CogLaboration

Collaborative Project

FP7 – 287888

D3.3 Human motion tracking

Lead Author: Tecnalia

With contributions from: Treelogic

Reviewer: SSSA

Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	30/06/2013
Actual delivery date:	15/07/2013
Version:	Draft 0.9
Total number of pages:	32
Keywords:	Openni package, ROS, colour segmentation, tf, Coglaboration industrial scenario.

Abstract

In this document we describe the tracking algorithms that have been developed to estimate the human position in the three different ergonomic configurations as defined within the Coglaboration [1] industrial scenario (refer to Deliverable 6.10). Based on the available tools (Robot Operating System (ROS) [21] and the commercial motion sensing input device *Kinect X360*), several tests have been conducted for the detection and the monitoring of the human body posture. The main objective of the human tracking module is to provide at each instant the location of the human hand, information that is required by the cognitive controller to adjust online the motion of the robotic system and therefore achieve the physical interaction with the human.

The shelf tracking solution provided with the Kinects sensor through the openni package permits to extract the Human body skeleton at video rate (30 Hz). Nevertheless, that tracker requires the person so stand in front the camera, configuration not respected by the “Lying under the car configuration”. Furthermore, the *openni_tracker* tends to provides noisy result when human limbs get close to another element from the environment (as could be the robotic arm getting closer to the robot). This is why we are proposing here to complement this tracking with some colour-based solution, with the objective to get a more robust person tracker. Since the control layer is mainly interested in the location of the human hand, we are here focusing on the location estimation of the user hands along time.

The solution proposed to perform a successful tracking combines the following techniques: skeleton tracker, Kalman filters, face detection and segmentation by colour. Furthermore, different ROS functions have been used, tested and combined in order to reach the best performance in terms of execution time and correct detection of the hands location.

Two different algorithms are proposed: *Openni_tracker* with HSV colour space plus proximity selection (OHSV_Prox) and *openni_tracker* with YCbCr colour space plus face detection (OYCbCr_Face). Different colour spaces were tried, but none of them permitted to get a consequent processing time reduction with respect to the others. The combination of the openni_tracker with proximity selection or face recognition permitted to assure the hand location. Important conclusions about the face recognition method applied to Coglaboration scenario have been summarized.

The worst case scenario for the execution time is presented for both algorithms. This measure is an indicator used in the real control applications design because it indicates the minimum frequency update of the hand location. Preliminary tests showed that the worst case for OYCbCr_Face had a frequency of about 3.62 Hz (0.27 s) while with the OHSV_Prox that worst case is around 21 Hz (0.04 s).

Executive summary

Task 3.3 is related to the vision-based tracking of the human partner being involved in the exchange with the robotic system. In that context, we are particularly interested in the detection and tracking of the human hand locations along time, since it is the main information driving the motion of the robotic system in the current development of the cognitive controller within WP4.

The perception modules are based on the use of the RGB-D Kinect sensor. The developments have been based on off-the-shelves solutions that are mainly the Robot Operating System (ROS [2] for the management of the different processing flows and the access to the human skeleton tracking, and the OpenCV library [5] for vision processing.

The openni package permits to process the visual and depth information from the Kinect sensor and to extract from it the location of a person in front of the camera as well as an estimation of the location of his limbs, or his skeleton. Nevertheless the tracking performed presents some limitations for our application. We are proposing in this document to improve this technique by combining it with a parallel node processing colour-based tracking of the human hand. We have been developing two alternatives (OHSV_Prox and OYCbCr_Face) to perform the tracking, and the current results obtained are better with the OHSV_Prox (a combination of *openni_tracker* and Kalman filter with colour segmentation and proximity selection) described in this document.

Document Information

IST Project Number	FP7 - 287888	Acronym	CogLaboration
Full Title	Successful Real World Human-Robot Collaboration: From the Cognition of Human-Human Collaboration to the Cognition of Fluent Human-Robot Collaboration		
Project URL	http://www.coglaboration.eu/		
Document URL			
EU Project Officer	JuhaHeikkilä		

Deliverable	Number	D3.3	Title	Human motion tracking
Work Package	Number	WP3	Title	Scene and situation understanding

Date of Delivery	Contractual	M20	Actual	M20
Status	version 9		final □	
Nature	prototype □ report X demonstrator □ other □			
Dissemination level	Public X restricted □			

Authors (Partner)	Tecnalia			
Responsible Author	Name	Mildred Puerto	E-mail	mildred.puerto@tecnalia.com
	Partner	Tecnalia	Phone	+34 946 43 08 50

Abstract (for dissemination)

In this document we describe the tracking algorithms that have been developed to estimate the human position in the three different ergonomic configurations as defined within the Coglaboration [1] industrial scenario (refer to Deliverable 6.10). Based on the available tools (Robot Operating System (ROS) [21] and the commercial motion sensing input device *Kinect X360*), several tests have been conducted for the detection and the monitoring of the human body posture. The main objective of the human tracking module is to provide at each instant the location of the human hand, information that is required by the cognitive controller to adjust online the motion of the robotic system and therefore achieve the physical interaction with the human.

The shelf tracking solution provided with the Kinects sensor through the openni package permits to extract the Human body skeleton at video rate (30 Hz). Nevertheless, that tracker requires the person so stand in front the camera, configuration not respected by the "Lying under the car configuration". Furthermore, the *openni_tracker* tends to provides noisy result when human limbs get close to another element from the environment (as could be the robotic arm getting closer to the robot). This is why we are proposing here to complement this tracking with some colour-based solution, with the objective to get a more robust person tracker. Since the control layer is mainly interested in the location of the human hand, we are here focusing on the location estimation of the user hands along time.

The solution proposed to perform a successful tracking combines the following techniques: skeleton tracker, Kalman filters, face detection and segmentation by colour. Furthermore, different ROS functions have been used, tested and combined in order to reach the best performance in terms of execution time and correct detection of the hands location.

Two different algorithms are proposed: *Openni_tracker* with HSV colour space plus proximity selection (OHSV_Prox) and *openni_tracker* with YCbCr colour

	space plus face detection (OYCbCr_Face). Different colour spaces were tried, but none of them permitted to get a consequent processing time reduction with respect to the others. The combination of the openni_tracker with proximity selection or face recognition permitted to assure the hand location. Important conclusions about the face recognition method applied to Coglaboration scenario have been summarized. The worst case scenario for the execution time is presented for both algorithms. This measure is an indicator used in the real control applications design because it indicates the minimum frequency update of the hand location. Preliminary tests showed that the worst case for OYCbCr_Face had a frequency of about 3.62 Hz (0.27 s) while with the OHSV_Prox that worst case is around 21 Hz (0.04 s).
Keywords	Openni package, ROS, colour segmentation, tf, Coglaboration industrial scenario.

Version Log			
Issue Date	**Rev. No.**	**Author**	**Change**
22/05/2013	001	Mildred Puerto	Draft Tecnalia
02/06/2013	002	Joaquin Canseco	Draft Treelogic
17/06/2013	003	Mildred Puerto	Trial data Tecnalia and document integration
19/06/2013	004	Joaquin Canseco	Trial data Treelogic
20/06/2013	005	Mildred Puerto	To review
25/06/2013	006	Marco Controzzi	Reviewed
05/07/2013	007	Mildred Puerto	Version corrected
09/07/2003	008	Anthony Remazeilles	Reviewed
14/07/2013	009	Mildred Puerto	Final version

Table of Contents

List of Figures

List of Tables

Abbreviations

API: Application Programming Interface

ROS: Robot Operating System

tf: Transform frame package in ROS

KUKA LWR: Light Weight Robot built by KUKA.

RGB-D: Depth sensor plus Colour model (Red, Green, Blue)

RGB: Colour model (Red, Green, Blue)

HSV: Colour model/colour space (Hue, Saturation, and Value)

YCbCr: Colour model/colour space (luma, blue-difference chroma, red-difference chroma)

Definitions

OpenNI framework: Open Natural Interaction framework for the development of 3D sensing middleware libraries and applications. It is an open source framework [5].

tf: Designs a package from the ROS environment. It keeps track of coordinate frames in distributed systems. Each process that has primary knowledge about the relationship between two coordinate frames broadcasts the relative transform for each pair of frames between which they have primary knowledge. When a transform between two specific coordinate frames is requested, tf uses the cached information to determine the minimum spanning set of transforms to get the motion between the requested frames. From this set of transforms it can compose the net transform between the requested frames [15].

Nodes: In the ROS context, nodes are processing units that perform computation. ROS has been designed to be modular, dividing a complex application into many nodes. A ROS node is written using a ROS client library, such as roscpp (for C++ language) or rospy (for python) [2].

Messages: Nodes communicate between them using messages. A message is simply a data structure, comprising typed fields. Standard primitive types like integer or floating point are supported, as there are arrays of primitive data types.

Topic: Messages are routed via a transport system with publish / subscribe structure (semantics). A node sends a message by publishing it to a specific topic. The topic is a name that is used to identify the content of the message. A node that requires some data needs to subscribe to the appropriate topic.
For a single topic, multiple concurrent publishers and subscribers can be occur, while for a single node can be published and/or subscribed to multiple topics.

OpenCV: Open CV **(**Open Source Computer Vision) is released under a BSD license and it is free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android operative systems.

Luminance or luma: Represents the brightness in an image (the "black-and-white" or achromatic portion of the image)

Chrominance or chroma: Represents the colour information of an image, separately from the accompanying luma signal.

1 Introduction

In a typical human-robot collaborative scenario, the knowledge by the robot of the human position is fundamental. This is true not only for safety reasons, but also to be able to successfully interact with it.

The solution proposed to perform the human tracking involves the combination of techniques provided by ROS like: skeleton tracker, face recognition and segmentation by colour. These techniques complement each other in order to ensure the location of a human hand, as specified in the CogLaboration project. Indeed the hand is obviously involved in the interaction process, by holding, grasping and releasing the object.

Human Robot interaction involves not only the hand, but also different body parts that could be close to the robot. The proposed algorithms explained in this document (OHSV_Prox and OYCbCr_face) also consider human safety, exploring the proximity of the human physical body and the differences between hand and face location (if an object moves approaching to a hand, the hand is adapted to catch it, but if the object approaches to the face, then it is perceives as a danger).

In this task, different ROS functions have been developed and tested with the final aim of tracking the human body position during the execution of the exchange tasks. Advantages and disadvantages of two different algorithms are presented in the present document.

2 Human motion tracking

In the Coglaboration project, human tracking was performed using a commercial Kinect camera. The processing of the acquired images combines several techniques, with the aim to ensure a proper detection of the human body and an estimation of the movement of the subject.

In the next section the issues identified with the off-the-shelf solution are described. Then a solution is introduced, and finally results applying the proposed method are presented.

The technical specifications of the system used to run ROS and OpenCV tools are the following ones:

- PC equipped with Ubuntu 12.04, 64-bits, GNOME 3.4.2 and Intel Xeon @3.2 GHzx8.
- ROS fuerte [2].
- VGA compatible controller: NVIDIA Corporation GF108.
- Kinect X360 [3].

2.1 Basic approach: Skeleton tracking

In computer vision the detection of the human structure is very challenging. In addition to the difficulties introduced by the movements of the subject and to his complex kinematics, the deformability of the human body has to be taken into account in order to get accurate information. The objective is to ensure human detection with an efficient and robust method able to provide information to the robotic system. Human detection, tracking and motion analysis are still an open research area.

2.1.1 Openni_tracker

The OpenNI framework provides a set of open source Application Programming Interfaces (APIs) for physical devices and several middlewares [4]. The interface selected for the Coglaboration experiments is the NITE middleware, since it automatically performs the skeleton tracking from the data acquired by the Kinect camera (Figure 1).

The *Openni_tracker* [5] estimates the human skeleton posture as a set of 15 human body joints (Figure 2) [5, 6]. The openni driver provided in ROS works in structured environments (i.e. with the person in front of the camera, without obstacles between the person and the camera view) and it provides the skeleton tracking from the image captured by the Kinect (Figure 2-2.b).

Openni_tracker runs the *robot_state_publisher* which captures the joint states on a topic and publishes the transform matrixes (**tf**) [15] (considering that a transform matrix contains the rotation and the translation components that permit to express the Euclidean motion in between two frames from the space).

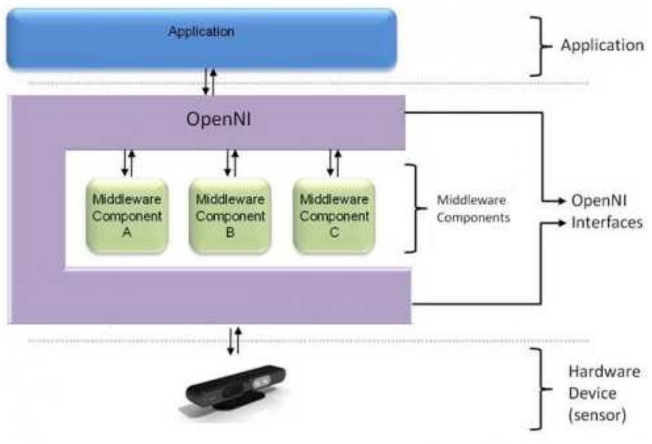
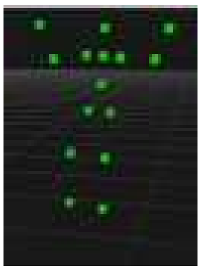


Figure 1.OpenNI utility for ROS developers using the Kinect.

Openni_tracker (Figure 2) estimates the human skeleton pose (joint positions and joint angles) and gets it accessible in the ROS environment through the **tf** node that keeps track of movements of the coordinate frames over time. Through this coordinate frame manager, it is possible access the posture of the 15 tf related to the human skeleton (Figure 2, 2.a).

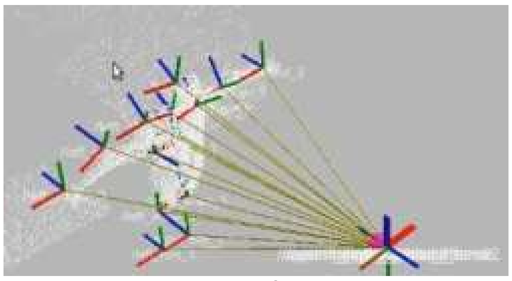
Openni_tracker publishes 15 tfs:

1. /head
2. /neck
3. /torso
4. /left_shoulder
5. /left_elbow
6. /left_hand
7. /right_shoulder
8. /right_elbow
9. /right_hand
10. /left_hip
11. /left_knee
12. /left_foot
13. /right_hip
14. /right_knee
15. /right_foot



The 15 tfs of openni_tracker

2.a



Openni_tracker tf: Frontal view

Frontal view perception

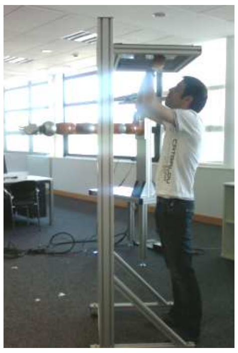
2.b

Figure 2: General information about the *openni_tracker* function used in the Coglaboration project. Left: name of the 15 frames being related to the human skeleton. 2a: illustration of the pose of the 15 mentioned frames on the human body. 2b: comparison of the image acquired and the skeleton estimated. The transforms are computed with respect to a frame associated to the camera (on the right of the image).

The human motion tracking needed by the robotic system has to estimate from the visual flow the location of the human limbs, focusing mainly on the location of the human arm used for the object exchange. The estimation of the human location is initially performed using the tracking facilities provided with the Kinect sensor and accessible from the ROS environment through the *openni_tracker package* [5]. In particular, as the person is not in front of the camera sensor, and the human body is not completely observed by the Kinect, the *openni_tracker* does not provide reliable information and it turns out that such pose estimation presents some stability issues in some of the previously defined configurations (Figure 3). One of the challenges of the Coglaboration project is related to the features extraction when the subject is partially occluded, such as in the "lying under the car" scenario.

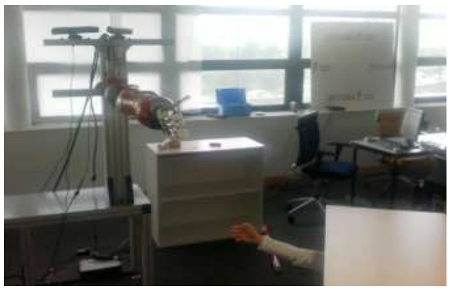
Working in the engine bay

3.a



Working under hydraulic ramp

3.b



Lying under the car

3.c

Figure 3: Different human body positions in the Coglaboration industrial scenario.

Using the camera mounted onto the robotic system, we have been performing some evaluations of the standard tracking approaches, considering the three defined configurations. In the results obtained, we have identified two main problems:

1. Lying under the car issues: the *Openni_tracker* is not able to track the subject's arm when he moves under an obstacle (Figure 4). In that situation, *Openni_tracker* does not show reliable information. It works fine when the whole body is visible or if the upper part of the body (down to the knees) is within the camera field of view.

   The Figure 4 shows the sequence of a human while he/she is moving under the car. Even if all the skeleton frames are located when the person is rotated and is made go under the car (4a), when the arm is the only part of the body that the camera can see (the movements of the arm can be retrieved from the different position of the point cloud (4b-4c-4d sequence)), the *openni_tracker* cannot recognize it.

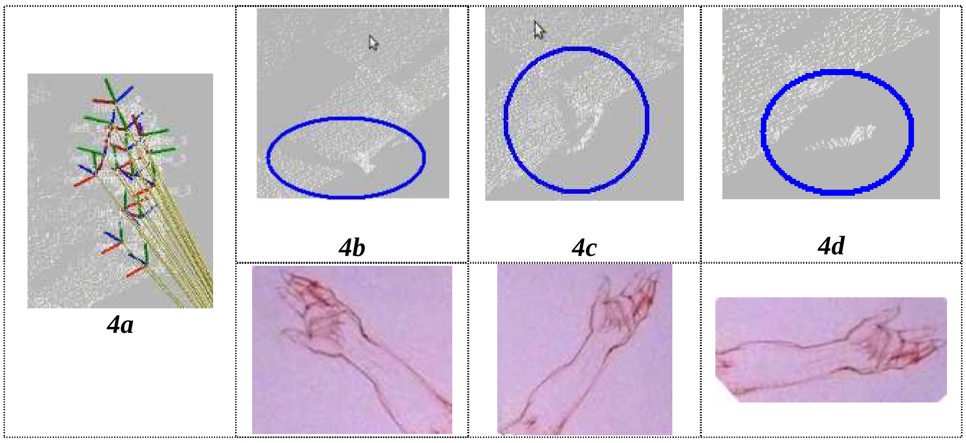
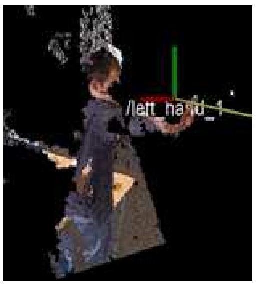
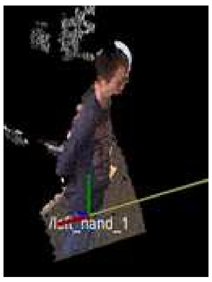


Figure 4: Detected problem in Coglaboration scenario. The point clouds shown in the figures indicates the human arm while the person 'under the car'. When the subject moves under an obstacle, the openni_tracker is not able to estimate the location of the human skeleton.

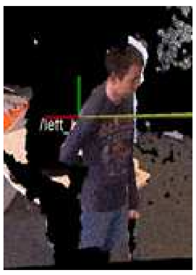
2. Incorrect skeleton detection: the estimated frames are unstable. It happens in special circumstances, for example when the arm is occluded (Figure 5), the depth map becomes a bit unreliable and the algorithm get sometimes confused and mix the frames of the arms, or detect the elbow instead of the wrist between the arms, the elbow and wrist joints. In some cases the arm frames are not updated and remains in their last positions. This problem has been solved using information from the colour segmentation technique (a similar problem was detected in [16]).



Correct location



False location: The hand is hidden



False location: The hand is hidden but it is wrongly located at the level of the shoulder.

Figure 5: Incorrect skeleton detection in Coglaboration demonstration scenario.

In order to solve the mentioned issues, we are proposing to combine the skeleton tracking with a colour based tracking of the hand by segmenting the image with respect to the human skin colour.

2.2 Robust hand tracking

Several approaches can be selected to detect human body parts, like forms detection or pattern detection, but since the hand changes its postures in a very complex manner and a complex kinematical architecture is involved, the colour-skin segmentation seems to be a simple and effective manner to detect it.

In order to reach robust hand detection, the clusters found using the colour-skin segmentation results must distinguish between the hand and the face. Because of that, proximity selection and face detection techniques have been implemented after the skin selection.

2.2.1 Colour segmentation

The image segmentation is the process of partitioning a digital image into multiple segments [19]. The segments can be selected in many ways, for example, identifying objects by its primitive shape (such as cylinders, spheres, rectangles and so on) or detecting boundaries through the colours [20]. The goal of the segmentation is to simplify and/or change the representation of an image into a space that is more meaningful and easier to analyse (some examples of segmented images can be found in Figure 6 and Figure 9**).**

The idea of segmenting the skin coloured parts of the image to find the human hands is simple and efficient but has some drawbacks that have to be considered. First, it is not possible to define which of the segmented parts of the image belongs to the hand involved in the object exchange movements. In addition, the scenario can have surfaces showing reflectance properties similar to the human skin [17].

To avoid the first issue, we assume that the area presented a colour skin nearest to the robot corresponds to the hand that will participate in the exchange. Unfortunately this assumption leads to an additional issue to be solved: in some of the postures of the subject, the human face can be nearest to the robot than the hands, which would make the face badly detected as the subject’s hand (an additional reason to implement the face recognition).

To solve the second problem (that means false positives caused by the presence of objects or background with similar colour characteristics than the human skin) we used various skin filters implemented using different colour spaces (Figure 6). Depth segmentation and clustering techniques can be also useful to remove undesired results [14].

To implement the previously explained segmentation by colour we have designed an algorithm working directly on the RGB-D image to get the 3d points of the scene with similar skin colour characteristics.

Thanks to those algorithms the system performance has been improved by avoiding a further conversion from the segmented image to the point cloud (which combines each pixel to the 3d point of the cloud).
In addition, having depth information, such as using 3D reconstruction, lets us to solve the typical overlapping problems, which are mistakes occurring when there are multiple skin coloured segments near each other, which one is in the background and others are in the foreground. In that situation two segments can be incorrectly identified as the same one, as if we were working with 2d image data. It is an additional advantage of our approach.
Although there are several techniques able to classify pixels (or points) into skin coloured or not skin coloured, (such as: bayes classifiers, lookup tables, self-organized maps (SOM), modelling skin distributions with Gaussians filters [16]) a simple thresholding method is used to keep low the computational cost of the algorithm.
We consider that an exchange task requires a tracking precision of the magnitude of centimetres. Indeed we assume that small errors in the robot positioning should be overcome by the human subject without affecting the fluency of the exchange (due mutual adaptation is an important phenomenon in human-human

interaction, it has been investigated the adaptation of the human to the robot using the natural ability of humans to adapt to robots is recently becoming increasingly attractive [22]). This is why we consider that a simply thresholding method, combined with the output of the openni_tracker module, should be precise enough for the exchange purposes.

There are many colour spaces suitable for thresholding the skin colour characteristics. A brief explanation of the colour spaces we use is presented here:

- **Normalized RG:** Since the RGB space does not have an explicit separation between chroma and luminance, the normalization tries to minimize the dependence on the luminance value.
  Normalized RG is a normalization of the classical RGB colour space where $R+G+B = 1$. Here only two of the three values are necessary, and normally R and G are selected since the human eye has a minimal response to the blue light.

- **HSV:** is a colour space that separates colour information (chrominance) from intensity information (luma). *Hue* is the chroma attribute describing the pure colour, the *saturation* defines the amount of white light mixed with a hue, while the *value* explains the brightness of the image. For the efficient conversion from RGB to HSV colour space.

- **YCbCr:** is a colour space with unambiguous separation of *luminance* and *chrominance* suitable for skin segmentation. *Y* represents the *luminance*, while *Cb* and *Cr* are respectively the blue-difference and red-difference *chroma* components [12].

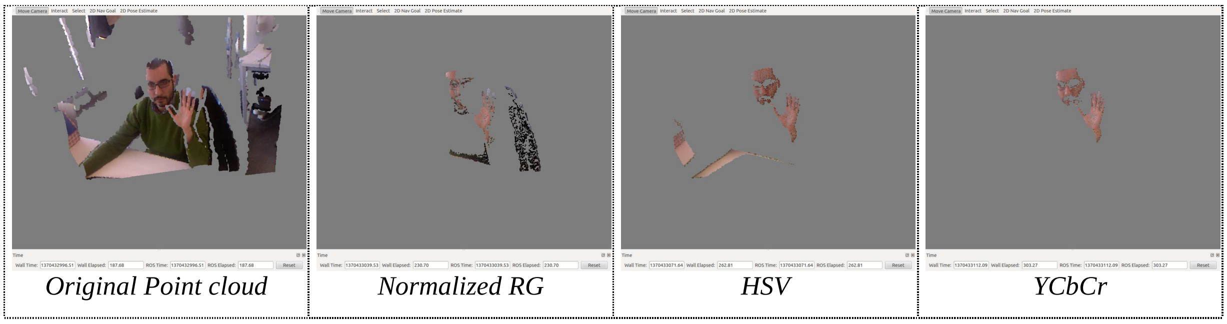


Figure 6. Different colour spaces applied to the Human Tracking detection.

We have been comparing the processing time of the mentioned colour spaces. To do so, we have measured the frequency of each function and the variability of its coefficients (mean of all frequencies in a 5 minutes test). As it can be seen on the following tables, all types of colour spaces have similar performances. The important factor to implement is the depth segmentation that decreases the points to process by the colour filters and improves the quality of the response as well as the processing time.

We have also tested the image pixel resolution. The most standard format are QVGA (Quarter Video Graphics Array) which has a resolution of 320 x 240 pixels and VGA (Video Graphics Array), of resolution 640 x 480. We can see that pixel resolution does not affect the execution time of the node when is used the depth information (refer to RGB-D image).

RG-Normalized colour space	VGA	QVGA
Hand detection with depth segmentation	27.28 Hz	24.68 Hz
Hand detection with *openni_tracker* fusion with depth segmentation	25.01 Hz	24.4 Hz
Hand detection with *openni_tracker* fusion without depth segmentation	3.62 Hz	21.74 Hz
HSV colour space	VGA	QVGA

Hand detection with depth segmentation	29.67 Hz	24.68 Hz
Hand detection with *openni_tracker* fusion with depth segmentation	24.3 Hz	24.4 Hz
Hand detection with *openni_tracker* fusion without depth segmentation	11.82 Hz	23.3 Hz

YCbCr colour space	VGA	QVGA
Hand detection with depth segmentation	29.42 Hz	29.87 Hz
Hand detection with *openni_tracker* fusion with depth segmentation	23.56 Hz	29.54 Hz
Hand detection with *openni_tracker* fusion without depth segmentation	7.84 Hz	29.66 Hz

Table 1: Frequency measurements of different colour spaces using combined with the *openni_tracker* and two image pixel resolutions.

In the trials performed with laboratory light conditions, the best result (with no false positives) was obtained employing the skin filter based on the YCbCr colour space previously described.

The next figure represents the variation of the different values of the HSV-based segmentation parameters (Figure 7) along the day in different days. Values can vary, but the algorithm performs an automatic tuning with the skin-colour captured in each frame, so it makes a colour optimization for correct detection of the cluster (which is the meaning of Hfinal, Sfinal and Vfinal).

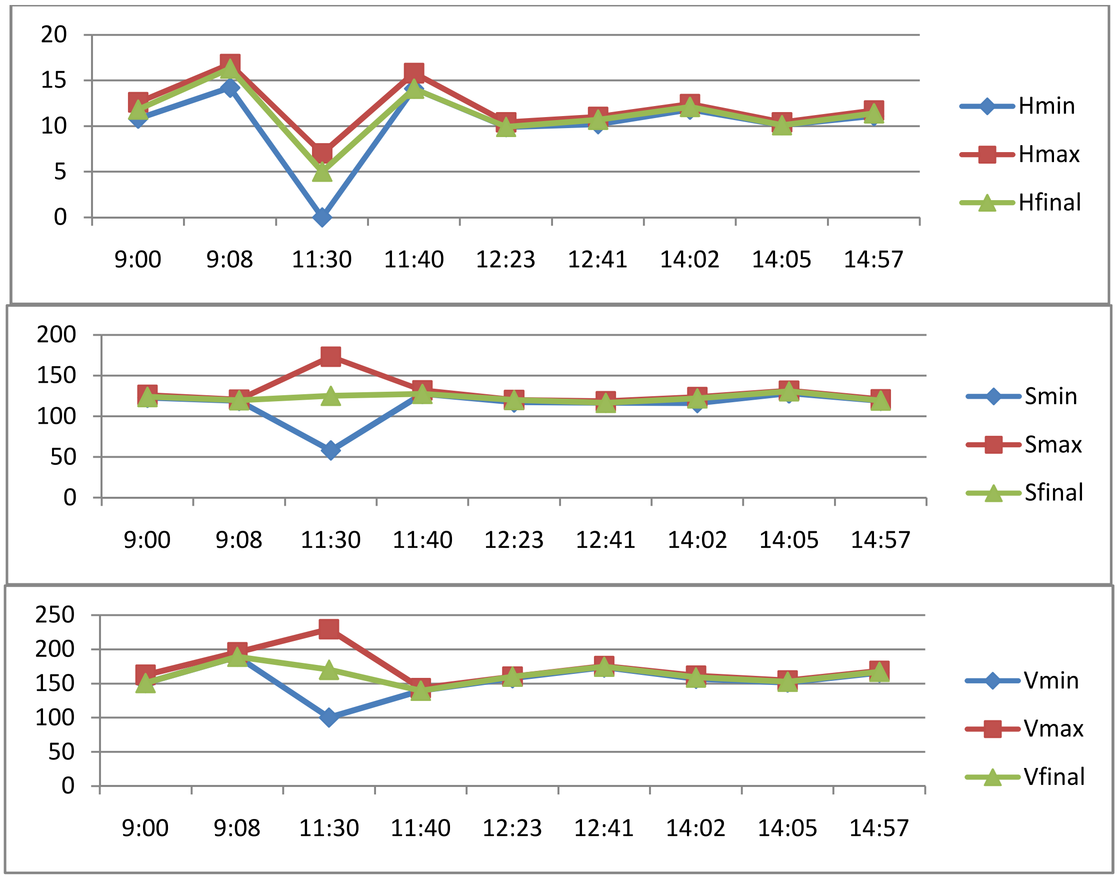


Figure 7: HSV coefficient variations. It involves indoor conditions in cloudy days being used by human users with clear skin colour.

2.2.2 Face Filtering

Depending on the skin clusters found, the detection may involve the face filtering procedure. After the skin colour segmentation, the face cluster can be confused with a hand, so there is the need of identify the face of the subject and to remove its cluster from the segmented scene.

The first solution investigated was to use a face detector to find faces within the image captured by the Kinect camera and then an algorithm able to remove them from the scene.
Two different face detectors packages (available from ROS) have been tested: the *face_detector* and the *pi_face_tracker* (Figure 8). The first package is based on the OpenCV Haar cascade classifier for frontal faces (this algorithm has been adapted to work with RGB-D)[12]. The second package uses the OpenCV's Haar face detection, and then combines it with a point tracker (the point being distributed onto the face detected) to follow the head. The point tracker is based on the classical differential image point tracker proposed by Lucas and Kanade [18].

A summary of the observed response of both face detectors is shown in the table. It has been tested with different resolution searching for the best conditions for implementing the functions in the human tracking algorithm. However VGA and QVGA did not help to the time execution or reliability of *face_detector* and *pi_face_detector* in the current conditions.

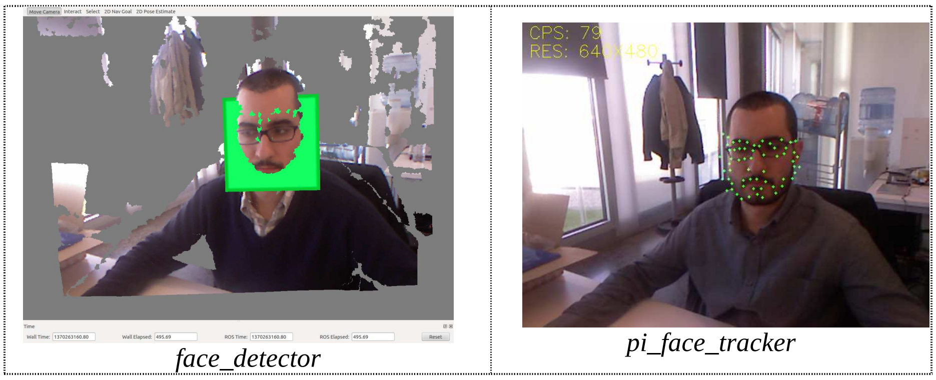


Figure 8. The two face detectors tested
(already available from the ROS)

Face_detector Day 1	VGA	QVGA
Only indoor light (little light)	3.62 Hz	21.73 Hz
With outdoor light in front of the camera (behind the face)	4.507 Hz	1.4 Hz *
With lateral outdoor light	5.6 Hz	19.4 Hz
With outdoor light behind the camera (in front of the face)	6.47 Hz	23.7 Hz

Face_detector Day 2	VGA	QVGA
Only indoor light (little light)	4.7 Hz	22.16 Hz
With outdoor light in front of the camera (behind the face)	4.01 Hz	2.3 Hz *
With lateral outdoor light	5.41 Hz	21.09 Hz
With outdoor light behind the camera (in front of the face)	6.97 Hz	23.82 Hz

Pi_face_tracker Day 1	VGA	QVGA
Only indoor light (little light)	0.141 Hz	0.623 Hz
With outdoor light in front of the camera (behind the face)	0 Hz *	0 Hz *

With lateral outdoor light	0.505 Hz	0.616 Hz
With outdoor light behind the camera (in front of the face)	0.453 Hz	0.835 Hz

Pi_face_tracker Day 2	VGA	QVGA
Only indoor light (little light)	0.209 Hz	0.512 Hz
With outdoor light in front of the camera (behind the face)	0 Hz *	0 Hz *
With lateral outdoor light	0.425 Hz	0.53 Hz
With outdoor light behind the camera (in front of the face)	0.74 Hz	1.314 Hz

Table 2: Frequency measurements of two face detectors using different lighting conditions and two image pixel resolutions.

*No face detected or only detected in sporadic frames.

The *Pi_Face_tracker* turns to be too much time consuming (and presented as well a low reliability and performance during our testings),. We therefore discarded it and focused mainly on the *Face_detector*.

3 Proposed solutions: OHSV_Prox and OYCbCr_Face

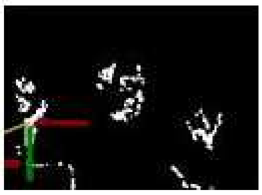
Two algorithms have been made using the techniques explained before: The OHSV_Prox and OYCbCr_Face. The OHSV_Prox algorithm uses the HSV colour space combined with a Kalman filter estimation and a proximity selection of the hand, its tf is compared with the tf given by the openni_tracker. The OYCbCr_Face uses an additional colour space: the YCbCr and to differentiate all the clusters of skin, a face detector has been also implemented.

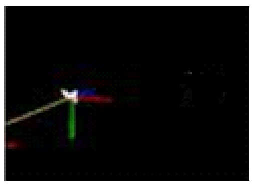
For human motion tracking focused on the hand, the present document is using only the Cartesian location of the tfs generated by the openni_tracker, colour segmentation and third node calculation. The axes x-y-z were recorded to locate the hand position. Orientation information is not used in this task.

3.1 OHSV_Prox: Openni_tracker with HSV colour space + Kalman filter + Proximity

This approach is based on the proximity of the subject to the robot. This algorithm interprets the hand as the human part closer to the robot (**Figure 9**, 9.b) and compares the information with the results coming from the *openni_tracker package*.

A combination of skeleton tracking and segmentation has been implemented (see Figure 9), taking the advantages of both systems to improve the system response and warranty the human detection (focused on the hand detection).


Skin Segmentation
9.a


Proximity selection
9.b

Frontal view real photo
9.c

Figure 9: General information about skin segmentation and proximity selection by colour using Kinect.

In order to improve the execution time of the image processing and the robustness of the hand localization, an estimation of the hand position is calculated using a Kalman filter. In our application the Kalman filter is used to ease the hand tracking in the images provided by Kinect camera. It estimates the position of the hand in the later frame, enabling that way to reduce the area within the image in which the hand will be searched.

The reduction of the image area is proportional to the distance of the image related to the camera and to the estimated error. Thanks to the reduction of the size of image that has to be analysed we have been able to reduce the processing time of the algorithm of 7%.

The Kalman filter estimates the state of a process having as inputs the sequence of noisy observations, in our case, it estimates the possible position of the hand in the next frame, reducing the area processed by the colour segmentation (Figure 10). The Kalman filter model is defined as:

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k$$

Where, F_k is the state transition which is applied to the previous state x_{k-1}, B_k is the control input model and w_k is the process noise. At the time k a measurement z_k of the state x_k is performed as, $\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$ where H_k is the measurement model which represents the true state in the observed space and v_k is the observed noise.

Using the `openni_tracker` package and the skin segmentation algorithm we can verify the hand position in each frame. A Kalman filter is used to track the hands position along the later images. This information gives to the system additional information about the possible situation in the image of the human's hand for every frame.

Basically, the algorithm implements three nodes. Two nodes are running on parallel and publishing each a tf. Then a third node chooses between those tfs, when the distance between both tfs is greater than 10 cm, then the segmentation tf is selected. Otherwise, the *openni_tracker* that executes faster publishes its tf.

A scheme of the proposed algorithm functionality is shown in Figure 10 and an illustration of the results is presented on Figure 11.. The location given by the openni_tracker is called */left_hand*, the location found by the skin segmentation is called */object* and the final location produced by the hand fitting node is called */hand_followed*.

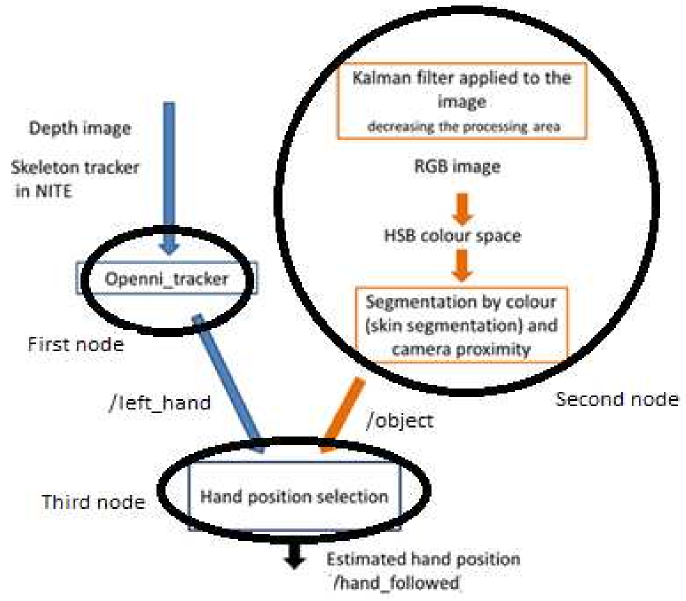

Depth image
Skeleton tracker in NITE
Openni_tracker
First node
/left_hand
Kalman filter applied to the image
decreasing the processing area
RGB image
HSB colour space
Segmentation by colour (skin segmentation) and camera proximity
Second node
/object
Third node
Hand position selection
Estimated hand position /hand_followed

Figure 10. Proposed solution A: combination of the openni_tracker and colour segmentation functions with a third node that selects the final hand location.

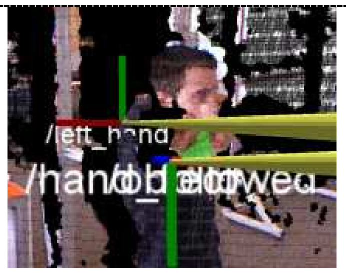
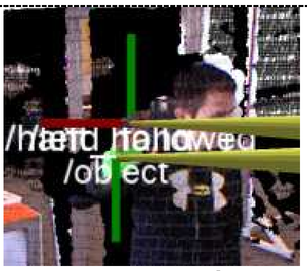
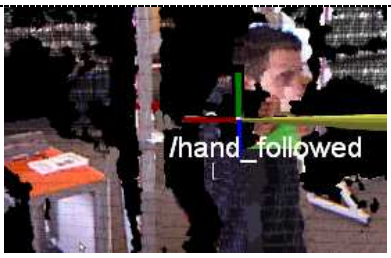
 *Skin detection is the selected /hand_followed.* ***10.a***	 *Openni_tracker→/left hand is the selected hand followed* ***10.b***	 */hand_followed generated by the hand position selection.* ***10.c***

Figure 11. Examples of the outcomes of the algorithm implemented for the human tracking detection.

Figure 11 shows the three tf generated by the OHSV_Prox algorithm. In 10.a, the estimated hand position is */object* (that the frames */object* and */hand_followed* are overlapping), in 10.b, the */hand_followed* frame is the one produced by the *openni_tracker (/left_hand),* and, finally in 10.c, we see that */hand followed* is the only output of the algorithm assuring the correct location of the human hand.

3.2 OYCbCr_Face: Openni_tracker with YCbCr colour space + face filtering

After getting the segmented points by the skin colour filters using RGB+HSV+YCbCr, an Euclidean clustering technique is applied to get groups/clusters of near points which are large enough to be consider as parts of the human body. This clustering consists in segmenting the point cloud in different parts, each part is composed by points with distance between them smaller than a predefined threshold (parameter known as *cluster tolerance*). There is also another threshold able to ignore clusters containing too many or too less points, helping to remove noisy or isolated points and clusters from the scene and to reduce the computational effort.

In order to perform an efficient search of the neighbour points during the cluster identification, a 3D space grid division structure called *kd-tree* has been used (Figure 12):

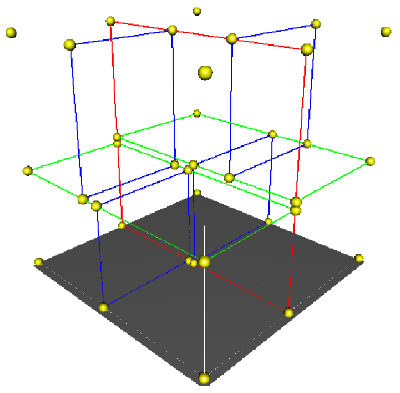


Figure 12. The kd-tree structure exploited for the efficient search procedure of the neighbour points.

Regarding the face detection, taking into account the performance of the perception system, and in order to limit the extra process required to find the face of the subject, the output of the *openni_tracker* has been exploited to remove the face cluster from the scene (considering that that technique also provides an estimation of the face location).
The output of *openni_tracker* gives a 3D location of the head of the subject (published as */head* tf), so all points within a fixed radius, starting from the localized position, are removed with a filter implemented in the *cog_tre_filters package*. To find these near points in an efficient way an additional kd-tree structure (similar to the one previously introduced) has been used.

As the human face has been detected and localized, another solution is to filter the scene image using the depth position of the face. This procedure allows considering just the points nearer to the camera than the face during the searching of the hand that will take part into the object exchange.

At this point we need to fuse data from different sources, mainly from the *openni_tracker* and from the hand detection processes. To achieve this and in order to create a more robust hand detection system, it has been implemented a temporal filter based on a sliding-window approach. The filter stores the last n pose estimations for the hand within a queue, and gives the average position depending on the stored information. In addition, the implementation of the temporal filter makes more robust the estimation by discarding hand positions which distances to the average position are higher than a defined threshold. The average value computed by the filter is the arithmetic mean of the stored locations. Since the hand location can change rapidly based on the speed of the movements, the number of locations stored in the queue (ie n) must be small as much as possible, avoiding that average values will be affected by locations far from the actual position.

This filter is also used to merge data from the hand detectors and the *openni_tracker* function.
By catching the **tf** data from the *openni_tracker* function when a new *msg* from the hand detector comes up and then by putting this data tighter within the temporal filter, it is possible to handle the instability issue observed with *openni* as previously described and to improve the robustness of the whole process (see Figure 13).

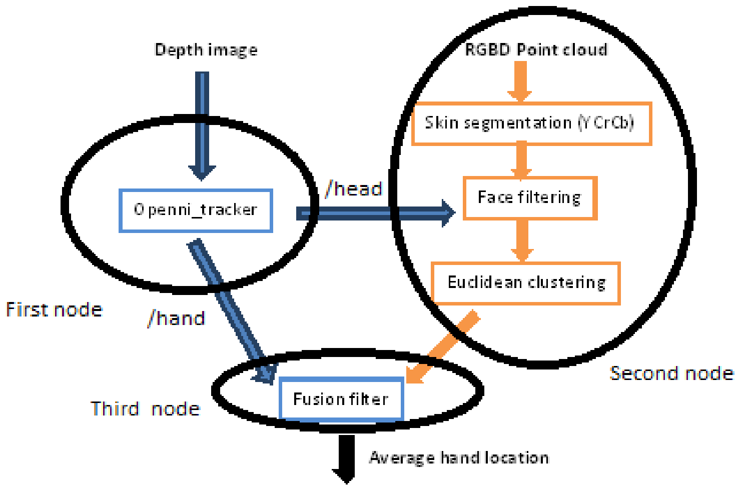
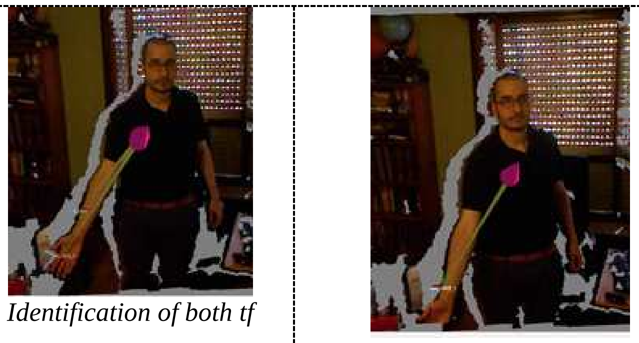


Figure 13. Proposed solution B: Point cloud parallel processing for hand position.



Identification of both tf

/Average_hand generated by the data fusion of both published tfs.

Figure 14: Example of the algorithm implemented for the human tracking detection

3.3 Actual processing times and examples of functionality

Proposed algorithm	Part of the algorithm	Execution time (mean time)
OYCbCr_Face: YCbCr colour space + face filtering	Openni_tracker	**30 Hz**
	Skin filtering in YCbCr	**29.42 Hz**
	Head detection	**3.6 Hz**

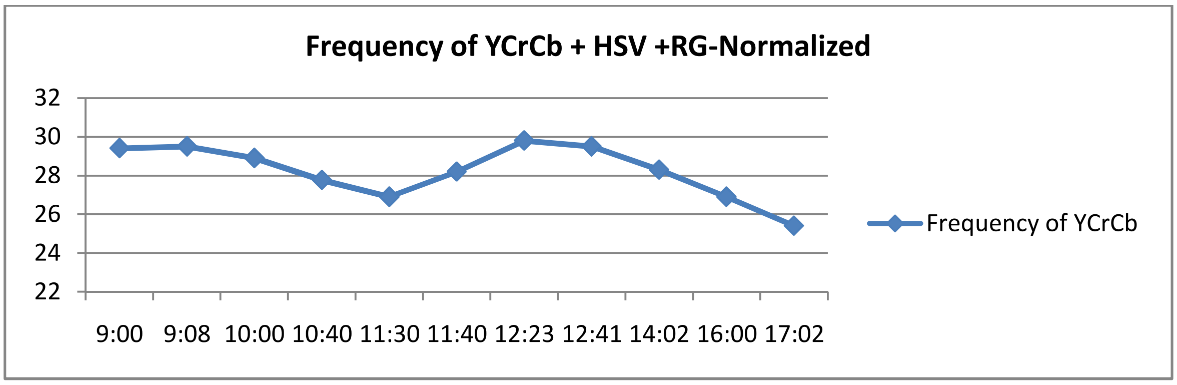
	Fusion filter	**6.8 Hz**	
Total time		Normal: 30**Hz**	Worse case: 3.6 **Hz**
OHSV_Prox: HSV colour space + proximity + Kalman filter	Openni_tracker	**30 Hz**	
	Without Kalman filter RGB+HSV colour space	**21Hz**	
	With Kalman filter RGB+HSV colour space	**35Hz**	
	Hand fitting	**100 Hz**	
Total time		Normal: **30 Hz**	Worse case:**21 Hz**

Table 3: Mean frequencies in the execution of the proposed algorithms for human motion capture focused on the human hand. Inside OYCbCr_Face and OHSV_Prox, two parallel lines are executed and the frequency of each of them has been also measured.

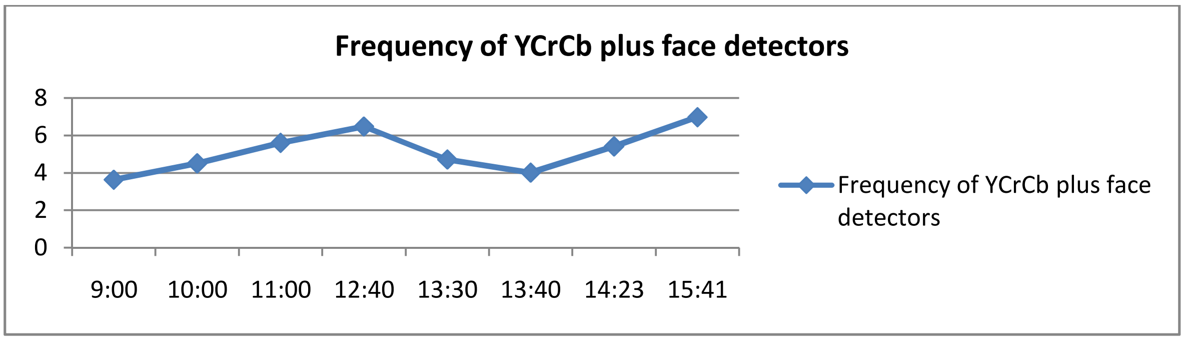
3.3.1. OYCbCr_Face Results:

OYCbCr_Face Results without face detection

The conditions of trials were: Camera configuration as in the Coglaboration scenarios (1.74 cm height, down slope), skin colour segmentation in YCbCr colour space, using *openni_tracker* and depth segmentation.



OYCbCr_Face Results with face detection



An average value between the different tfs is published by the fusion node (third node). The tracking of the hand is shown in the next figure (Figure 15) and to give more detail the selection, a comparison of each of the axis on the Cartesian space is included. (Figure 16)

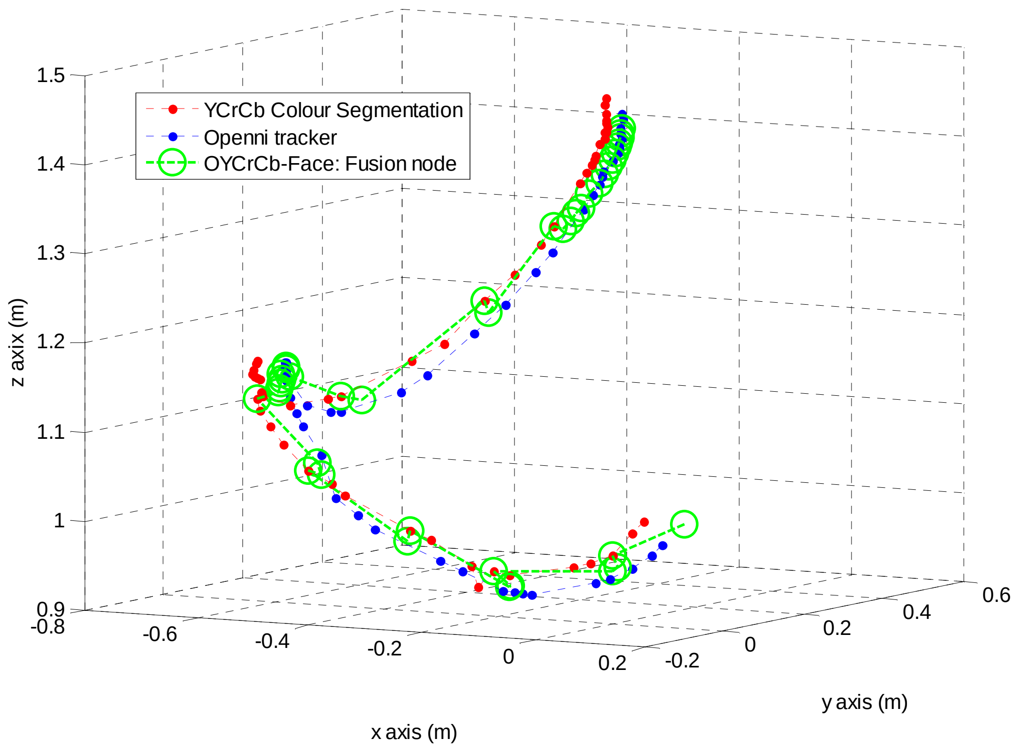
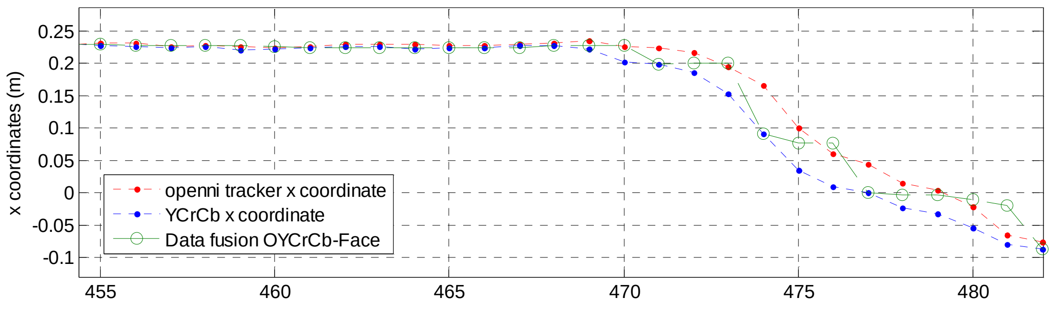


Figure 15: OYCbCr_Face results for hand tracking. The output of the algorithm is the green one.



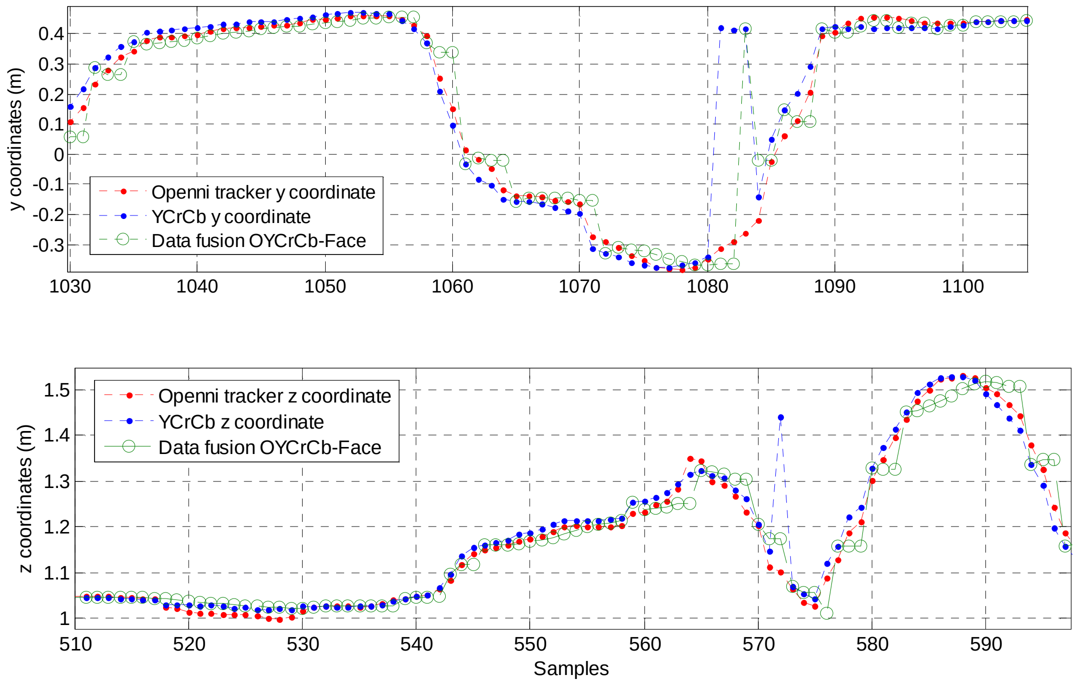


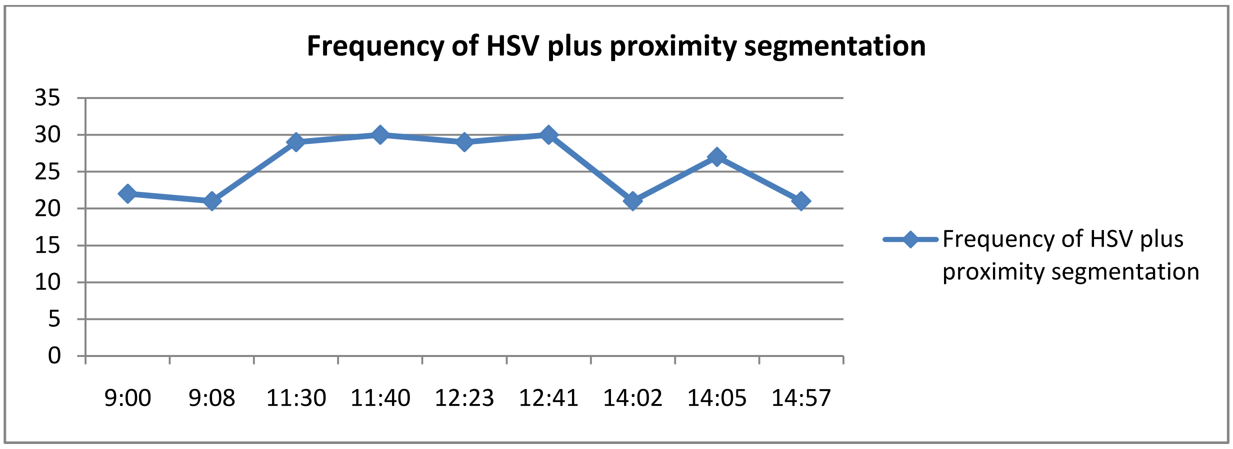
Figure 16: Cartesian space comparison of the human motion tracking in each of the x-y-z axes.

3.3.2. OHSV_Prox Results:

The conditions of trials were: Camera configuration as in the Coglaboration scenarios (1.74 cm height, down slope), skin colour segmentation in HSV colour space, using openni_tracker and depth segmentation and proximity selection.

In the tests analysed from 250 cycles performed by the vision system, the third node published 99 times the *openni_tracker* tf (*/left_hand→/hand_followed*) and 151 times the colour space segmentation + Kalman filter tf (*/object→ /hand_followed*).

This algorithm executes faster than the one implemented with the face recognition approach and regarding the reliability, it has been shown a robust implementation.



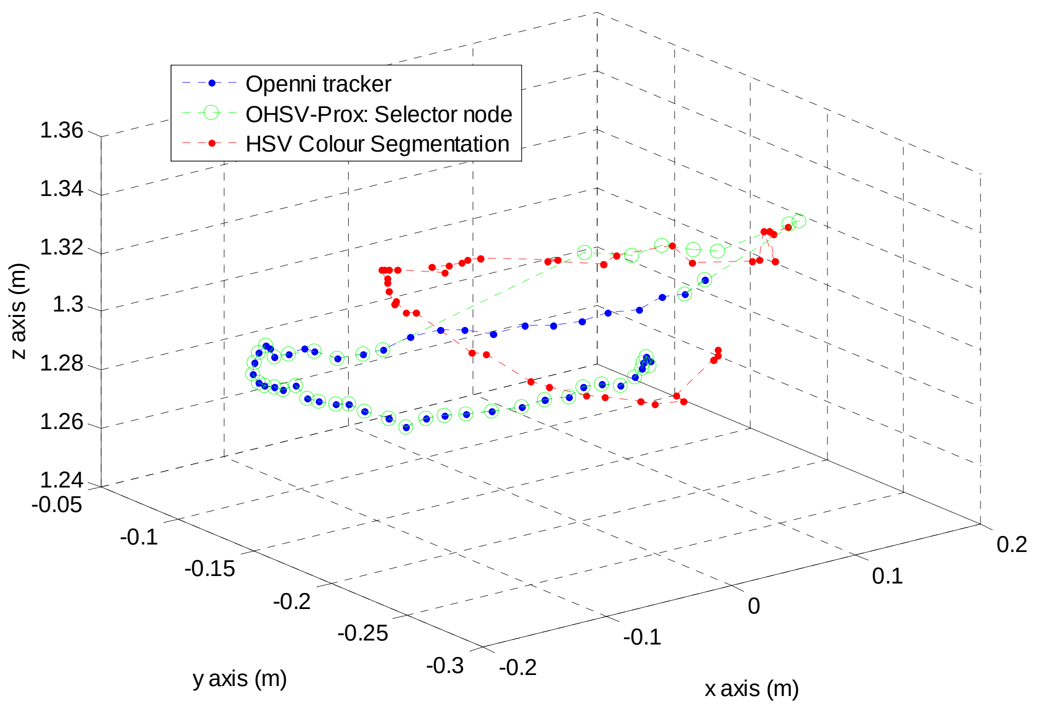
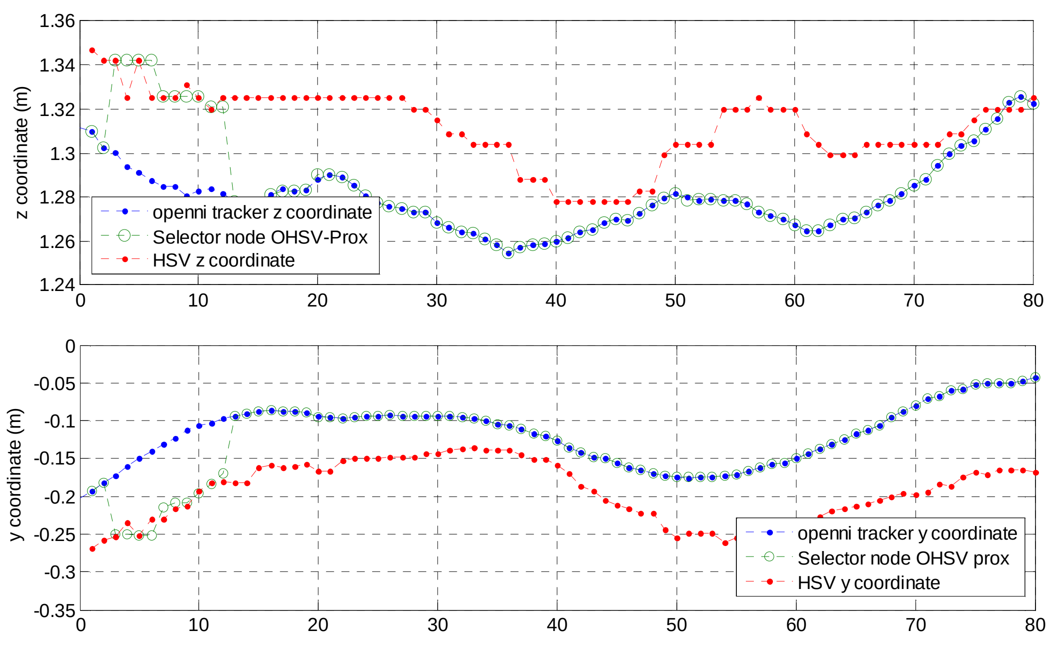


Figure 17: OHSV_Prox results for hand tracking. The output of the algorithm is the green one



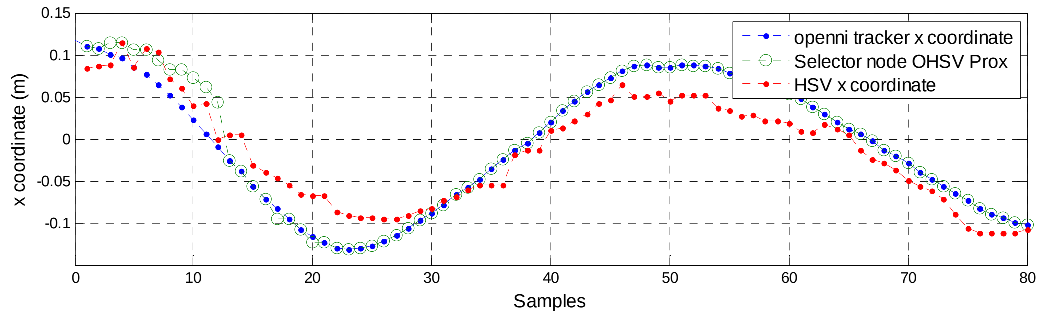


Figure 18: Cartesian space comparison of the human motion tracking in each of the x-y-z axes.

3.3.3. Reliability

The reliability of the algorithms for the Human tracking was tested and analysed. The parallel node added to the *openni tracker* ensures the correct hand location of the person. This combination increases the reliability of the location but also increases the execution time of the overall algorithm, due to the additional computational load. It was decided to have a confident hand location with an increasing processing time from the *openni_tracker* (33 ms) to 44-330 ms (depending on the OHSV_Prox or OYCbCr_Face worst case scenario respectively).

The *openni_tracker* can cause false positives if the human body is occluded in the field of vision. The implemented algorithms showed how the proposed solution by detecting colour solves these problems (partial occlusion and incorrect skeleton detection). We estimate the location of the hand through colour (hand_skin) and we compare it with the position estimated by the *openni_tracker*:

- If it is a similar frame position, the location of the hand is assured and the selected tf is the one generated by the *openni_tracker* that executes faster (30 Hz).
- If the hand_skin has a different location, then the hand_skin is the predominant value. In this case the information refreshes below that 30 Hz. It depends on the used alternative algorithm: The colour segmentation HSV plus proximity (30 to 21 Hz) or the colour segmentation YCbCr plus face recognition (3.6 to 7.8 Hz) (Table 1). An average value between the different tfs is not recommended because it could be include an error position.
- If is not an input from hand_skin colour, then the system publishes an error and waits.

Regarding the colour segmentation, it is known that differences in the solar radiation changes the colour perception of the Kinect [17], this problem is presented in the colour spaces based on Luminance and chrominance (as the HSV and YCbCr). For this reason initializing the system in a correct range of colour for the human skin influence deeply the location of the human hand.

A special mention is dedicated to face recognition. This technique requires more processing time and tends to produce some problems in the current implementation. For that reason, the face recognition will not be used as the first option in Coglaboration demonstration scenario, but it will be tested if it is necessary. We want to remark that the face recognition is applied only to remove it from the skin detection, so, if the face is not removed the response of the algorithm will generate a (potentially very dangerous) false positive. Some examples of the reliability issues that the face recognition presents in the current trials are shown below:

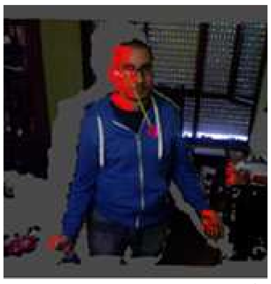
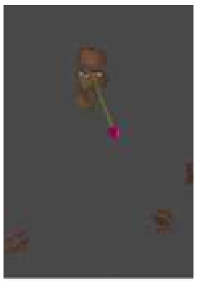
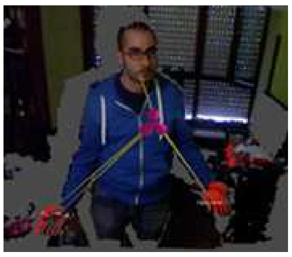
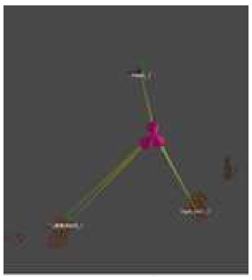
 *Skin detected over pointcloud, no face removed*	 *Skin segmented, no face removed*
 *Skin detected over pointcloud, with face removed*	 *Skin segmented, no face removed*

Figure 19: Example of problems detected with the face recognition implementation.

However, if is necessary to implement face recognition later in the project in order to cope special requirements for the face location (we are keeping in mind the second scenario with seniors in a domestic scenario), we could affirm that:

- For face detection the illumination of the face is very important. Best performance is achieved with the light in front of the face, worst with the light behind the face (if the illumination is poor and with shadows, there is not detection at all).

- ROS nodes for face recognition are too slow to be used in a real time system, alternative safety considerations must be considered.

4. Conclusions

- After running several trials of the regular human body tracking technique in the Coglaboration industrial scenario, some issues have been detected due the partial occlusion of the person. In order to solve them and improve human motion tracking functionality, two different algorithms have been proposed.

- As the monitoring of human movement was focused on the detection of the hand, different features provided by the ROS environment have been explored: the *openni_tracker*, *rgbd_assembler*, *face_detector* and *pi_face_tracker*. Then, the reliable and faster functions have been selected for the different approaches (OHSV_Prox and OYCbCr_Face).

- Preliminary results suggest that for the light conditions in laboratory tests, the YCbCr colour space has the best response for skin detection.

- In the worst case scenario, the algorithm OYCbCr_Face updates the position of the hand at 3.6 Hz (every 0.27 s), while OHSV_Prox processes faster (about 21 Hz, or 0.04 s) (Table 3). These results suggest integrating in the demonstration scenario the OHSV_Prox algorithm for Human Motion Tracking. However, it is not fast enough to apply directly to the robot controller and an interpolation is required.

References

[1] http://www.coglaboration.eu.

[2] http://ros.org/wiki/fuerte

[3] http://www.xbox.com/es-ES/Kinect

[4] Chuck Pheatt and Jeremiah McMullen. ***Programming for the Xbox Kinect™ sensor: tutorial presentation***. Journal of Computing Sciences in Colleges. Pag: 1937-4771. V 27. 2012

[5] http://www.openni.org/

[6] Bor-Jeng Chen; Cheng-Ming Huang; Ting-En Tseng; Li-Chen Fu. ***Robust head and hands tracking with occlusion handling for human machine interaction.*** *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on* , vol., no., pp.2141,2146, 7-12 Oct. 2012 doi: 10.1109/IROS.2012.6386113

[7] *Abhishek Kar*. ***Skeletal Tracking using Microsoft Kinect. Technical Report***. Berkeley University. 2011

[8] http://www.ros.org/wiki/rviz

[9] *G. Hackenberg, R. McCall, W. Broll, G. Wolfgangenberg and R. McCall*. ***Lightweight palm and finger tracking for real-time 3D gesture control***. IEEE Virtual Reality Conference (VR), 2011.

[10] *Fan Hai Xiang and Shahrel Azmin Suandi*. ***Fusion of Multi Color Space for Human Skin Region Segmentation***. International Journal of Information and Electronics Engineering, Vol. 3, No. 2, March 2013

[11] V. Vezhnevets, V. Sazonov, A. Andreeva. **A Survey on Pixel-Based Skin Color Detection Techniques.** Faculty of Computational Mathematics and Cybernetics, Moscow State University

[12] G. Kukharev, A. Novosielski. **Visitor identification elaborating real time face recognition system**, In Proc. 12th Winter School on Computer

[13] *A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss and W. Burgard*. ***OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees***. Autonomous Robots. 2013. DOI 10.1007/s10514-012-9321-0

[14] L. Busin, N. Vandenbroucke and L. Macaire. ***Color spaces and image segmentation***. Université des Sciences et Technologies de Lille. France. 2007.

[15] http://www.ros.org/wiki/robot_state_publisher and http://ros.org/wiki/tf/Overview/Transformations

[16] Man-Woo Park, A. Makhmalbaf and I. Brilakis, ***Comparative study of vision tracking methods for tracking of construction site resources.*** Automation in Construction, Volume 20, Issue 7, November 2011, Pages 905-915, ISSN 0926-5805, http://dx.doi.org/10.1016/j.autcon.2011.03.007.

[17] P. Debeec, T. Hawkins, C. Tchou, Chris, H-P Haarm-Pieter, W. Sarokin and M.Westley. ***Acquiring the reflectance field of a human face***. Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH 2000). ISBN: 1-58113-208-5. DOI: 10.1145/344779.344855.

[18] http://www.cognotics.com/

[19] Yu-Jin Zhang. ***Advances in Image and Video Segmentation***. IRM Press, 2006.

[20] D. Chakraborty, G. Kumar Sen and S. Hazra. ***Image Segmentation Techniques: Methods for Segmenting High Spatial Resolution Images***. Lambert. 2012. ISBN: 978-3-8473-1137-9.

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. R. Berger and A. Ng. ***ROS: an opensource Robot Operating System***. *ICRA workshop on Open Source Software*, 2009.

[22] Yasser F. O. Mohammad, T. Nishida and S. Okada: ***Unsupervised simultaneous learning of gestures, actions and their associations for Human-Robot Interaction***. IROS 2009: 2537-2544