RGBD Human-Hand recognition for the Interaction with Robot-Hand

C. M. Mateo, P. Gil, *Member, IEEE,* J. A. Corrales, S.T. Puente and F. Torres, *Member, IEEE*

***Abstract*— New low cost sensors and the new open free libraries for 3D image processing are permitting to achieve important advances for robot vision applications such as tridimensional object recognition, semantic mapping, navigation and localization of robots, human detection and/or gesture recognition for human-machine interaction.**

In this paper, a method to recognize the human hand and to track the fingers is proposed. This new method is based on point clouds from range images, RGBD. It does not require visual marks, camera calibration, environment knowledge and complex expensive acquisition systems.

Furthermore, this method has been implemented to create a human interface in order to move a robot hand. The human hand is recognized and the movement of the fingers is analyzed. Afterwards, it is imitated from a Barret hand, using communication events programmed from ROS.

I. INTRODUCTION

The recognition and tracking processes of non-rigid 3D objects have been widely studied with different and variable success rates. They have been used in different fields: human body recognition, human tracking, activity identification, robotic learning by the imitation of human movements, augmented reality and semantic mapping.

In recent years, the emergence of low cost 3D sensors [1] in the video game industry and the development of open-source and free libraries [2,3,4,5] have provided new developments and advances in the recognition of articulated structures such as human limbs [6.8], facial gestures [7] or hands [9,10,11]. Kinect is a camera system which uses a depth sensor combined with a RGB camera. Thus, the system provides a point cloud which is used to model three-dimensional data. Thus, the segmentation process of articulated objects and the human activity from unknown and complex environments is facilitated.

Until recently, the most common methods about hand detection and gesture recognition are implemented with skin segmentation techniques from colour images. In these cases, the main problem is the lighting conditions. Thereby, the detection of intensity value of each color can be affected by the human race, skin tone and how the light rays reflect on the surface of each human body part. Moreover, another problem of the colour images acquired from RGB cameras is the presence of occlusions. They can be caused by shadows, the presence of multiple objects at different planes of depth in the scene. The depth information can be useful to divide the planes of the scene and to remove unwanted objects even though they have similar features such as colour, silhouette or shape. For instance, [12] presents a method that treats 3D hand pose recovery as a minimization problem whose objective function is the discrepancy visual observations of a human hand and 3D hand model.

The method presented in this paper uses the depth information obtained from IR sensor of Kinect in order to identify the gestural movement of human fingers and the RGB sensor to recognize the skin which covers the bone structure of the hand.

The paper is organized as follows: Section II explains the Kinect technology and the software architecture of the method proposed. Section III describes the steps implemented in order to detect the hand from RGBD data. Section IV explains the implementation in ROS of the communication between the robot hand and the recognition method. Some experimental results are also discussed and shown in section III and IV. Finally, some important conclusions are reported in Section V.

II. ARCHITECTURE

The proposed method uses a Kinect camera (Figure 1) to acquire range images as point clouds. Kinect consists of two two sensors: an IR CMOS and RGB in order to acquire colour images at 640x480 (307200pixels). The acquisition speed of the device is 30fps at that resolution. Furthermore, Kinect uses an infrared projector which sends out a fixed pattern of light and dark speckles. Depth is calculated by triangulation against a known pattern from the projector. The pattern is memorized at a known depth and then for each pixel, a correlation between known pattern and current pattern is done, providing the current depth for these pixels.

In the last two years, since Kinect was marketed, some drivers and libraries have been developed for image acquisition with Kinect using PC, such as KinectSDK [2],

This work was supported in part by the Valencia Regional Government and the Research and Innovation Vice-president Office of the University of Alicante for their financial support through the projects GV2012/102 and GRE10-16, respectively.

C. M. Mateo is with the Physics, Systems, Engineering and Signal Theory Department, University of Alicante, Campus de San Vicent del Raspeig, Alicante, E 03690 Spain (phone: 34-96-5903400; fax: 34-96-5909750; e-mail: cmma@alu.ua.es).

P. Gil is with the Physics, Systems, Engineering and Signal Theory Department, University of Alicante, Campus de San Vicent del Raspeig, Alicante, E 03690 Spain (phone: 34-96-5903400; fax: 34-96-5909750; e-mail: pablo.gil@ua.es).

J.A. Corrales is with the Institute for Intelligent Systems and Robotics, Universite Pierre et Marie Curie, 4 place Jussieu, CC 173, Paris, 75252 France (phone: 33-1-44279629; e-mail: corrales_ramon@isir.upmc.fr).

S.T. Puente is with the Physics, Systems, Engineering and Signal Theory Department, University of Alicante, Campus de San Vicent del Raspeig, Alicante, E 03690 Spain (phone: 34-96-5903400; fax: 34-96-5909750; e-mail: santiago.puente@ua.es).

F. Torres is with the Physics, Systems, Engineering and Signal Theory Department, University of Alicante, Campus de San Vicent del Raspeig, Alicante, E 03690 Spain (phone: 34-96-5903400; fax: 34-96-5909750; e-mail: fernando.torres@ua.es).

OpenKinect [3] and OpenNI [4]. Moreover, the latter two can run on Windows and Linux operating systems, versus the first driver that can only run on Windows systems.

The proposed method has been implemented in C/C++ libraries using PCL (Point Cloud Library), ROS (Robot Operating System) [13] and Linux platform. The used hardware is a standard PC, 3.0Ghz Intel Core i5. In this paper, Kinect and PCL are used for the human hand recognition. Then, the fingers are detected and a simple model of them is used in order to analyze their movements. Later, these movements are replicated and/or imitated by robotic hand as Barret hand [14]. The control and the communications between Kinect and the robotic hand have been implemented on ROS as shown in Fig. 1

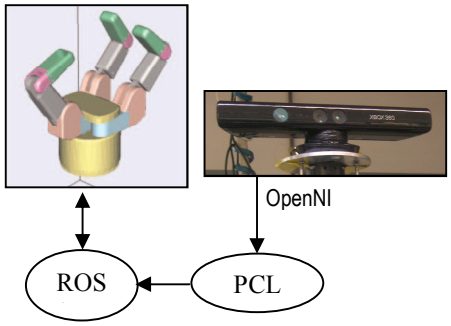

Figure 1: Hardware and software architecture.

The detection and recognition processes of the human hand using OpenNI and PCL were carried out by following the steps shown in Fig. 2:

1. Acquisition of a range image which defines a point cloud, XYZ, where each point provides RGB color information (Fig. 3).

2. Band-pass filtering of the point cloud, according to the work area. Thereby, the region of interest is restricted according to the permitted work distance and the points are removed of point cloud because they are considered noise. Kinect can work between 1 and 8 meters, approximately. This filter is more restrictive and it limits more the work distance depending on the distance between human and sensor, too close or too far.

3. Subsequently, a sampled process is done. This process provides far higher speed to the next steps such as hand recognition and visualizer. Thus, the distance sampling can be adjusted to prioritize the speed (point cloud less dense) or precision (point cloud more dense) of the algorithm.

4. In this step, the point cloud is segmented to extract interest points for each frame. The interest points define pixels whose colour corresponds with the skin tone.

5. The point cloud is divided using concurrent programming techniques. In this way, the points which define candidate objects to be human body parts are grouped in subsets. Each subset is a cluster of points which defines an object with human skin colour in the point cloud for each frame.

6. The combination of the step 4 and step 5 defines the set of objects (subsets) that represent human body parts. Consequently, in this step, a descriptor is computed for each subset. The descriptor classifies the kind of object according to shape. Thus, it is possible to recognize and locate the human hand in each frame.

7. Finally, the detected hand is modeled by a viewer process like a voxel. A voxel is a 3d mesh which contains all the points inside its coordinates. Thus, instead of managing thousands of points per frame, the algorithm manages a few voxels. This voxels are used to define a pseudo-skeleton of the hand. This skeleton is characterized by the centroid of the voxel and the vertex points to locate the end of the fingers.

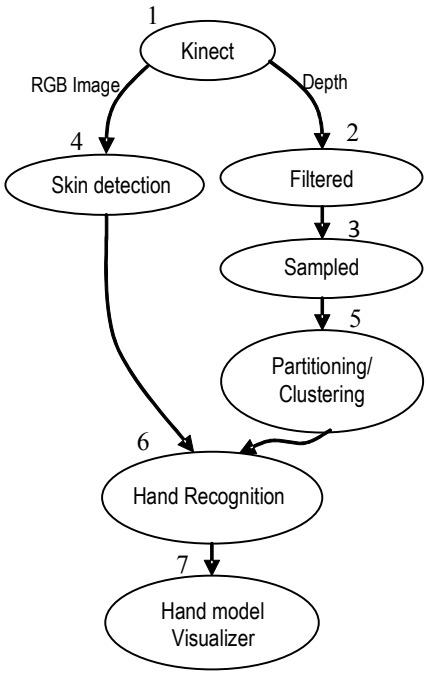

Figure 2: Flow-chart of method steps.

III. HUMAN-HAND RECOGNITION

The acquisition of range images, 3D, by using two sensors, RGB and IR, requires the synchronization of signals and a rectified process. Using OpenNI and PCL, this problem is solved and a point cloud is created. The point cloud defines a RGB-D image (Fig.3). However, the processing of point clouds causes either a high computational cost (processing time) or a high cost space (storage). In our case, the last one is not a problem because the modern computers have enough memory and we do not require the storage of large image sequences.

A. Filtered and sampled

At this stage, the drawback of high computational cost is minimized between 30% and 60%. This goal is accomplished by reducing the density of the point clouds in each frame. The percentage depends on the complexity of the scene. The number of objects, their features such as colour, texture, or shape and the ambient lighting determine this complexity. The filtered step eliminates the background (objects far away) and the information too close to the sensor (objects very close to lens) accordingly outside of the working range of Kinect.

To accomplish this, the algorithm uses the depth information, z, of each point, p of the point cloud. A distance bandpass filter is applied to initial point cloud as shown in equations (1) and (2).

$$\forall p \in PC_f : p(z) > 1.2m \wedge p(z) < 2.2m \quad (1)$$

$$PC_f \subseteq PC_o \quad (2)$$

where PC_o is the original point cloud and PC_f is the point cloud obtained as filtered result. Consequently, the number of objects in point cloud has been reduced. The working distance range, expressed in (1), was empirically chosen; the hand had to be observed with representative proportions but not taking in consideration close positions.

Figure 3: Original RGB-D Image, PC_o

Afterwards, in the sampled step, PC_f was filtered again. A distance criterion was used to reduce the density of point cloud. The sampling ratio, r_d, is a minimum of Euclidean distance between adjacent points in the point cloud. A value of r_d around 5mm provides good results in the detection process and it improves the detection speed in several indoor environments. Nevertheless, if this environment is very complex, this factor could be adjusted according to environment. Ratios, r_d between 5 and 10mm have been successfully tested.

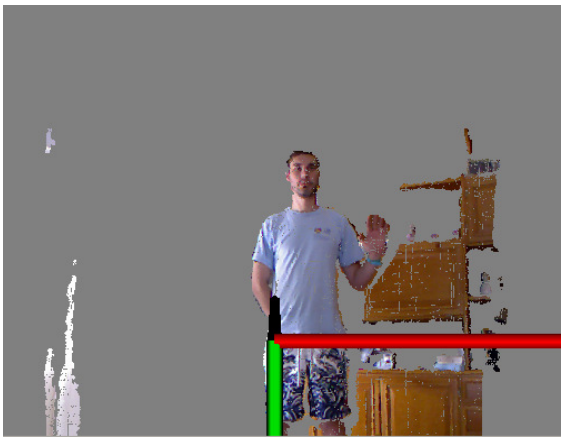

Figure 4: Filtered and sampled Image, PC_f

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. Thus, the density of PC_f usually is between 5 and 10 times smaller than PC_o. These values are appropriated to detect, recognize and localize the hand, by using colour segmentation and shape descriptors without causing discontinuity in the surface of objects. The discontinuities could alter the mesh which models their volume.

B. *Skin detection*

The objective of this step is to implement a colour segmentation to identify the set of points in the cloud which have a texture similar to human skin. The adopted approach uses the knowledge in traditional colour segmentation to narrow the search space in PC_f even more. In addition, other human body parts (such as hands, arms, legs, etc.) are unwittingly detected.

From the viewpoint of computer vision, there are many studies about the chromatic properties of human skin [15][16]. In general, they are based on the colour representation model such as RGB, HSV, HSL, LSM, etc. and colour histograms. Those can be simple or two-dimensional space by comparing two components of the colour model. In this method, a combination of two colour spaces, RGB and HSV, has been implemented. This approach achieves robustness and avoids false positives in the detection. Thereby, the skin detection is more independent of how the environment is illuminated and how the objects reflect the light beams.

According to the studies proposed in [11], the rule to segment human skin when the scene is lighted with natural light (sunlight) can be written as:

$$\begin{aligned}&(r > 95) \wedge (g > 40) \wedge (b > 20) \wedge (\max\{r,g,b\} > 15) \wedge \\ &\wedge (|r - g| > 15) \wedge (r > g) \wedge (r > b)\end{aligned} \quad (3)$$

where r, g and b are the intensity values of each pixel in RGB image from range image. They are between [0, 255].

Moreover, this rule (3) can be rewritten as (4) when the scene is lighted with artificial light.

$$\begin{aligned}&(r > 220) \wedge (g > 210) \wedge (b > 170) \wedge \\ &\wedge (|r - g| \leq 15) \wedge (r > b) \wedge (g > b)\end{aligned} \quad (4)$$

Hence, a pixel is considered skin when it satisfies any (3) or (4), so the skin points are given by:

$$\text{Equation } \{3\} \cup \text{Equation } \{4\} \quad (5)$$

In addition, according to the studies proposed in [15, 17], HSV segmentation attenuates the dependences between colour tone and the illumination. The component of hue is not dependent of saturation and intensity. Thus, our method considers that a point is skin when its RGB pixel is transformed to HUE pixel and it is limited as follows:

$$h < 33 \wedge h > 310 \quad (6)$$

where h is the hue value of each pixel in RGB image from range image. A hue value must belong to [0, 360]. Lastly, rules (5) and (6) are summarized in (7) to decide if a point belongs to a skin region or not.

Equation {5} ∩ Equation {6} (7)

The result of this stage is shown in Fig. 5.

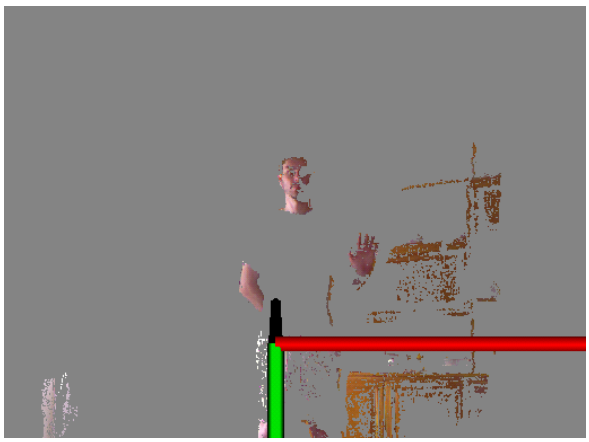

Figure 5: Skin detection by means both RGB and HSV, PC_s

C. Hand Descriptor

Subsequently, PC_s which is the point cloud computed on figure 5 is partitioned using a k-d tree. Thus, the skin candidate points are organized as subclouds:

$$\{PC_{c1}, PC_{c2} \ldots PC_{cn}\} \in PC_s \quad (8)$$

where each subcloud, PC_i, represents the volume of a candidate object. Every subcloud has to fulfill a quality factor in order to be considered as candidate object. This factor is the density of neighbor points, p_i, (distance between points and number of points). Fig. 6 shows the hand object is coloured after it was detected as a good candidate object from the whole set, PC_s.

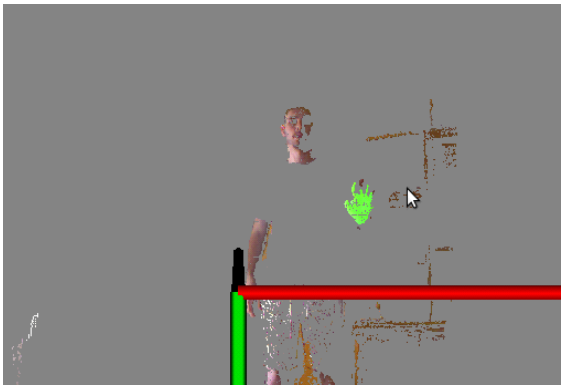

Figure 6: Skin detection by means both RGB and HSV, PC_s

Finally, in this last stage of the proposed method, a shape descriptor is calculated as a model hand to identify this type of object in the scene. Then, from the candidate objects, according to (8), the algorithm determines the number of hands in the scene and their locations. Afterwards, the 3d centroid is computed for each candidate object represented by each subcloud. It is computed as follows:

$$\bar{x} = \sum_x \sum_y \sum_z x \cdot p_i / size(PC_{ci}), \bar{y} = \sum_x \sum_y \sum_z y \cdot p_i / size(PC_{ci})$$
$$\bar{z} = \sum_x \sum_y \sum_z z \cdot p_i / size(PC_{ci}) \quad (9)$$

Later, the 2D convex-hull is computed for each PC_i. In this part of the algorithm, we have dispensed with the depth. The 2D convex-hull for the set of points of the subclouds is the minimal polygonal convex shape containing PC_i. Geometrically, V_i are the vertices which define the polygonal contour in the convex-hull i, and they are used as descriptor in order to identify the shape of PC_i (Fig. 7a).

In other words, a candidate object is considered as a hand if the set of points, PC_i satisfies the next two constraints.

On the one hand, its set of V_i must be greater than 7 and smaller than 15. At best, the hand shape model is represented by 7 vertices: 5 are at the end of the fingers and 2 are in the wrist (joining the hand and the forearm). The maximum limit of vertices is 15 because the joint position of the hand and a 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 algorithm tests the structure of V_i and the vertices not keeping a Euclidean distance in three-dimensional space of PC_i are removed. (Fig. 7b).

Consequently, the vertices near the neighborhood are considered as false vertices. Thereby, the 3D-Euclidean distances among the vertices and centroid are computed. Then, the vertex farthest to the centroid represents the middle finger, if it has two neighboring vertices (fingertip of first and ring fingers). In addition, the vertices which represents first and ring fingers must have two neighboring vertices, too. Against, the vertices which represent thumb and pinky fingers, respectively, must only have one neighbour vertex.

On the other hand, the geometrical model of the human hand keeps a relation among fingers and palm. That is to say, the model defines the distance ratio among fingers and palm. Therefore, from the centroid which determines the position of palm (bone structure which consists of metacarpals and carpals) and from the vertices which determine the position of fingers (distal phalanges) are removed the vertices which do not identify the ends of the fingers (Fig. 7c).

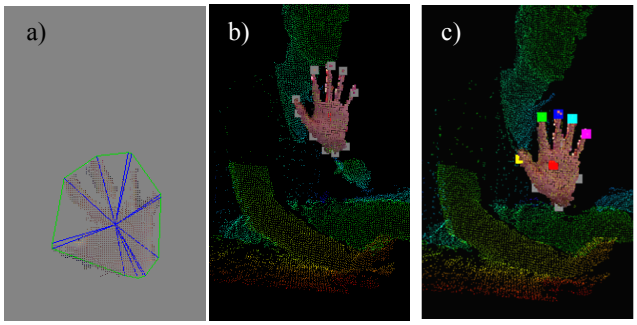

Figure 7: a) Computing the hand descriptor. b) and c) Detection of fingertips and palm.

The method works in real time. The estimated runtime is between 20 and 30ms in indoor environments such as domestic rooms and university laboratories (Fig. 8). To detect a human hand, the method requires that the hand is initially open, also the detection does not depend on its position and orientation whatever indoor environment. This method does

not use the human skeleton recognition to limit the search space in order to detect the hands.

Fig. 8 shows the recognition in several indoor environments. The first figure represents some human limbs in a same scene at similar distance to the Kinect sensor. The second represents the human hand positioned at a complex background because it has a colour similar to human skin.

Figure 8: Experiments in some environments

IV. HUMAN-ROBOT INTERACTION

A. *Robot-Hand Model*

The Barrett Hand [18] is an under-actuated robotic hand which is widely used as an end-effector of current robotic manipulators. This hand is composed of three articulated fingers and a palm which integrates four servomotors and the control electronics. The two phalanges of each finger are driven by one of the servomotors (the corresponding rotation axes are shown in Fig. 9) while the bases of fingers 1 and 2 rotate around the palm in a spread movement which is generated by the fourth servomotor. In addition, this hand contains a breakaway system which decouples the inner and the outer phalanxes when the inner phalanx contacts an object with a specific force, so that an enveloping grasp of the object is possible.

In order to map the human hand configuration to the robotic hand, the relative Cartesian position of the fingertips of the human has been used to determine the joint configuration of the robotic hand. In particular, the XYZ coordinates of the fingertips of the thumb, first and middle fingers with respect to the palm of the human hand are used as input to the inverse kinematics equations of the Barrett Hand (see [14] for a detailed description of these equations). Then, the corresponding joint angles of the three fingers are obtained and applied to the robotic hand.

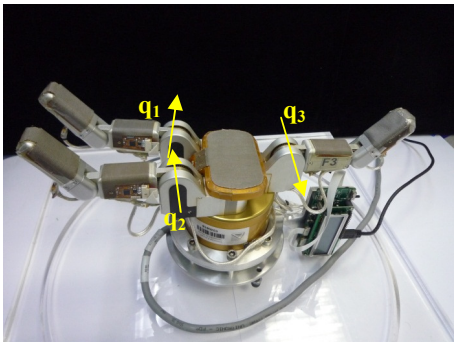

Figure 9: Barrett Hand used for human hand imitation.

B. *Communication with Robot Hand*

In order to implement the algorithms described in this paper, the ROS (Robot Operating System) software platform [19] has been used. This platform enables the development of robot software by providing similar services to an operating system such as hardware abstraction, inter-process communication and library integration. This integration permits ROS to cover most robotics fields: software simulation (Gazebo), stereo vision (PCL), computer vision (OpenCV), robot navigation, robot controllers, path planning, grasp planning (OpenRave), etc.

The distributed architecture of the ROS system is used to combine the kinematic control of the robotic hand with the human hand recognition. In fact, these functionalities are implemented into two ROS nodes: one for the hand control and the other one for the visual acquisition and processing of the hand recognition. The diagram in Fig. 10 shows the software architecture developed in ROS in order to communicate both nodes.

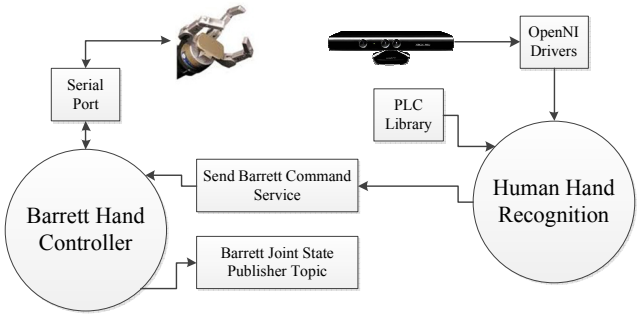

Figure 10: Node architecture in the ROS platform.

The first ROS node acts as server of the kinematic control of the robotic hand and is executed in a computer which is directly connected to the robotic hand through the serial port. This node implements a service which receives the joint angle of the robotic hand which has to be modified and sends the corresponding command through the serial port in order to execute it. In addition, 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 on real-time.

The second ROS node is a client of this service which implements the human hand recognition algorithm and which can be executed in another different computer. In particular, this second ROS node is connected to the Kinect camera by a USB port, communicates with the camera through the

OpenNI drivers [4] and uses the PCL library [5] in order to implement the point cloud processing functions previously described. The communication between both nodes is implemented through the standard message-based architecture of the ROS system.

C. *Experimental Results of Human-Robot Mapping*

To evaluate the human-robot interaction, some movements have been imitated. The fingertip positions (thumb, index, and middle) have been mapped to the robotic fingers.

Fig. 11a shows an imitation of a claw hand position. In this case, the fingertips detected from RGBD-image change its positions in relation to the palm and the fingers from Barret hand are raised, together. Fig. 11b shows an imitation of an extended hand position. In this other case, the fingertips change their positions again, and the robotic fingers are opened and so the phalanges increase their joint positions. In addition, in this last experiment, the human thumb is displaced away from the index while keeping the depth plane. In this last case, the third robotic finger according to Fig. 9, is moved and it is separated from the other two.

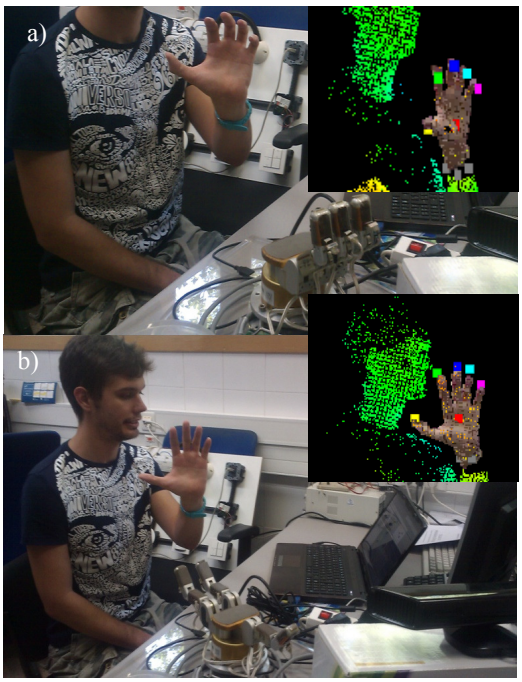

Fig 11: a) Closing hand. b) Opening hand.

V. CONCLUSION

This paper presents a method for detecting and tracking hands, by using a Kinect camera and acquiring and processing in real time. The implemented method was tested for different configurations of the hands, with different size and color. 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 does 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. Later, simple gestures have been analyzed and measured to imitate the human movements with a Barret robotic hand. Only three fingers of the detected human hand anatomy were used in this task due to the restriction of the anthropomorphic structure of the Barret hand. To do this, an emerging and versatile communication platform such as ROS was used to program the master-slave structure.

REFERENCES

[1] Microsoft Corp. Redmon WA. Kinect for Xbox 360. Disponible en: http://www.xbox.com/kinect

[2] Kinect SDK (2012). Disponible en: http://research.microsoft.com/enus/um/redmond/projects/kinectsdk

[3] OpenKinect (2011). Disponible en: http://openkinect.org/wiki/Main_Page

[4] OpenNI (2011). Disponible en: http://www.openni.org

[5] Rusu, R.B. and Cousins, S., (2011) "3D is here: Point Cloud Library (PCL)". Proc. of International Conference on Robotics and Automation (ICRA), Shanghai, China, IEEE Press.

[6] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., Blake, A., (2011) "Real-Time Human Pose Recognition in Parts from Single Depth Images". Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Colorado Springs, USA, IEEE Press.

[7] Bellmore, C., Ptucha, R., Savakis, A., (2011). "Interactive Display Using Depth and RGB Sensors for Face and Gesture Control", Proc. IEEE Western New York Image Processing Workshop (WNYIPW), Rochester, USA, IEEE Press.

[8] López-Méndez, A., Alcoverro, M., Pardàs, M., Casas, J.R. (2011). "Real-Time Upper Body Tracking with Online Initialization using Range Sensor". Proc. IEEE International Conference on Computer Vision Workshops, IEEE Press, pp. 391-398

[9] Keskin, C., Kiraç, F., Emre, Y., Akarun, L. (2011). "Real Time Hand Pose Estimation using Depth Sensors". Proc. IEEE International Conference on Computer Vision Workshops, IEEE Press, pp. 1228-1234

[10] Malassiotis, S., Strintzis, M. (2008). "Real-Time hand posture recognition using range data". Image and Vision Computing, 26(7), pp. 1027-1037.

[11] Ren, Z., Yuan, J., Zhang, Z. (2011). "Robust hand gesture recognition based on finger-earth mover's distance with a commoodity depth camera". Proc of the 19th ACM International Conference on Multimedia, ACM Press, pp. 1093-1096.

[12] Oikonomidis, I., Kyriazis, N., Argyros, A.A. (2011). "Efficient Model-based 3D tracking of hand articulations using Kinect". Proc of the British Machine Vision Conference (BMVC), BMVA Press, pp. 101.1-101.11.

[13] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T.B., Leibs, J., Wheeler, R., Ng, A.Y. (2009). "Ros: an open-source robot operating system". Proc of International Conference on Robotics and Automation (ICRA)- Workshop on Open Source Software, Anchourage, USA, IEEE Press.

[14] Corrales, J.A., Jara, C.A., Torres, F. (2010). "Modelling and simulation of multifingered robotic hand for grasping task". Proc of International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, Malaysia, IEEE Press.

[15] Nusirwan Anwar bin Abdul Rahman, Kit Chong Wei and John See "RGB-H-CbCr Skin Colour Model for Human Face Detection". Faculty of Information Technology, Multimedia University.

[16] P. Peer, J Kovac, F. Solina, "Human Skin Colour Clustering for Face Detection", EUROCON1993. Ljubljana, Slovenia, pp. 144-148, September 2003.

[17] Corrales, J.A., Gil, P., Candelas, F.A. Torres, F. (2009). "Tracking based on Hue-Saturation Features with a Miniaturized Active Vision System". Proc of 40th Internacional Symposium on Robotics (ISR), Barcelona, Spain, pp. 107-112, March 2009.

[18] Townsend, W. (2000). "The BarrettHand grasper – programmably flexible part handling and assembly", Industrial Robot: An International Journal, 27(3), pp. 181-188.

[19] ROS (2012). Available at: http://www.ros.org/