3D Visual Sensing of the Human Hand for the Remote Operation of a Robotic Hand

Regular Paper

Pablo Gil1,*, Carlos Mateo1 and Fernando Torres1
1 Physics, Systems Engineering and Signal Theory Department, University of Alicante, Spain
* Corresponding author E-mail: Pablo.gil@ua.es

Received 02 May 2013; Accepted 11 Dec 2013
DOI: 10.5772/57525

© 2014 The Author(s). Licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract New low cost sensors and open free libraries for 3D image processing are making important advances in robot vision applications possible, such as three-dimensional object recognition, semantic mapping, navigation and localization of robots, human detection and/or gesture recognition for human-machine interaction. In this paper, a novel method for recognizing and tracking the fingers of a human hand is presented. This method is based on point clouds from range images captured by a RGBD sensor. It works in real time and it does not require visual marks, camera calibration or previous knowledge of the environment. Moreover, it works successfully even when multiple objects appear in the scene or when the ambient light is changed. Furthermore, this method was designed to develop a human interface to control domestic or industrial devices, remotely. In this paper, the method was tested by operating a robotic hand. Firstly, the human hand was recognized and the fingers were detected. Secondly, the movement of the fingers was analysed and mapped to be imitated by a robotic hand.

Keywords RGB-D, Kinect, Hand Recognition, Interaction Human-Robot

1. Introduction

The recognition and tracking processes of non-rigid 3D objects have been widely studied. They have been used in different fields: human body recognition [1][2], human tracking [3], activity identification [4], imitation of human movements and gesture control [5], etc. In recent years, the emergence of low cost 3D sensors such as Microsoft’s Kinect has introduced a new mechanism of interaction between users and digital characters in the video game industry. Furthermore, the development of open-source and free libraries [6] have provided new developments and advances in the recognition of articulated structures such as human limbs [7], facial gestures [8] or hands [9] as well as human-robot interactions [10][11]. Low cost 3D sensors allow for the development of a strong emotional connection between the player and the character inside the video game but also they will help in the control of industrial and domestic devices, such as robots or consumer and home appliances, in the near future.

Until recently, the most common methods [4] of hand detection and gesture recognition involved skin segmentation techniques, edges, silhouettes, moments, etc., from 2d colour or greyscale images. In these cases,
the main problem was the lighting conditions. They limited the extent to which features could be extracted from images. In this method, the detection of the intensity value of each colour can be affected by race, skin tone and how the light rays reflect on the surface of each human body part. Moreover, another problem of using colour images acquired from RGB cameras is the presence of occlusions. These can be caused by shadows or the presence of multiple objects at different planes of depth in the scene. The depth information can be useful for dividing the planes of the scene and for removing unwanted objects even though they have similar features such as colour, silhouette or shape.

In the last few years, some approaches have been developed which use RGBD or ToF cameras supplying real time depth information per pixel. For instance, in [12] the authors reviewed computer vision techniques using Kinect. Han et al. summarized some interesting works on hand detection. Rayi Yamu Tara et al. [13] considered hand segmentation as a depth clustering problem. The critical aspect of this was to determine a threshold, indicating the depth of the level at which the hand was located. Liang et al. [14] detected hands in depth images using a clustering algorithm followed by morphological constraints. Afterwards, a distance transform to the segmented hand contour was performed to estimate the palm and its centre. Caputo et al. [15] used the human-body skeleton generated by Kinect. On this skeleton map, the positions of both hands can be easily extracted using the given hypothesis. Given the 3D position of a hand, the algorithm found the corresponding hand size in a look-up table, which stored the sizes of a standard human hand at different depth levels. In other works, as in [16][17], the authors approached the hand detection process as a pixels labelling problem. The skin colour detector on the RGB image and the clustering on the depth image determined two conditions with which to qualify a hand pixel. The hand region was the intersection of a skin region and a uniform region that was closer to the camera.

The method presented in this paper uses the depth information obtained from Kinect to identify the gestural movement of human fingers. Afterwards, the recognition method is tested by a robotic hand which mimics the movements of the detected human hand. Along similar lines, Zhang et al. [18] combined 3d vision techniques to extract hand movements and these techniques were applied to teleoperate a robotic arm and its effector. That method had several undesirable assumptions, as the user should be in a specific pose and also should be in the foreground of the image range. Also, there could be no objects between the sensor and the human operator. Those assumptions are very restrictive to controlling robotic devices or agents (i.e., the hand should be joined to a skeleton body or the hand should be the first object in the image or it should to be in a pose in a location within the range image). The presented approach overcomes these problems and disadvantages. As such, the proposed method provides several advantages in relation to other similar methods, like [11]. First, the method does not need to search the human skeleton to find the human hand, nor does it require the human operator to adopt a specific position. Second, it works in unknown environments without any prior knowledge of the objects being operated. In addition, unlike the method presented in [16][17], the proposed approach does not require a 3D hand model to be built in order to search for discrepancies between visual observations of a human hand and model. Moreover, it does not take a set initial reference position to find the human hand in the range image. Our approach only assumes the initial pose of the human hand, the neutral pose, but not a specific location in the scene. Its weakness is that only a Caucasian person’s hand can be detected correctly. Another advantage is that the proposed method uses a very simple model of the hand whereby the hand is considered a single object as opposed to in the method presented in [16] which used a complex hand model. It considered the hand as having different volumes and they were based on a set of 27 assembled geometric primitives.

The visual sensing of human hands to control robotic agents has limitations, such as delays between detection and real time teleoperation and feedback information loss during the recognition process, but there also considerable benefits, such as cost reduction if compared with a teleoperation bilateral system where a joystick gives master commands to a slave which performs tasks. Moreover, the purpose of using human gestures to control robotic agents or consumer appliances is that it is more intuitive and accessible for the human operator. The human operator does not require specific training to control the devices. However, force feedback from the slave (i.e., the robotic agent) to the master (i.e., the human hand) representing tactile information cannot be managed from a vision system. The advances in human gesture recognition for controlling robotic devices may provide simple ways to control domestic devices and home appliances such as robotic vacuum cleaners, toy robots, domotic systems, smart TVs, etc.

The paper is organized as follows: Section 2 explains the sensors and the software architecture of the proposed human-robot interaction method. Section 3 describes the steps implemented to detect the hand using the RGBD data. Section 4 explains the human and robot models and the implementation in ROS of the communication between the robotic hand and the gesture recognition method. Some experimental results are also discussed and shown in section 5. These experimental results
shown in a real-world scenario demonstrate the robustness of the approach. Section 6 presents the conclusions of the work described in this paper.

2. Implemented Architecture

The proposed method uses Kinect to acquire range images like point clouds. Kinect consists of two sensors: an IR CMOS and a RGB to acquire colour images at 640x480 pixels. Its frame rate is 30 fps which is enough to sense motion. The IR sensor provides a depth map of up to 2048 levels of sensitivity. The commercial software of Kinect limits the workspace to between 1.2 to 3.5 meters, approximately. However, the sensor can work between 0.7-6 meters.

In the last three years, some drivers and libraries have been developed for image acquisition with Kinect using a PC, such as KinectSDK, OpenKinect and OpenNI. In this work, the proposed method has been implemented in C/C++ using open-free libraries: PCL (Point Cloud Library) [6], ROS (Robot Operating System) [20], both running on a Linux platform installed on a standard PC. For this work, we have used neither commercial software nor other free-open software which provides full-body 3D motion capture or gesture recognition.

4. A segmentation process is applied. In this step, the point cloud is segmented to extract interest points for each frame. The interest points define pixels the colour of which corresponds with the Caucasian skin tone. This module is composed of two blocks: The first block, (see Figure 1, block 4.a), uses global thresholds determined from prior knowledge, to label candidate colour regions. The second block, (see Figure 1, block 4.b), estimates the adaptive thresholds in HSV colour space based on the entropy and statistics average of the hue and saturation from a frame sequence, dynamically.

5. A clustering process is used to divide an unorganized point cloud into smaller parts. The clustering process is carried out using concurrent programming techniques. In this way, the points which relate the 3D-candidate region to human body parts are grouped into subsets. Each subset is a cluster of points which defines a 3D-region with human skin colour. This is done for each frame of the image sequence.

6. The combination of steps 4 and 5 defines the set of 3D-regions (subsets of points) that represents the human body parts, $S$ (see Table 1.) Consequently, in this step, a descriptor is computed for each subset, $P_i$. The descriptor classifies the kind of object according to its shape. Thus, it is possible to recognize and locate the human hand in each frame, as a point cloud. The hand is modelled like a voxel. A voxel is a 3D mesh which contains all the points inside its coordinates. This voxel is used to define a representation of the pseudo-skeleton of the hand. Thus, instead of managing thousands of points per frame, the algorithm manages a few voxels.

7. Finally, a tracking process is carried out during the estimation of the finger positions and it provides the search and detection process of the fingers in each frame of the video sequence when the movements occur. Thus, the computation time is reduced for the initial detection process of the human hand.
2.2 The interaction process

The procedure for the remote operation of the robotic hand is carried out by calculating the pose of the human hand for each frame of the video sequence. The ROS platform [20] is used as the communication system to send to the robotic hand the desired position for each finger. Thus, the distributed architecture of the ROS system is used to combine the kinematic control of the robotic hand with the human hand recognition. ROS provides similar services to operating systems such as hardware abstraction, inter-process communication and library integration, similar to the code for the recognition process. In our case, two ROS nodes were implemented: one for the robotic hand control and another for the visual acquisition and processing of the hand recognition as shown in Figure 2.

![Figure 2. Architecture of the system and communication between devices in the ROS platform](Image)

1. The human hand, palm and fingertips are recognized and tracked in a video sequence, in advance. Then, in this step, the pose of the hand and the positions of the fingers are computed using a simple model based on Euclidean distances and angles between the palm and fingertips to determine the movement between consecutive frames of the sequence.

2. Later, these movements of the fingertips are mapped to the fingers of the robotic hand. To do this, a human-to-robot grasp mapping scheme is defined depending on the type of robotic hand used. In this work, we have used a Barrett [21] hand with three fingers as shown in Figure 2. The type of robotic hand determines the kind of movements that can be replicated, the configuration of the fingers when they set a pose to imitate the movements of the human hand correctly, and the position and orientation of the fingertips. Unlike in the work studied in [22], the movements are not evaluated based on the simulation of movements in a virtual environment for the fingers and robotic hand. In this paper, the robot directly imitates human behaviour in real time. These functionalities have been implemented using ROS. Moreover, the kinematic structure of the human hand is computed from the image and we have not used position mappings from the exoskeleton as in [23].

3. Finally, the method establishes communication between the robotic hand and the recognition process based on the standard message structure of ROS which sends information to ROS processes implemented as nodes. The first ROS node acts as the server of the kinematic control of the robotic hand and it is executed in a computer which is directly connected to the robotic hand through the serial port. This node performs a service which receives the joint angle of the robotic hand and it sends the corresponding command through the serial port in order to execute it. This node also publishes the joint angles of the hand into a topic so that the current joint state of the hand can be monitored in real-time. The second ROS node is a client of this service which implements the human hand recognition algorithm and it can be executed on a different computer. Specifically, this second ROS node is connected to the Kinect camera by a USB port. It communicates with the camera through the OpenNI drivers and it uses the PCL library [6] to implement the recognition process previously mentioned. It is described in the next section.

<table>
<thead>
<tr>
<th>$P = {p_1, p_2, ..., p_n}$</th>
<th>Point cloud: 3d points set</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p=(x,y,z)$</td>
<td>The position of a 3d point</td>
</tr>
<tr>
<td>$(r,g,b)$</td>
<td>The colour of a 3d point</td>
</tr>
<tr>
<td>$P_f$</td>
<td>Point cloud filtered by depth to eliminate the background</td>
</tr>
<tr>
<td>$P'_f$</td>
<td>Point cloud sampled in order to reduce the density of the points.</td>
</tr>
<tr>
<td>$S = {P_1, P_2, ..., P_n}$</td>
<td>A set of n point clouds</td>
</tr>
<tr>
<td>$P_{fs}$</td>
<td>Point cloud of a skin region</td>
</tr>
<tr>
<td>$P_{fns}$</td>
<td>Point cloud of a non-skin region</td>
</tr>
<tr>
<td>$V_i = {v_1, ..., v_k}$</td>
<td>Vertices of a 2d convex-hull</td>
</tr>
<tr>
<td>$\bar{P}_i$</td>
<td>Centroid of a 3d convex-hull</td>
</tr>
<tr>
<td>$H_k(p_i) = {V_i, \bar{P}_i}$</td>
<td>Human hand descriptor</td>
</tr>
</tbody>
</table>

Table 1. Symbols list and notations

3. Human hand Recognition Process

In the literature [24] appear some works that combine the use of a RGB camera and depth information from a ToF camera to obtain range images, but they require two off-line calibration processes to obtain the internal camera parameters and external camera parameters: rotation and translation between the coordinate systems of each sensor. In our case, the acquisition of range images, using Kinect, provides uncalibrated data (colour and depth) from two different sensors RGB and IR, too. However, these sensors are not aligned and for further processing steps the synchronization of signals and a rectified process are required; using OpenNI together with PCL, this problem is solved and a point cloud is created. The point cloud defines a RGB-D image. Sometimes, the
processing of point clouds results in both a high computational cost (processing time) and a high space cost (storage). Therefore, the implemented algorithms and methods must be as fast as possible.

3.1 Filtered and sampled

At this stage, we filter the input data using the depth information, \( z \), of each point, \( p \), of the point cloud, \( P \). Thus, let \( P \) be the scan data set for the image. It consists of 3D points represented as follows:

\[
P = (p_1, p_2, ..., p_n)
\]

(1)

Thus, a distance band pass filter is applied to an initial point cloud as shown in (1) and (2).

\[
\forall p_1 \in P \rightarrow H(P, t) = P_f
\]

(2)

where \( P_f \) is the filtered data set. It consists of 3D points where \( z \) is the depth between 1.2 and 3.5 meters, this parameter is imposed by the working distance between the human operator and the sensor. \( H \) is the function of filtering \( P \) and obtaining the output \( P_f = (p_1, p_2, ..., p_m) \) where \( m \ll n \). Thus, this process removes information from the background that could be considered noise. To improve the accuracy and efficiency of the method, one step more has been considered. \( P_f \) is filtered again, as follows

\[
H_s(P_f, t) = P_f
\]

where \( H_s \) is a function by which to sample the point cloud. To do this, for any point, \( p_i \), assuming they are the surviving points after applying the depth filter from (2), we remove all the neighbouring points, \( p_f \), into the open ball of the radius, \( d_s > 0 \) around \( p_i \) as follows:

\[
B(p_i, d_s) = \{ p_j \in P_f : \sqrt{\sum_{j=1}^{n}(p_i, p_j)^2} < d_s \}
\]

(3)

where the sampling ratio, \( d_s \), is the radius of the ball and it is computed as the minimum allowable Euclidean distance between any point that belongs to \( P_f \) and its nearest neighbours. In our experiments, a value of 3mm provided good results in the detection process and improved the detection speed in indoor environments.

The purpose of these two processes is to reduce the density of points to be processed in the next stages while maintaining an acceptable resolution to distinguish the hand palm and the fingers (Figure 3a-b) without causing significant discontinuity in the surface of these human body parts.

3.2 Skin detection

The workspace can contain surrounding objects (i.e., other body parts, other persons, walls, doors, furniture, etc.). The objective of this step is to implement a colour segmentation to identify the set of points in \( P_f \) which has a texture similar to Caucasian human skin. However, this process means that other human body parts (such as hands, arms, legs, etc.) are unwittingly detected. In this method, a combination of two strategies has been implemented to resolve the problem of colour skin detection in a video sequence. First, a pre-segmentation process is carried out (Figure 4a). This is based on a rule set in which fixed global thresholds of two space colour, RGB and HSV are used. Second, an adaptive segmentation process based on HSV is implemented to find the values of the optimal threshold for hue, saturation and value (Figure 4b). Then, the segmentation is dynamically adapted to light conditions while new images are acquired.

In the studies [19][25], two rules based on thresholds are used to detect human skin in the pre-segmentation process. The first rule is used when the indoor environment is illuminated with natural light (sunlight). The second rule is used when the indoor environment is illuminated with artificial light. These rules are defined below in equations (5) and (6). These values are well suited to discriminating skin pixels from non-skin pixels in 3D images captured of white-skinned people. Sometimes the environment is illuminated with both kinds of light, sunlight and artificial, then the more restrictive rule is applied in those cases, (5) or (6). The colour temperature, hot or cold, and the lumens determine which equation is suitable.
This process is completely automatic and it does not require any user interaction but these parameters are image dependent and they have to be selected based on the properties of the class of images used as the input to the sensor. For offline processes, we have adjusted them according to the Gaussian distribution in the colour space of 3D images. A colour image with skin-colour regions is transformed into a greyscale image so the value at each point shows the likelihood of the point belonging to the skin tone. (i.e., some different sequences have been used in some environments to fix these by studying these Gaussian distribution). In order to handle uncontrolled imaging conditions, a dynamic or adaptive threshold classification technique is used to determine some initial thresholds from the average intensity of the high gradient. Different ways to compute these thresholds exist in the literature. However, the thresholds used in this work are used to extract skin information in the point cloud without losing points of skin. This does not hinder the hand shape recognition. Then, the thresholds \( T = (t_f,t_g,t_h,t_m,t_d) \) are used to split the 3D image into two groups, \( P_f = \{ P_{f1}, P_{f2}, ..., P_{f_k} \} \) where \( P_f = (p_1, p_2, ..., p_k) \) with \( k < n \). This situation can be modelled using the following function:

\[
\frac{P_{i/skin}}{P_{i/non-skin}} \geq T \rightarrow P_f = \{ P_{f1}, P_{f2}, ..., P_{f_k} \} = \{ 1 - T, T + 1 - k \} \quad (4)
\]

where \( P_{i/skin} \) and \( P_{i/non-skin} \) denote the points of skin and non-skin, respectively.

Note that the values of \( T \) that classify the points for each candidate group are given by the following two expressions, (3) and (6). Thus, the rule for pre-segmenting human skin when the scene is lit by sunlight can be written as:

\[
(r_f > f) \land (g_f > g) \land (h_f > h) \land (r_f - \min(r, g, h)) < t_m \land (g_f - g) < t_d \land (r_f > h) \land (g_f > g) \land (r_f > h) (5)
\]

where \((r_f, g_f, h_f)\) are the intensity values of each point of the cloud \( P_f \) and the threshold can be modelled by \( T = (t_f, t_g, t_h, t_m, t_d) = (95,40,20,15,15) \).

In addition, the pre-segmentation process is redefined when the scene is lit by artificial light.

\[
(r_f > f) \land (g_f > g) \land (h_f > h) \land (r_f - g_f) \leq t_d \land (r_f > h) \land (g_f > h)\quad (6)
\]

where \((r_f, g_f, h_f)\) are the intensity values of each point of the cloud \( P_f \) and the threshold can be modelled by \( T = (t_f, t_g, t_h, t_m, t_d) = (220,210,170,15,15) \). Hence, a point is considered skin when it satisfies either (5) or (6).

In this work, the segmentation process is based on the assumption that skin colour is homogeneous. In other words, the human hand does not contain other colours. Obviously, objects with similar colour could be detected (other human body parts, wood furniture, etc.). Then, if the hand is of uniform colour, we improve on the previous noise reduction by adding a second step in which we keep only the points of \( P_f \) which are of a similar hue. Then, a point is considered skin if and only if its hue satisfies the expression:

\[
(h_i < h_m) \lor (h_i > h_m) \lor (s_i < s_m) \lor (s_i > s_m)
\]

where \( h = (h_m, h_s) = (33,310) \) is the hue thresholds, they must belong to \([0, 360] \). Also, \( s = (s_m, s_s) = (0.19,0.81) \) is the saturation thresholds, they must be between \([0, 1] \). The hue is computed as (8).

Hence, a pixel is considered skin when it satisfies (5) or (6) depending on whether the environment is illuminated by sunlight or artificial light, respectively. Moreover, only one rule is applied when both lighting systems are used, the rule which is satisfied more accurately. Furthermore, at the same time, the pixel must complete equation (7).


g - h_{max} < g - h_{min} = \left\{ \begin{array}{ll}
60 & \text{if}\left(\max(f, g, b) = r\right) \land (g \geq b) \\
60 & \text{if}\left(\max(f, g, b) = r\right) \land (g < b) \\
60 & \text{if}(\max(f, g, b) = g) \\
60 & \text{if}(\max(f, g, b) = b)
\end{array} \right. \quad (8)
\]

**Figure 4.** (a) Pre-segmentation. (b) Adaptive segmentation.

After the pre-segmentation process, an adaptive segmentation is implemented. The adaptive thresholds are computed from the results of the search of the average of the minimum and maximum values in the hue and saturation channels of the HSV space for every three frames. The distribution of the hue values for the skin tone is usually close to 0 or 360 degrees. Those values represent yellows and purple tones, respectively. Consequently, the minimum, \( h_{min} \), and the maximum, \( h_{max} \), for the values nearest to 0, and another minimum, \( h_{max} \), and maximum, \( h_{min} \), for the values nearest to 360 are
estimated for every three frames in the video sequence. The rules for this step can be written as (9) and (10):

\[
(b_i > h_{\text{lin}}) \land (b_i < h_{\text{us}}) \lor (b_i > h_{\text{us}}) \land (b_i < h_{\text{lin}}) \quad (9)
\]

\[
(s_j > s_{\text{m}}) \land (s_j < s_{\text{m}}) \quad (10)
\]

where \(S_{\text{e}}\) is the minimum value of saturation and \(S_{\text{m}}\) is the maximum value of saturation.

### 3.3 Hand descriptor and tracker

Once the regions with skin colour are obtained, the goal is to estimate the regions which represent the human hands. However, sometimes, the workspace can contain surrounding objects that are not skin (i.e., doors, furniture, etc.). In order to accurately detect and track the human hand, this information must be filtered and eliminated.

The main idea of our approach is to search the nearest neighbour to cluster the point cloud \(P_{\text{fs}}\). Then, given a set of points, \((p_1, p_2, \ldots, p_m)\) in the metric space of \(P_{\text{fs}}\) and belonging to \(P_{\text{fs}}\), new points \(p_i\) in the same metric space are sought and added to the set if they are the closest to \(p_i\). Thus, in this step, we have used Kd-tree structures to search for the k-nearest neighbour and to find the closest \(K\) neighbours within a certain radius, the used search distance is 1mm. The Kd-tree structures recursively divide the data points \(P_{\text{fs}}\) using binary trees. The space partitioning data are organized in k-dimensional space, which is very efficient for low dimensionality data. This algorithm allows us to organize \(P_{\text{fs}}\) (skin candidate points) as point subclouds:

\[
P_{\text{fs}} = \{p_1, p_2, \ldots, p_k\} \quad (11)
\]

where each point subcloud, \(p_j\), represents a candidate object. Every point subcloud has to fulfill a quality factor in order to be considered a candidate for a body human part.

Afterwards, a classifier to discriminate between the different regions using standard descriptors is added, to detect the human hand in the set \(P_{\text{fs}}\). This classifier consists of a global descriptor of the human hand and a multi-part description based on spatial relations between parts of the human hand, such as the fingers and the palm. On the one hand, the shape is used as a global descriptor to identify the type of object in the scene. Therefore, for each \(p_i\) of \(P_{\text{fs}}\), the shape is calculated and it is compared with a human hand model. The number of samples for the human hand model limits the poses that can be used to detect the hand in the initial position (in our case, neutral pose). Then, from the candidate objects, according to (11), the algorithm determines the number of hands in the scene and their locations. Afterwards, the convex-hull of \(P_{ij}\), \(cH_{3d}(P_{ij})\) is computed. The convex-hull is defined as the smallest convex set containing \(P_{ij}\). In addition, a point in \(P_{ij}\) is a vertex, \(v_{ij}\), of the convex hull of \(P_{ij}\), if it is an extreme point in relation to \(P\) (i.e., it is part of the boundary). In the presented method, the depth has been discarded, then a 3D convex-hull is calculated as a 2D convex-hull, and it represents the minimal polygonal containing the set of points of the \(P_{ij}\). Geometrically, \(V_{ij} = (v_1, v_2, \ldots, v_k)\) are the vertices which define the polygonal contour in the 2D-convex-hull, \(cH_{2d}(P_{ij})\) and they are used as a descriptor in order to identify the shape of \(P_{ij}\) together with the 3D-centroid as follows (see Figure 3d):

\[
H_k(P_{ij}) = \{V_{ij}, \tilde{P}_j\} : V_{ij} \in cH_{2d}(P_{ij}), \tilde{P}_j = \sum_{P_{ij}} p_{i:j} \quad (12)
\]

where \(\tilde{P}_j\) is the 3D-centroid and \(V_{ij}\) are the vertices of 2D-convex-hull, \(cH_{2d}(P_{ij})\).

On the other hand, the structure of the human hand is modelled using the relations between the parts of the human hand, in this case between the vertices, \(V_i\) and \(\tilde{P}_j\) between each other, \(V_{ij}\), according to the error function shown in (13). The first part accumulates the distance between the vertices and the second part accumulates the distances of the centroid for the vertices of the convex-hull.

\[
E(H_k(P_{ij})) = \sum_{V_{ij} \in H_k(P_{ij})} \left| V_{ij} - \tilde{P}_j \right| + \sum_{V_{ij} \in H_k(P_{ij})} \left| V_{ij} - \tilde{P}_j \right| \quad (13)
\]

Moreover, other compositional rules are added to represent the human hand as a combination of these descriptors, \(H_k(P_{ij})\). In other words, a candidate subcloud is considered as a human hand, if the set of points, \(P_{ij}\), satisfies the next constraints:

The number of vertices \(V_{ij}\) must be greater than seven and smaller than 15. At best, the hand shape model is represented by \(k=7\) vertices: Five are at the end of the fingers (fingertips) and two are in the wrist (joining the hand and the forearm).

The maximum limit of the vertices is \(k=15\) because the joint position of the hand and the background in the range image with a similar texture can generate a polygonal shape for the convex-hull with a large number of vertices. These false vertices can cause confusion in the shape descriptor. For this reason, the proposed method tests the structure of \(V_{ij}\) and the vertices which do not keep a Euclidean distance in the 3D space of \(P_{ij}\) in
accordance with (13), are removed. Consequently, the vertices near the neighbourhood are considered as false vertices. In addition, a semantic interpretation of the human hand is used to refine the recognition process. This method uses the specific knowledge of how, in the geometrical model of the human hand, the relation between the fingers and palm will be (Figure 5a). This way, the recognition is adjusted such that \( E(H_b(P_f)) \), and the following semantic constraints are enforced. Therefore, it is necessary to consider that:

The farthest vertex to the centroid represents the middle finger, if it has two neighbouring vertices (fingertip of first and ring fingers).

In addition, the vertices which represent the first and ring fingers must have two neighbouring vertices, too. The vertices which represent the thumb and small fingers, respectively, must only have one neighbouring vertex.

This semantic description is used as a similarity measure to compare the template of the human hand (model), \( HM \), with each descriptor computed for each point subcloud, \( E(H_b(P_f)) \), as follows:

\[
E(H_b(P_f),HM) = \arg\min \sum_{P_f \in \Omega(H_b)} |P_f - HM|
\] (14)

The centroid determines the position of the palm (bone structure which consists of metacarpals and carpals) and the vertices determine the position of the fingers (distal phalanges).

The last step of the recognition process is to present a framework for the estimation of the position of the hand (palm and fingertips) from multi-modal (colour and depth) video sequences. The computation of the 3D optical flow or range flow from image sequences is used for hand tracking when the pose of the hand is moved and the viewpoint between the camera and the hand is changed. The algorithm used is based on previous work on 2D tracking, such as [26]. However, in this case, we assume a 3D voxel to track. This is a small 3D neighbourly mask (in our case 0.3 mm), for each finger \( f_i \) and the palm. Then, if \( f_i = (x, y, z) \) is this voxel of a finger \( f_i \) at time \( t \) and it moves with a displacement \( \Delta(x, y, z) \) from \( f_i = (x, y, z, t) \) to \( f_i = (x + \Delta x, y + \Delta y, z + \Delta z, t + \Delta t) \) over time \( \Delta t \). Then one can obtain:

\[
\begin{align*}
\dot{f}_i(x, y, z) + \ddot{f}_i(x, y, z) + \dot{f}_i(t) + \ddot{f}_i(t) = 0
\end{align*}
\] (15)

where \( \dot{v}_x, \dot{v}_y, \dot{v}_z = (\Delta x/\Delta t, \Delta y/\Delta t, \Delta z/\Delta t) \) is the 3D optical flow and \( \dot{f}_i(x, y, z) \) are the 3D spatio-temporal derivatives of the 3D spatial intensity gradient.

4. Human-Robot Interaction

4.1 Robotic hand vs human hand model

In order to operate the robotic hand using the human hand, an implementation of appropriate position mappings is required due to the different kinematic structures of the human hand and robotic hand. In this section, the approach for the position mapping of a human to robotic hand algorithm is presented.

The Barrett Hand is an under-actuated robotic hand which is widely used as an end-effector in current robotic manipulators. This hand is composed of three articulated fingers, \( f_1 \), which have an identical mechanical structure, and a palm \( C_p \) which integrates four servomotors and the control electronics. Thus the robotic hand can be modelled as \( H_r = \{f_1, f_2, f_3, n_{-2}\} \) where \( f_i \) denotes the position of each finger in relation to the palm (opening and closing fingers) and where \( n_{-2} \) represents the position of the fingers \( f_1 \) and \( f_2 \) when a spreading movement is generated (repositioning in the fingers configuration). The spreading movement of fingers \( f_1 \), and \( f_2 \) is driven by the fourth and last servomotor. In other words, the bases of \( f_1 \) and \( f_2 \) can rotate around the palm, driven by the fourth servomotor. In addition, each finger, \( f_i \), is composed of two phalanges and they are driven by one servomotor. Then, each of the three finger motors must drive two joint axes. Additionally, this robotic hand contains a breakaway system which decouples the inner and the outer phalanges when the inner phalanx makes contact with an object with a specific force, so that an enveloping grasp of the object is possible.

The set of movements of the robotic hand considered in this work are shown in the Figure 5b. The equivalence in the human hand is shown in Figure 5a.
The human hand model $HM = \{f_1, f_2, f_3, f_4, f_5, c_p\}$ has been limited to achieve a proper imitation of hand movements. In particular, only three fingers of the human hand are considered although the five fingers were detected in the recognition process, discussed in the previous section. Then, the model is defined as $HM = \{v_1, v_2, v_3, \tilde{P}_1\}$, where $v_1, v_2$ and $v_3$ are the $xyz$-coordinates of the fingertips in relation to the palm, $\tilde{P}_1$, for the fingers: thumb, middle and first, respectively. The index, middle and thumb human fingers are chosen because they can be easily moved without moving other fingers involuntarily. In addition, they allow the user to simulate the grasp pose, open pose, close pose, etc. In our method the full kinematic configuration of the human hand with 19 links and 24 DoFs is not considered [27], only a simplified model. In addition, in order to map the human hand configuration to the robotic hand, the relative Cartesian position of $HM$ is mapped to the robotic hand $H_r$. This method uses the kinematics equations of the Barrett Hand and then, the corresponding joint angles of the three fingers are obtained and applied to the robotic hand.

On the one hand, according to the direct kinematics, the encoder counts for the servomotors of the Barrett hand are related to the joint angles as:

$$ f_i = [\theta_{i1}, \theta_{i2}, \theta_{i3}] = \left[ \frac{2q_i}{351}, \frac{q_i}{125}, \frac{q_i}{375} : q_i \in [0.17500, 0.31500] \right] $$

(16)

where $\theta_{i\ast}$ is the k-joint angle for the robot finger $f_i$ and where $q_i$ are the values that can be adopted by the servomotors. They can adopt values of between 0 and 31500 for the servomotor which controls the spreading movement of fingers $f_1$ and $f_2$. They can also adopt values of between 0 and 17500 for the other three servomotors which control the movements of opening and closing of all the fingers.

Furthermore, the maximum values for $\theta_{i\ast}$ is $140^\circ$, $\theta_{i2}$, are $47^\circ$ and $\theta_{i1}$ is $\pm180^\circ$. Since the robotic hand fingers cannot be stretched completely, the joint angles $\theta_{i2}$ and $\theta_{i3}$ are constantly $2.46^\circ$ and $50^\circ$, respectively. Then, the angle position for each finger is computed from (16) as:

$$ f_i = [\theta_{i1}, \theta_{i2}, \theta_{i3}] = [\theta_{i1}, \theta_{i2} + 2.46, \theta_{i3} + 50] $$

(17)

On the other hand, according to the fingertip Cartesian 3D-position $(x_i, y_i, z_i)$ for each $f_i$ (see Figure 6), it is computed as in equation (18).

$$ \begin{bmatrix} \theta_{i1} \\ \theta_{i2} \\ \theta_{i3} \end{bmatrix} = \tan^{-1}\left( \frac{z_i}{\sqrt{x_i^2 + y_i^2 - l_i}} \right) - \tan^{-1}\left( \frac{l_3 \sin \theta_{i3}}{l_2 + l_3 \cos \theta_{i3}} \right) $$(18)

$$ \cos\left( \frac{\sqrt{x_i^2 + y_i^2 - l_i}^2}{2l_2^2 - l_2^2 - l_3^2} \right) $$

where $l_i = [l_{1i}, l_{2i}, l_{3i}] = [50, 70, 56]$ mm are the length of the limbs for each of the robot’s fingers, $i = 1, 2, 3$.

The forward kinematics of the robotic hand describes the Cartesian 3D-position for each fingertip as:

$$ \begin{bmatrix} x_{i1} \\ y_{i2} \\ z_{i3} \end{bmatrix} = \left[ \left( l_2 \cos \theta_{i2} + l_3 \cos(\theta_{i3} + \theta_{i2}) + l_1 \right) \cos \theta_{i1} \right] $$

$$ \left( l_2 \cos \theta_{i2} + l_3 \cos(\theta_{i3} + \theta_{i2}) + l_1 \right) \sin \theta_{i1} $$

where $(l_1, l_2, l_3) = (l_{1i}, l_{2i}, l_{3i}) \geq 0$

$$ \begin{bmatrix} x_{i1} \\ y_{i2} \\ z_{i3} \end{bmatrix} = \left[ \left( l_1 - l_2 \cos \theta_{i2} - l_3 \cos(\theta_{i3} + \theta_{i2}) \right) \cos \theta_{i1} \right] $$

$$ \left( l_1 - l_2 \cos \theta_{i2} - l_3 \cos(\theta_{i3} + \theta_{i2}) \right) \sin \theta_{i1} $$

where $(l_1, l_2, l_3) = (l_{1i}, l_{2i}, l_{3i}) \leq 0$

### 4.2 Communication with the robotic hand

The communication, between the recognition process and the robotic hand via the serial link, produces latencies when short data (command frames) are sent. The Barrett hand supports 21 commands of these different types: opening fingers, moving any combination of the four axes to an array of positions, incrementally opening and closing by default or user-defined distances. The firmware of the robotic hand allows us to read the strain gauges, but not the position of the fingers to monitor if the remote operation process is correctly realized. The commands of the Barrett to achieve the robot finger poses from the servomotor steps are:

- **GHOM**: Initialization
- **SM 310b**: The robotic hand, $H_r = [f_1, f_2, f_3, \eta_{-2}]$ imitates the initial position of the human hand detected. $f_1, f_2$ are extended and positioned in line with $f_3$. Similar to the position shown in Figure 10a
- **xIO value**: An extending movement (open) with decreasing values of servomotor steps (value) for finger $x$ where $x$ can be $f_1, f_2, f_3$
- **xIC value**: A flexing movement (close) with increasing values of servomotor steps (value) for finger $x$ where $x$ can be $f_1, f_2, f_3$. Similar to the position shown in Figure 10b-c
For the finger tracking process, the robot commands are sent when a specific human finger is opening or closing.

![Figure 6](image)

**Figure 6.** Geometric relationships of the finger phalanges

5. Experimental Results

5.1 Recognition

To carry out the hand recognition, some video sequences of human hand movements were analysed. These video sequences were captured with Kinect from different viewpoints using different illuminations (Figures 7-8). Figure 7 shows how the skin adaptive segmentation applied after the pre-segmentation process provides the human hand detection with more robustness.

![Figure 7](image)

**Figure 7.** a) Static pre-segmentation (fixed thresholds). b) Dynamic segmentation (adaptive thresholds).

Figure 8 shows a sample of results obtained for three images of three different sequences. In this case each image represents three different illuminations in the same scene. From the above results, we can argue that the method can optimally recognize the hand and detect its fingertips. Furthermore, Figure 9 shows the recognition process when the hand is moved. Different human hand poses (open hand, wrist rotations in six degrees of freedom, close hand, grasp position, etc.) are successfully detected, notwithstanding the fact that for some poses false fingertips can be detected (white marks in the images) due to the complexity of the model of an articulated structure like the fingers, where there are so many small joints in a little space.

![Figure 8](image)

**Figure 8.** Recognition process in different light conditions: (a) Artificial and sunlight. (b) Only sunlight. (c) Only artificial light (by night).

![Figure 9](image)

**Figure 9.** Recognition process with different human hand poses

<table>
<thead>
<tr>
<th>Motion</th>
<th>5 fingers</th>
<th>4 fingers</th>
<th>3 fingers</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1-S5</td>
<td>93%-100%</td>
<td>95%-100%</td>
<td>95%-100%</td>
</tr>
<tr>
<td>S2-S6</td>
<td>76%-95%</td>
<td>78%-95%</td>
<td>93%-95%</td>
</tr>
<tr>
<td>S3-S7</td>
<td>91%-89%</td>
<td>94%-90%</td>
<td>99%-90%</td>
</tr>
<tr>
<td>S4-S8</td>
<td>98%-97%</td>
<td>98%-97%</td>
<td>100%-97%</td>
</tr>
<tr>
<td>Mean</td>
<td>90%-95%</td>
<td>91%-96%</td>
<td>97%-96%</td>
</tr>
</tbody>
</table>

**Table 2.** Results of the method: Correct detection rate for five fingers and when one or two fingers are missing

The experiments were tested with three people using eight video sequences/person composed of 200 frames/sequence (1600 frames/person). The sequences represent two light conditions, S1...S4 artificial light and S5...S8 both artificial and sunlight (see Figure 8). In addition, four different motion sequences with coupling between translation and rotation were considered. Thus, S1-S5, S2-S6 and S3-S7 represent hand movements in which the wrist rotation is done in yaw, pitch and roll, respectively. S4 and S8 represent finger movement (flexion/extension). The sequences that have been chosen for the fingers and palm can always be observable, then the fingers cannot be self-occluded due to their motion. The experiments have been performed using eight video sequences/person composed of 200 frames/sequence (1600 frames/person). The sequences represent two light conditions, S1...S4 artificial light and S5...S8 both artificial and sunlight (see Figure 8). In addition, four different motion sequences with coupling between translation and rotation were considered. Thus, S1-S5, S2-S6 and S3-S7 represent hand movements in which the wrist rotation is done in yaw, pitch and roll, respectively. S4 and S8 represent finger movement (flexion/extension). The sequences that have been chosen for the fingers and palm can always be observable, then the fingers cannot be self-occluded due to their motion. Table 2 shows the accuracy (detection rate) of each sequence with a person in the background. In these experiments, the proposed method has hit rates greater than 90%. In addition, the method is free from changes to the illumination and environment and it can be successfully used to interact and remotely operate using a robotic hand.
5.2 Interaction and remote operation

To evaluate the human recognition as a method of human-robot interaction, some movements were imitated (Figure 10). The fingertip positions (thumb, index, and middle) were mapped to the robotic fingers. In this work corrective movements were performed when the operation failed, the control of the robotic hand is always in an open loop.

The pose of the robot fingers was accurately imitated by the proposed method in this paper. Figure 10 shows two sequences of images extracted and processed from a video captured in real time. In these examples, the fingers of the human hand were moved then the proposed method recognized the hand and it detected the fingertips positions in relation to the palm. Afterwards, the fingers of the robotic hand were moved imitating those movements. The mapping of the robotic fingers’ position could be correlated to the human fingers’ position. The 2D and 3D distances in the image, between the fingers and palm, were computed for each finger. Later, they were used to determine the measurements for each servomotor step. The mapping algorithm adapted the measured distance of the human fingers to the workspace of the robotic hand by using a scaling factor.

Figure 10a shows an imitation of a spreading movement. In this case, the human thumb was displaced away from the index while maintaining the depth plane. Then, the fingertip $v_1$ detected from RGBD-images changed its position in relation to the palm according to a rotation, $\theta_{ij}$. Figure 10b shows an imitation of an extended hand position and later, the index and middle fingertips, $v_3$ and $v_2$, changed their positions closing the phalanges. In this case, the robot fingers were closed increasing their joint positions for the limbs. Figure 11a shows the pose computed from RGBD-images for the fingertip $v_3$ when the flex movement was carried out. In addition, Figure 11b shows how the distance between the palm and fingertip changed when an extension movement followed by another flex movement was carried out. In this case, $f_3$ was automatically operated and its pose was computed according to the robotic hand kinematics, as in (19) and (20). In order to imitate that movement, an example of the values used in the robotic hand is shown in Figure 11c. From the initial iteration to 110, the flex movement was executed in the robotic hand, and from 111 to 180, the extension movement to return it to the starting position.

![Figure 10. Video sequences, in real time, imitating the movements of human hand by Barrett. (a) $v_1$ was moved and $f_1$ rotated $\theta_{ij}$ (spreading movement). (b) $v_3$ and $v_2$ were moved then $f_3$ and $f_2$ changed.](image1)

![Figure 11. (a) Trajectory of $v_3$ for flexing movement. (b) Comparison between the distance $s_3$ computed from the flexing and extending movement of $v_3$. (c) Robot-finger joint angles for flexing and extending movement of $v_3$.](image2)
Table 3. Running time of the method

<table>
<thead>
<tr>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Kurtosis</th>
<th>Asymmetry</th>
</tr>
</thead>
<tbody>
<tr>
<td>21ms.</td>
<td>±18.9ms</td>
<td>0.46</td>
<td>1.15</td>
</tr>
</tbody>
</table>

In this case, the method took a computation time of 21ms, at each iteration, and the statistical analysis of its distribution is shown in Table 3. These computation times allow the robotic hand to operate in real time from the 3D visual sensing of a human hand.

6. Conclusions

This paper presents a method for detecting and tracking hands, by using a Kinect camera and acquiring and processing images in real time. The implemented method has been tested for different configurations of hands, with different people. Furthermore, some tests were done for several different indoor environments where the working distance was changed, the number, type and features of the objects were different and where the ambient light was also different. Moreover, this method did not require any previous calibration process. Initially, the system detects the human hand independently from the environment in spite of the fact that there are several body parts which are visible. However, to detect a human hand, the method requires that the hand initially be open. This method does not use the human skeleton recognition to limit the search space in order to detect the hands. In this work, to test the validity of our method, simple gestures of the human hand were analysed and measured to imitate human movements with a Barrett robotic hand considering its inverse kinematics. Only three fingers of the detected human hand’s anatomy were used in this task due to the restrictions of the anthropomorphic structure of the Barrett hand. The advantage of our multi-fingered manipulation system is its low cost and that it does not require a pose control using an exoskeleton.

7. Acknowledgments

The research leading to these results received funding from the Spanish Government and European FEDER funds (DPI2012-32390) and the Valencia Regional Government (GV2012/102 and PROMETEO/2013/085).

8. References


