction and Orientation Detection

1 Introduction

The Kinect [2,15] is a motion sensing device launched by Microsoft in 2010, capable of locating a person, map a room in 3D, identify gestures, etc. The creation of open source drivers [3] for it, allowed research in other areas of robotics: robots that respond to human gestures (gesture recognition) [2] or interact with the environment (e.g. picking up objects), etc.

The iCub [9] is an open-source humanoid robot, developed in Europe, that imitates the size and form of a 3.5 year old child. Its design was motivated by the embodied cognition hypothesis, where human-like manipulation plays a vital role in developing human cognition. The goal of this project is to train it to detect if the user is pointing, and in what direction he/she is pointing.

Detecting the hand or pointing gesture from an image is not a new problem. There is lot of work in gesture recognition, but not much research in fingertip detection. In this work we want to do something robust, that locates and extracts the hand automatically, independently of the initial position, and that detects the
fingertip. To achieve this we propose the solution in Fig. 1. Our real challenge is to combine locating the hand automatically with gesture recognition, and identifying the direction in which the finger is pointing.

Fig. 1: Diagram of our proposed solution to detect the finger pointing vector (direction and orientation)

To increase the segmentation accuracy we chose to use the skin detection, since the nearest point to the camera was not accurate enough. We selected the best skin option for our hand segmentation. To classify the pointing hand gesture, we constructed a hand classifier, and as we had limited time we chose not to improve it, we focused on our main goal: to detect the finger pointing vector. We tested the accuracy of the finger pointing vector, using the manual classification ground truth after segmentation. Where we correctly detected the fingertip, that we will paint in blue, with 74% accuracy, and return the pointing vector.

2 Contributions

One of our main goals, as described above, was to locate the hand automatically, without knowing an initial position. We achieved that by using skin detection techniques combined with using the nearest point to find the hand. The main idea of our proposal is to first detect the skin pixels by defining the skin color thresholds manually or dynamically. In the non-dynamic option we define manually these thresholds; Then, after detecting the skin points, the hand is highly likely to be the closest skin point, and thus we use the depth information and the detected skin points to find the skin point that is closest to the camera. In the dynamic option we find the face first with face detection, and use the nose as a clean sample of the skin, this allows us to be independent of the skin color of the person; then use this skin sample to detect similar pixels in the image, using a grey scale transformation or the new hue information.

Also, with the intention to achieve a more robust method, we proposed a new hue transformation, that centers the skin color in the hue scale, and therefore
allows us to define a better interval for this color and identify better the other skin pixels in the image. We also tried a new dynamic formula (“Maxmin”) to automatically fine tune the threshold used for skin detection.

We also achieved our main goal, which was to detect if the user is pointing, and in which direction, by using classification trees to identify the gesture and using the eigenvectors and eigenvalues in the images classified as pointing to detect the fingertips and orientations. We proposed two different methods to detect the fingertip and determine the pointing orientation: one by counting the points around the pointing axis extremities, and another by projecting all points into that axis and building a histogram of it.

3 Related Works

Lin et al. [8] divided the problem of gesture recognition into 3 sub-problems. In our work we divided the problem into 5 parts: 1) Video Synchronization, 2) Hand Detection, 3) Hand Segmentation, 4) Hand Classification, 5) Finger Direction and Orientation Detection.

1. To segment the hand [8, 5], it is necessary to know its position in each frame. Most of the existing work requires an initial position to begin with. One way is to fix a position where the user must place his hand. Ghotkar et al. [5] experimented with the same methods as Lin et al. They also used a mouse click to identify the center of the hand, in the initial frames, and a hand tracking algorithm in the remaining ones (by using skin detection). Elgammal et al. [4] as an alternative, proposed to detect the skin on a colored image. Pavlovic et al. [10] compared many hand detection methods. Yogarajah et al. [14] proposed a novel dynamic threshold approach for colored images, that we have implemented.

2. There exist several Hand segmentation algorithms, but almost all are based on segmenting regions. Lin et al. [8] converted the depth image into a point cloud. Then segmented the hand isolating it in a sphere combined with the Connected Components algorithm, and Jing et al. [7] used the depth value of the hand joint, where the relevant user and hand is the one the closest to the Kinect sensor. Ghotkar et al. [5] after getting the central point of the hand, applied morphological operations and a transversal algorithm, to segment it.

3. To classify a hand gesture [10], Lin et al. [8] constructed a feature vector with 6 values (3 invariant moments and 3 eigenvalues). Pavlovic et al. [10] explored to recover, classify, interpret the parameters from the features, and partition the data for training the classifiers (e.g. having few examples of one class, will make it harder to detect the gestures of that class).

4. Sadjadi et al. [12, 8] establishes the generalizations of 3D geometrical moments \(J_i\) to recognize three-dimensional objects independent of their size, position, orientation, and gave some intuitive geometrical meaning to them.

5. We need to identify the Finger pointing direction and orientation. Jing et al. [7] compared the fingertips with the hand joints (hand joint is always
in the opposite direction of the fingertips). They used a dynamic model, Kalman filter and constructed a virtual touch screen in front of the Kinect sensor, the intersection determines the finger pointing direction. Pavlovic et al. [10] explored how to recover the fingertips, the most effective solution is using gloves and markers extraction by color histogram-based techniques. The other way is to use the curvature of the fingertips or an accumulative displacement torque approach.

4 Implementation

We divided the work in 5 parts, and used ROS (Robot Operating System, an open source software framework) to record the videos (RGB and Depth, their camera parameters and timestamps) from the Kinect into a rosbag file\(^1\).

To successfully segment the hand, during the recordings, we set up some requirements: The user needs to be facing, the Kinect sensor, (since we use a face detector); Avoid occlusion of the hand; Keep a minimum distance of 1 meter from the sensor and stay near enough to capture a large enough hand region; Record in a 8-bit int RGB and a 32-bit Depth float format to allow a correct color detection and depth segmentation.

Since the recorded videos were unsynchronized, we needed to pair each RGB image with the depth image that had the smallest timestamps “drift”.

4.1 Hand detection by skin color

After trying several methods, we finally implemented the article [14] and improved on it. This article has two parts, the non-dynamic, where you manually define the thresholds to apply to all the frames, and the dynamic part where the thresholds are calculated by a statistical formula. They used only one grey scale threshold, where we introduced 3 thresholds. We also introduced the alternative Hue instead of the grey scale they use, and a min-max as an alternative to the statistical formula they used. The dynamic option needs the user to face the Kinect sensor to detect the face.

We choose the HSV color space, because the color cluster is more compact, separates luminance from chrominance and is more intuitive. We also implemented a grey scale (based on article [14]), and to achieve better results, combined these two color spaces. We used Matlab Computer Vision System Toolbox to detect the face, extract the skin sample from the nose, and compared this with the rest of the image to detect the skin.

Modified implementation of article [14]. We converted the image to a grey scale form, based on its luminance \((I)\) (\(\times\) represents matrix multiplication):

\[
I(x) = \begin{bmatrix} r(x) \\ g(x) \\ b(x) \end{bmatrix} \times \hat{\alpha}, \quad x \in 1, ..., n, \quad \hat{\alpha} = \begin{bmatrix} 0.298936021293775390 \\ 0.587043074451121360 \\ 0.140209042551032500 \end{bmatrix}
\]

\(^1\) Rosbag command-line: [Online; accessed 2014-Dec-20]
\[ e(x) = I(x) - \hat{I}(x), \quad \text{where} \quad \hat{I}(x) = \max (g(x), b(x)), \quad x \in 1, \ldots, n \quad (2) \]

Now if \( e(x) \in \) error range, then it is skin. The error range can be defined manually or dynamically, based on a normal distribution \([14]\) of a skin sample. Then we estimate the mean and standard deviation \((\hat{\mu}, \hat{\sigma})\) with the Matlab function \(\text{normfit}\) in \([\hat{\mu}, \hat{\sigma}] = \text{normfit}(e(x))\):

\[
\text{error range} = [\hat{\mu} - w_1 * \hat{\sigma}, \hat{\mu} + w_2 * \hat{\sigma}], \quad w_1, w_2 \in \{0.1, 1, 1.5, 2, 3, 4\} \quad (3)
\]

We created an alternative based on the skin sample. The minimum and maximum value of \(e(x)\) fine-tuned by multiplying the weights: \(w_1\) and \(w_2\):

\[
\text{error range} = [w_1 * \min e(x), w_2 * \max e(x)], \quad w_1, w_2 \in \{0.1, 1, 1.5, 2, 3, 4\} \quad (4)
\]

As a last step, we used erode and dilate filters.

**Modified new hue implementation.** We also defined a new hue metric, corresponding to a 180° degree rotation of the normal one. Our intention was to concentrate the skin range in the center of the scale:

\[
\text{newhue} = \begin{cases} 
\text{hue} - 0.5 & \text{if} \quad \text{hue} \geq 0.5 \\
\text{hue} + 0.5 & \text{if} \quad \text{hue} < 0.5 
\end{cases}
\quad (5)
\]

### 4.2 Hand segmentation from depth image

We take the skin detection mask and the Euclidean distance in real-world co-ordinates \([6]\) to calculate the nearest point to the camera. We use this point to build a bounding sphere and drop everything outside it. In alternative we run a seed-growing algorithm \([8]\), using this point as the starting point, and determine if the neighbours belong to the same region. To compensate for the drawbacks of each of these methods, we used a combination of both.

### 4.3 Hand Classification

Now we need to classify the hand (if existent in the frame) into: open, closed, pointing, or unable to determine. As a classifying feature that needed to be location invariant, we selected the 2nd and 3rd order 3D central moments \([12, 8]\). And since we also needed the 3rd order moment to be rotation invariant, we changed the coordinates to the hand referential, before calculating them.

\[
\begin{bmatrix} V, D \end{bmatrix} = \text{eig} \begin{bmatrix} \mu_{200} & \mu_{110} & \mu_{101} \\
\mu_{110} & \mu_{020} & \mu_{011} \\
\mu_{101} & \mu_{011} & \mu_{002} \end{bmatrix}, \quad D_{3x3} = \begin{bmatrix} \lambda_1 & 0 & 0 \\
0 & \lambda_2 & 0 \\
0 & 0 & \lambda_3 \end{bmatrix} \quad (6)
\]

\[
(x, y, z)_H = V \times ((x, y, z)_C - (x_c, y_c, z_c)), \quad (7)
\]

Where \((x_c, y_c, z_c)\) represents coordinates of the center of the hand, \((x, y, z)_H\) represents the coordinates in the hand referential, \((x, y, z)_C\) represents coordinates in the camera referential, \(\lambda_i\) are the eigenvalues and \(V_{3x3}\) the eigenvectors.
We used the Binary Classification Tree [11, pp. 702–704], separated the data into training and testing data, and trained it to classify the 5 classes. We tested different combination of features: 1) 2nd order moments; 2) 2nd and 3rd order moments. To increasing its accuracy [11, pp. 706–707], we pruned it, on the average best level obtained by cross-validation. Initially we used one bag with all the 5 gestures, then we added two extra bags to equalize the percentage of hand gestures of the same type, resulting in 6 tree variations.

4.4 Identifying the finger pointing vector

Our intuition behind the fingertip detection is that there are more points at the wrist than at the fingertip. In the first attempt we count the points at the extremities, in the second we try to establish if the points increase or decrease with the axis.

We use the pointing frames, eigenvectors \( \vec{v} \) and eigenvalue \( \lambda \) (the largest value belongs to the finger pointing axis direction) and \( p_i = (x_i, y_i, z_i) = 2 \sqrt{\lambda} \times \vec{v} \pm (x_c, y_c, z_c) \) where \( i = \{1, 2\} \) to give us the finger extremities, [1, pp. 121–122], therefore \( p_1, p_2 \) are extremities of the pointing vector, \( (x_c, y_c, z_c) \) represents coordinates of the center of the hand and \( \lambda = \text{eigenvalue} \) and \( \vec{v} \) = pointing direction vector.

Now we have two options: Detecting fingertips by counting the points around the pointing axis extremities, where we extract them with different radius and the smallest amount belongs to the fingertip; or projections of the points [1, pp. 670–671] on the finger pointing axis and a histogram, of 4 bins and calculated \( c_{2i} = |\text{bin}_i - \text{bin}_{i+1}| \), where \( i = \{0, 1, 3, 4\} \). The maximum of the differences \( \text{fingertip} = \max(c_i, c_{i+1}) \), where \( i \) is the bin, gives us the orientation of the vector, Fig. 5.

5 Results

To do a correct evaluation, we used the metrics Precision (PPV), Recall (TPR), True Negative Rate (SPC), Negative Predictive Value (NPV), Accuracy (ACC) and F1-score. For the skin we used existing segmented data sets [13]. For the rest we had to manually classify all the data.

5.1 Skin detection

We used several parameters, combined in 6 main algorithms: Fix [14], Dist-Norm [14] (eq 3), MaxMin [14] (eq 4), FixHue (eq 5), DistNormHue (eq 3 and 5), MaxMinHue (eq 4 and 5). The manual calibration of the threshold varies according to the light conditions and skin tone. We attempted to maxime the \( F1\)-score metric, since it represents a balance of Precision with the Recall, Table 1, 2 and 3. Individually, in this skin detection problem, Single-weighed dynamic variations with lower weights achieved the best \( F1\)-score. The non-dynamic approaches also get more than 50\% \( F1\)-score. Now, the problem is to select the best one from the 15 variations that results in best segmentation.

Table 1: Non-Dynamic skin detection

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>TN</th>
<th>FP</th>
<th>FN</th>
<th>PPV</th>
<th>TPR</th>
<th>SPC</th>
<th>NPV</th>
<th>ACC</th>
<th>F1-score</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>FixHue</td>
<td>2611296</td>
<td>1750117</td>
<td>240296</td>
<td>42.84%</td>
<td>84.52%</td>
<td>59.87%</td>
<td>91.57%</td>
<td>66.34%</td>
<td>56.86%</td>
<td>15.48%</td>
<td></td>
</tr>
<tr>
<td>Fix</td>
<td>233262</td>
<td>36.11%</td>
<td>84.97%</td>
<td>65.61%</td>
<td>50.69%</td>
<td>15.03%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Double-weighted dynamic skin detection

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>TN</th>
<th>FP</th>
<th>FN</th>
<th>PPV</th>
<th>TPR</th>
<th>SPC</th>
<th>NPV</th>
<th>ACC</th>
<th>F1-score</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxMinHue</td>
<td>595317</td>
<td>4300574</td>
<td>68839</td>
<td>96657</td>
<td>50.73%</td>
<td>38.36%</td>
<td>98.61%</td>
<td>51.80%</td>
<td>82.79%</td>
<td>93.99%</td>
<td>61.64%</td>
</tr>
<tr>
<td>distNorm</td>
<td>755202</td>
<td>3636628</td>
<td>724785</td>
<td>979692</td>
<td>51.16%</td>
<td>48.92%</td>
<td>83.38%</td>
<td>82.10%</td>
<td>74.34%</td>
<td>50.02%</td>
<td>51.06%</td>
</tr>
<tr>
<td>MaxMin</td>
<td>912241</td>
<td>2958481</td>
<td>1406572</td>
<td>639653</td>
<td>35.34%</td>
<td>58.78%</td>
<td>67.75%</td>
<td>82.20%</td>
<td>65.40%</td>
<td>47.14%</td>
<td>41.22%</td>
</tr>
<tr>
<td>MaxMinHue</td>
<td>1530939</td>
<td>893636</td>
<td>355017</td>
<td>21891</td>
<td>30.09%</td>
<td>98.60%</td>
<td>18.40%</td>
<td>97.37%</td>
<td>39.51%</td>
<td>46.11%</td>
<td>1.40%</td>
</tr>
<tr>
<td>distNorm</td>
<td>1025103</td>
<td>2431913</td>
<td>1929500</td>
<td>526791</td>
<td>34.70%</td>
<td>66.05%</td>
<td>55.76%</td>
<td>82.20%</td>
<td>58.46%</td>
<td>45.49%</td>
<td>33.95%</td>
</tr>
<tr>
<td>DistNormHue</td>
<td>1543348</td>
<td>475669</td>
<td>3885714</td>
<td>8546</td>
<td>28.43%</td>
<td>90.45%</td>
<td>10.91%</td>
<td>98.24%</td>
<td>34.14%</td>
<td>44.22%</td>
<td>0.55%</td>
</tr>
</tbody>
</table>

Table 3: Single-weighted dynamic skin detection

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>TN</th>
<th>FP</th>
<th>FN</th>
<th>PPV</th>
<th>TPR</th>
<th>SPC</th>
<th>NPV</th>
<th>ACC</th>
<th>F1-score</th>
<th>FNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxMinHue</td>
<td>1339894</td>
<td>3423133</td>
<td>938280</td>
<td>212945</td>
<td>68.80%</td>
<td>86.28%</td>
<td>78.49%</td>
<td>94.14%</td>
<td>80.53%</td>
<td>69.94%</td>
<td>13.72%</td>
</tr>
<tr>
<td>DistNormHue</td>
<td>692471</td>
<td>3840503</td>
<td>520910</td>
<td>589423</td>
<td>64.88%</td>
<td>62.02%</td>
<td>88.06%</td>
<td>86.99%</td>
<td>81.22%</td>
<td>63.42%</td>
<td>37.66%</td>
</tr>
<tr>
<td>distNorm</td>
<td>849479</td>
<td>3889661</td>
<td>471752</td>
<td>702415</td>
<td>64.29%</td>
<td>54.74%</td>
<td>89.18%</td>
<td>84.70%</td>
<td>80.14%</td>
<td>59.13%</td>
<td>45.26%</td>
</tr>
<tr>
<td>MaxMin</td>
<td>1288660</td>
<td>2770999</td>
<td>1590414</td>
<td>265234</td>
<td>44.72%</td>
<td>82.91%</td>
<td>63.03%</td>
<td>91.26%</td>
<td>68.62%</td>
<td>58.10%</td>
<td>17.09%</td>
</tr>
<tr>
<td>DistNormHue</td>
<td>1397561</td>
<td>2351527</td>
<td>2100141</td>
<td>154283</td>
<td>39.96%</td>
<td>60.06%</td>
<td>51.85%</td>
<td>93.61%</td>
<td>61.88%</td>
<td>55.35%</td>
<td>9.94%</td>
</tr>
<tr>
<td>DistNorm</td>
<td>2444362</td>
<td>1903865</td>
<td>2457548</td>
<td>107332</td>
<td>37.02%</td>
<td>93.07%</td>
<td>43.65%</td>
<td>94.65%</td>
<td>56.62%</td>
<td>52.97%</td>
<td>6.93%</td>
</tr>
</tbody>
</table>

5.2 Hand segmentation

We tried to segment the hand by using three different methods: Isolating the hand in a sphere (ISphere); Using the Seed-growing algorithm (Seed-grow); Combination of both (Sphere-seed-grow). To evaluate the best skin detection options for the hand segmentation, we constructed a testing video by selecting 15 random frames from each bag, and ran the the best skin detections options of the previous section, using the bound sphere (radius = 0.14 m), Table 4.

Table 4: Hand Segmentation with different skin options

<table>
<thead>
<tr>
<th>skin option used</th>
<th>w1</th>
<th>w2</th>
<th>TP</th>
<th>TN</th>
<th>FP</th>
<th>FN</th>
<th>TPR</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>FixHue</td>
<td>0</td>
<td>0</td>
<td>340</td>
<td>1</td>
<td>0</td>
<td>109</td>
<td>75.72%</td>
<td>86.19%</td>
</tr>
<tr>
<td>DistNormHue</td>
<td>0.1</td>
<td>1.5</td>
<td>340</td>
<td>1</td>
<td>0</td>
<td>109</td>
<td>75.72%</td>
<td>86.19%</td>
</tr>
<tr>
<td>Fix</td>
<td>0</td>
<td>0</td>
<td>330</td>
<td>1</td>
<td>0</td>
<td>119</td>
<td>73.50%</td>
<td>84.72%</td>
</tr>
<tr>
<td>DistNormHue</td>
<td>1</td>
<td>3</td>
<td>326</td>
<td>0</td>
<td>0</td>
<td>123</td>
<td>72.61%</td>
<td>84.13%</td>
</tr>
<tr>
<td>DistNorm</td>
<td>0.1</td>
<td>1.5</td>
<td>320</td>
<td>1</td>
<td>0</td>
<td>129</td>
<td>71.27%</td>
<td>83.22%</td>
</tr>
<tr>
<td>DistNormHue</td>
<td>1</td>
<td>3</td>
<td>318</td>
<td>1</td>
<td>0</td>
<td>131</td>
<td>70.82%</td>
<td>82.92%</td>
</tr>
</tbody>
</table>

We want to measure how many of the frames had the hand segmented correctly, therefore maximize TPR (or recall). We also evaluated the algorithm without the skin detection (“all skin”), and achieved 30.73% TPR, which shows clearly that the closest point isn’t always a point in the hand, and that correct skin detection is essential.
We grouped the best TPR results, highlighted in blue (all above 70%). We have FixHue and DistNormHue (both with 75.72%) and Fix with 73.50%, but since in our work we already used the Fix as a baseline, all the necessary data for the evaluation was manually classified, and the variation is less then 3%, we decided to continue the project with Fix skin option.

The global efficiency of ISphere with the skin detection in all our videos is 79.84%. All of the segmentation methods have their drawbacks. ISphere ends up capturing part of the non-wanted background and/or other body parts, Fig. 2. The Seed-grow, besides being slow, also ends up selecting an excessively big region as the hand, Fig. 3. And in the combination (Sphere-seed-grow), the ISphere limited Seed-grow to a certain region, segmenting only the continuous region. This strategy did not work very well, since our depth videos tend to have flaws interrupting the continuity of the finger and the hand, Fig. 4.

![Fig. 2: RGB image, drawbacks of the hand segmentation by isolating in a sphere](image1)

![Fig. 3: RGB image, drawbacks of hand segmentation by seed-growing algorithm](image2)

(a) RGB image before Seed-grow (b) Flaws in white (NaN values) (c) RGB image after Seed-grow

![Fig. 4: Flaws in the recording of the depth image](image3)

5.3 Hand Classification

We used 6 variations for the Classification Trees: different features, amount of data and pruning strategy, as described earlier. For the optimized tree, we used the average of 20 calculations (pruning strategy section 4.3). We used 30 data sets divided in 1 or 3 training sets and 29 or 27 testing sets.

We also experimented with different types of classification data as ground truth (that is, classified manually) to construct the training trees: Before hand segmentation (“Classification variant Original Data”); After hand segmentation without excluding the flawed frames (“Classification variant Segmented Data”);
and after hand segmentation excluding the flawed frames ("Classification variant Segmented Data without Flaws") and choose the best Classification Tree, in Tables 5.

Table 5: Assessment Classification data used to construct the Classification Trees

<table>
<thead>
<tr>
<th>Classification variant</th>
<th>Original Data</th>
<th>Segmented Data without Flaws</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 and 3 opt tree</td>
<td>6433 37.81% 25.39% 36.46% 79.27% 29.94% 35.81% 37.81% 37.81%</td>
<td>6433 37.81% 25.39% 36.46% 79.27% 29.94% 35.81% 37.81% 37.81%</td>
</tr>
<tr>
<td>2 opt tree</td>
<td>4881 28.69% 18.69% 28.54% 76.23% 22.59% 28.69% 28.69% 28.69%</td>
<td></td>
</tr>
<tr>
<td>2 and 3 no opt tree</td>
<td>6023 35.40% 23.81% 33.75% 78.47% 27.92% 35.40% 35.40% 35.40%</td>
<td></td>
</tr>
<tr>
<td>2 no opt tree</td>
<td>4822 28.34% 18.52% 29.35% 76.11% 22.71% 28.34% 28.34% 28.34%</td>
<td></td>
</tr>
<tr>
<td>ExtraData 2 and 3 opt tree</td>
<td>7023 41.28% 27.09% 37.01% 80.43% 31.28% 41.28% 41.28% 41.28%</td>
<td></td>
</tr>
<tr>
<td>ExtraData 2 opt tree</td>
<td>5921 34.80% 22.73% 33.37% 78.27% 27.04% 34.80% 34.80% 34.80%</td>
<td></td>
</tr>
</tbody>
</table>

To evaluate this phase we used multi-class classification metrics, and maximized $F1$-score, since we want to maximize the gesture recognition. Table 5. The best $F1$-score$_\mu$ = 48.01%, $TP = 48.01%$ and $F1$-score$M = 33.59%$ results are using the optimized (pruned) tree with the extra training data added and with the 2nd and 3rd order moments as features.

In the classifications results of the whole process starting from skin detection until the hand segmentation, we also get the best $F1$-score$_\mu$ = 54.61%, $TP = 54.61%$ and $F1$-score$M = 35.09%$ results using the optimized (pruned) tree with the extra training data added with the features being 2nd and 3rd order moments, Table 6.

Table 6: Assessment of Gesture Classification Total Testing Data Of All Bags, for each classification tree variant for whole project

<table>
<thead>
<tr>
<th>Classification variant</th>
<th>Originals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segmented Data without Flaws</td>
<td>6554 38.53% 25.77% 28.08% 79.51% 27.28% 38.53% 38.53% 38.53%</td>
</tr>
<tr>
<td>2 and 3 opt tree</td>
<td>6433 37.81% 25.39% 36.46% 79.27% 29.94% 35.81% 37.81% 37.81%</td>
</tr>
<tr>
<td>2 opt tree</td>
<td>4881 28.69% 18.69% 28.54% 76.23% 22.59% 28.69% 28.69% 28.69%</td>
</tr>
<tr>
<td>2 and 3 no opt tree</td>
<td>6023 35.40% 23.81% 33.75% 78.47% 27.92% 35.40% 35.40% 35.40%</td>
</tr>
<tr>
<td>2 no opt tree</td>
<td>4822 28.34% 18.52% 29.35% 76.11% 22.71% 28.34% 28.34% 28.34%</td>
</tr>
<tr>
<td>ExtraData 2 and 3 opt tree</td>
<td>7023 41.28% 27.09% 37.01% 80.43% 31.28% 41.28% 41.28% 41.28%</td>
</tr>
<tr>
<td>ExtraData 2 opt tree</td>
<td>5921 34.80% 22.73% 33.37% 78.27% 27.04% 34.80% 34.80% 34.80%</td>
</tr>
</tbody>
</table>
5.4 Finger pointing vector

We tried different \( w_r \) (weight to establish the radius, to surround the points to be counted on the finger extremities); bins in the histogram. To evaluate these methods, we extracted a sequence of 51 frames from a bag, used the manual classification data of the pointing gesture for the tests, and classified the results into true detections and false detections, Table 7 and 8. Having the best result for the histogram method with 4 bins.

(a) True positive fingertip  (b) False positive fingertip

Fig. 5: Fingertip of the pointing finger marked in blue

<table>
<thead>
<tr>
<th>( w_r )</th>
<th>True positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>(~60.78%)</td>
</tr>
<tr>
<td>0.50</td>
<td>(~64.71%)</td>
</tr>
<tr>
<td>0.25</td>
<td>(\sim56.86%) (29/51) to (\sim60.78%) (31/51)</td>
</tr>
<tr>
<td>0.125</td>
<td>(\sim33.33%) (17/51) to (\sim37.25%) (19/51)</td>
</tr>
</tbody>
</table>

Table 8: Results by the histogram detection

<table>
<thead>
<tr>
<th>bins</th>
<th>True positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>(~74.51%) (38/51)</td>
</tr>
<tr>
<td>8</td>
<td>(~23.52%) (12/51) to (~25.49%) (13/51)</td>
</tr>
</tbody>
</table>

6 Conclusions

In this thesis, our aim was to teach an iCub to identify the pointing directions and orientation of the fingertip. At first it looks like an easy task, but it is a complex problem that we divided into 5 parts to solve independently. This is a chain process with the input on one being the output of another, and the efficiency and accuracy of the final result depends on the efficiency and accuracy of all the previous stages.

The automatic hand location detection is achieved through skin detection, and through many experiments we reached the conclusion that the best method include the modified implementation of the article [14] and a new color space we constructed (new hue). We used the non-dynamic option from the article (the 3rd best), but the results can be improved by using the new color space. We achieved two very good options with the new hue: one dynamic, the other non-dynamic. Both of them look very promising, but the dynamic has the advantage of having no need for manual experimentation to fine-tune the thresholds.
The second step is to segment the hand, and through many experiments we reached the conclusion that the best method is to insert a bounding sphere around the hand, and use the point with the nearest euclidean distance to the camera as a starting point.

The third step is classifying the hand gesture. Through experiments using different tree variations and features, we get the best results with the optimized (pruned) tree with the extra training data added and using the 2nd and 3rd order moments as features.

Finally, the fourth and last step is to identify the fingertip of in the pointing gesture. The best results were with the 4-bin histogram of the projected pixels on the pointing axis, where we calculated the greatest difference between bins to find out where the finger was pointing.

6.1 Limitations
Despite we had very good results in the skin detection, the best option for skin detection does not necessary become good for hand segmentation. The skin detection can either improve or deteriorate the hand segmentation results, and it was very difficult to predict what options can improve the hand segmentation.

The hand segmentation is capable to segment the hand independently from the skin, using only the depth information, but not very efficiently. To improve this, we used the skin detection, but the results depend more on not ignoring the hand pixels than on the accuracy of detecting the whole skin. We also came across the problem of the Kinect sensor not registering the hand well, creating flaws that make the correct gesture classification difficult or impossible.

The fingertip detection is totally dependent in identifying the pointing frames. It was considerably harder then we expected to increase the true positives. Nevertheless, we managed to complete our tasks and achieve our main goal, which was finding a way of detecting the user pointing direction.

6.2 Future work
There are many aspects that can be improved to increase the accuracy or achieve more speed. Research alternatives to reduce the flaws in the depth image, by predicting absent information. Research new features, or better algorithms to improve the gesture classification. Research better methods to detect fingertips, increasing the algorithms efficiency. Instead of only identifying the pointing gesture, it would be interesting to identify what object it is pointing to, and also identify more gestures. Remove some restrictions, like improving the speed for real-time application, tolerate partial occlusion, allow greater differences in illumination, allow greater hand size differences, allow multiple users and multiple hands. Also explore non-classical techniques to achieve faster, more robust and more accurate gesture recognition systems and fingertip detection.

References
2. Beleboni, M.G.S.: A brief overview of microsoft kinect and its applications