Fabio Caniglia

Robot Gesture Recognition

Department of Electronic Systems
Electronics & IT
Fredrik Bajers Vej 7 B
9220 Aalborg
Phone 9940 8600
http://es.aau.dk

Title:
Robot Gesture Recognition

Project Period:
Feb 11th 2011 –
June 7th 2011

Group member:
Fabio Caniglia

Supervisor:
Kamal Nasrollahi

Censor:
Thomas B. Moeslund

Number of copies:
3

Report pages:

Contents of DVD:
Report.pdf
Source code
Videos of the Tests

Fabio Caniglia

Abstract
In the last years some applications have been developed to achieve new ways of interaction between human and robot, the aim of this interaction is to make simpler and more intuitive the way in which the user can interface himself with a robot without using a keyboard, a mouse or anything else.
The aim of this project is to define a system which is able to recognize gestures used to drive a robot. For this system some simple gestures are defined such as stop, come here, turn left, turn right and so on, this kind of gestures are very common to drive a robot in any condition. This report will be divided in x section, the first section is an introduction about the problem of the gesture recognition with the definition of poses and gesture used in this system,
The second is a review about related works, the third deal with problem delimitation of this project, the forth describe the system in its components (hardware and software) and how the system itself works, the fifth section describe the experimental result of this project and the sixth section deal with the conclusions of this project and future works.

Contents

1 Introduction

1.1 HCI nowadays

The interaction between humans and machine/computer (HMI/HCI) has developed from the use of hand-controller device to a more intuitive interaction. The most common user interfaces for HMI/HCI are keyboard and mouse, the feedback is given via message from the field video and other sensors. In the last few years some new systems are being developed to find a more natural interface to interact with computer and machine, such as speech, handwriting, gesture recognition.
There is a large number of researches working on gesture recognition, and a few of them are related to robot control. This kind of systems should be developed to be efficient and a real time gesture recognition system to perform more human-like interface between human and robot. Two approaches are commonly used to interpret gestures for human machine interaction. One is gloved-based approach that requires wearing of cumbersome contact devices and generally carrying cables that connect the device to a computer or via wireless connection with the computer. Another approach is vision-based technique that does not require wearing any contact devices on human body parts, but uses a set of video cameras and computer vision techniques to interpret gestures. For gesture interpretation system, gestures' modeling is the first step that mainly depends on the intended application of those gestures. An appearance-based algorithm is a strong tool for object recognition. A variety of object appearances is stored as a statistical model and used in the recognition task. The gestures are modeled by relating the appearance of any gesture to the appearance of the set of predefined template gestures.
The visual gesture interface is a reasonable choice, since it is easy to deploy and can be used anywhere in the field of view of a visual camera and is being realized by identifying human postures and movement. The operator does not need to master special hardware. Vision also allows a variety of gestures to be used, since it is software based.

1.2 Gesture

Gesture literally means "an expressive movement of a part of the body, the hand or head in order to bring forward intentions and attitude". For the present application we have considered only arm movements. A gesture is characterized by the movements of the arm considering it as a composition of three elements:

- Shoulder

- Elbow
- Hand.

These three elements move in 3D space, so they have three coordinates (X, Y, Z) which can change frame by frame. Of course a particular robot motion is associated with a specific gesture.

The set of gestures can be divided broadly into two classes:

1. Gestures for which the position of the arm's parts remain unchanged over a motion sequence
2. Gestures for which the position of the arm's parts changes from frame to frame.

In this project only the second class of gesture will be considered, this means that in 3D space the position of the arm's parts are extracted as a feature from representative frames to exploit the gesture recognition.

Anyway a good gesture design should possess the following features:

1. It should be natural, easy to demonstrate, and consistent.
2. The captured images should possess information mainly related to the gestures so that the gesture changes can be clearly identified. And some alterations due to illumination and background noises will not affect the identification results.
3. The background should not confuse the gestures.
4. The processing time should be short. Speed is always desirable for real time HMI/HCI, so the gestures should be applicable for fast processing algorithm.

1.3 Issues in Gesture Recognition

In order to drive the robots operating in uncontrolled real world environments there are several constraints on human-robot-interaction as a special case of human-computer-interaction:

1. The camera visual must cope with variable and possibly complex backgrounds. A system requiring uniform background is not flexible enough for real world applications.
2. The system must be person independent. Many users should be able to operate it, without the necessity for retraining.
3. The system must not require the user to wear markers or colored gloves, this being too tedious.
4. The lighting conditions are uncontrolled. The camera must cope with various lighting situations.
5. The system must be capable of real time performance.

Automatic visual gesture recognition holds the promise of making man-machine interfaces more natural.

By using the OpenNi framework we can solve all these issues, now we will explain what we have done for each of these issues:

1. By using the depth image and the classes provided by the OpenNI framework we can use this system in complex background since it is able to track the user after he stand in a particular position, this allows to the system to recognize what is and what is not a user.
2. This system can be used by any person that stands in front of the camera under the constrains that it must be the first and stand in a particular position; this is allowed by the OpenNi framework
3. This system recognize the user body regardless for his clothes, because the system uses the depth image to recognize a body and not the RGB image so it not important what kind of clothes are worn.
4. As we said above the system use the depth image that is less affected by different light conditions.
5. The system checks the frame sequence frame by frame to recognize a gesture, this is done in order to have a good performance and allows to the system to be more accurate.

1.4 Problem Statement

The purpose of this project is to drive a robot through the arm gestures; as we said above the gesture that will be used during this project can be classified in as a Dynamic arm gesture. This kind of arm gesture means that the user moves his own arm while the camera acquires the image and then the system recognizes the exact gesture; the dynamic arm gesture is complex, so there can be many combinations of dynamic gestures, this leads us to be able to associate a lot of combinations of dynamic gesture to the robot’s actions.

The project will be tested in two different ways:

- Gesture recognition
- Robot driving

The aim of the gesture recognition experiment is to test the recognition rate of each gesture that will be described later in this section. The aim of Robot driving experiment is to test how the system behaves in a real situation.

1.4.1 Coordinate System

Before describing the gesture modeling of this project we need to talk about the Coordinate System used in it. Since we use a (X,Y,Z) coordinate system, we must explain how these coordinates are extracted in the figure below we can see the a snapshot of the UI’s part. This picture shows the min and max value for each value

- X ϵ [0,640],
- Y ϵ [0,480].

As we can see the Origin of the system is up on the left, and the max down on the right.

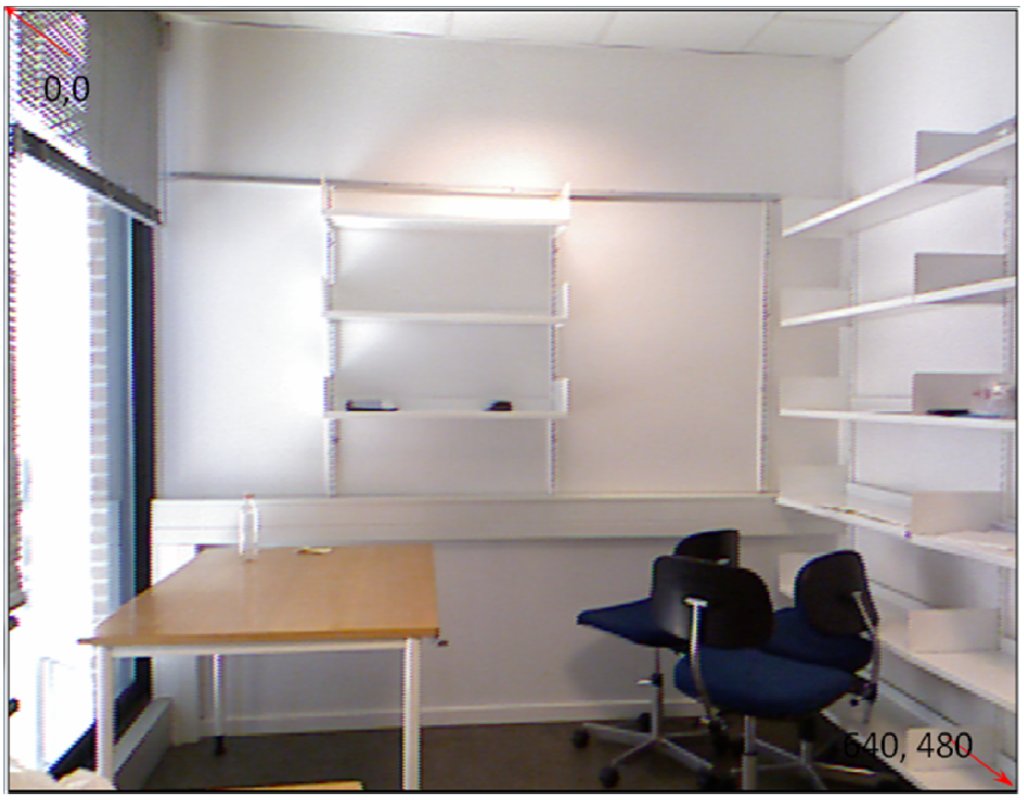


Figure 1: X and Y coordinates and their min and max values

In the next figure we can see how the Z coordinate works, this coordinate is based on the distance between the Kinect and the user; as we told above the distance is calculated by using the two depth sensors. In this case the origin is the sensor itself.

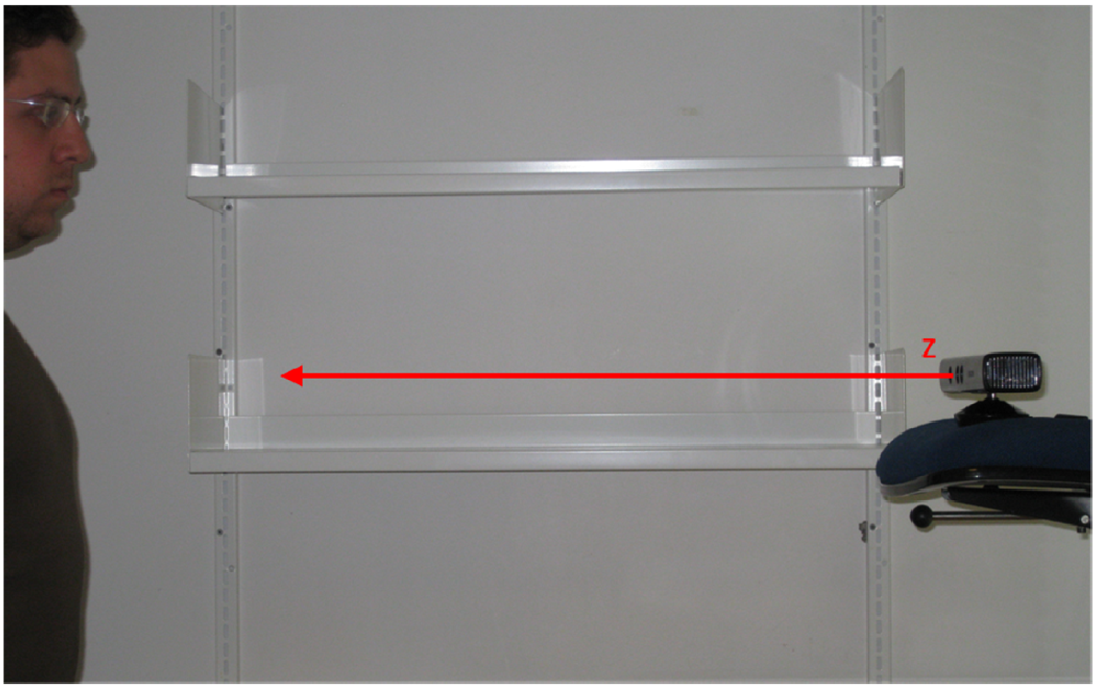


Figure 2: The Z coordinate System

1.4.2 Pose Definition

We can define 5 kind of pose which will be used to define the gestures, we describe each pose with a picture.

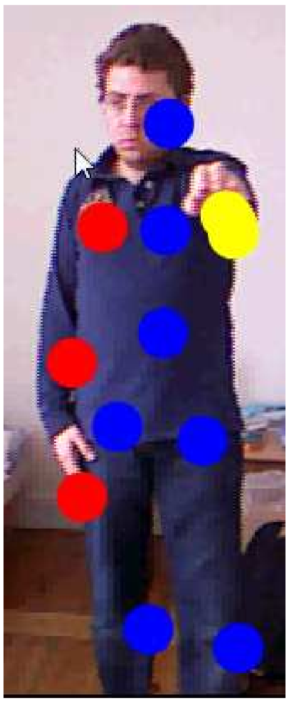


Figure 3: Front pose

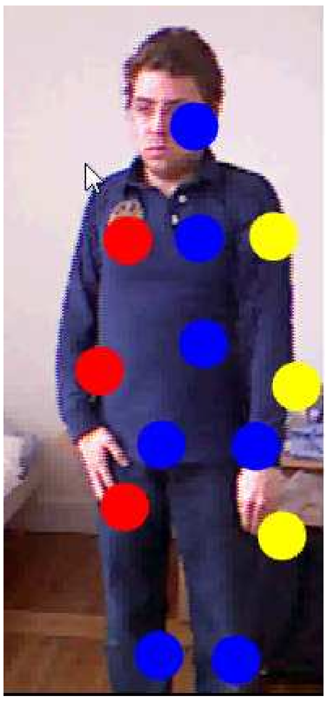


Figure 4: Lower Pose

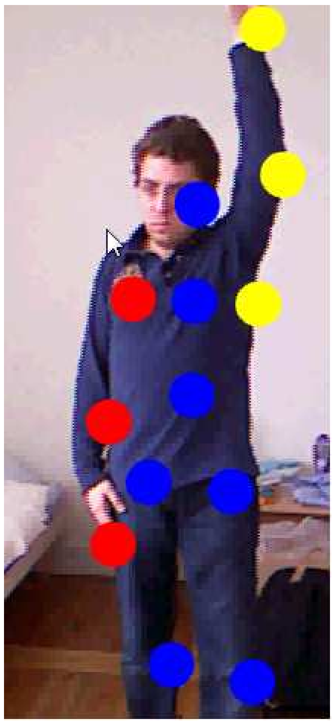


Figure 5: Stop pose

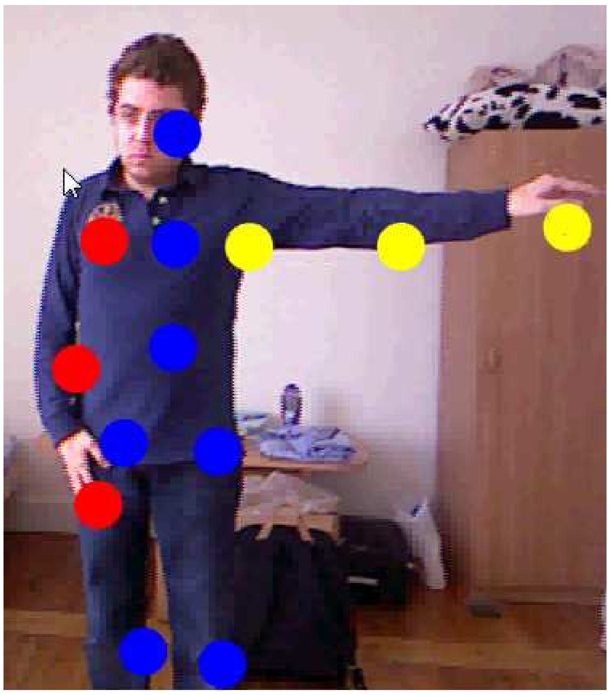


Figure 6: Turn Right pose

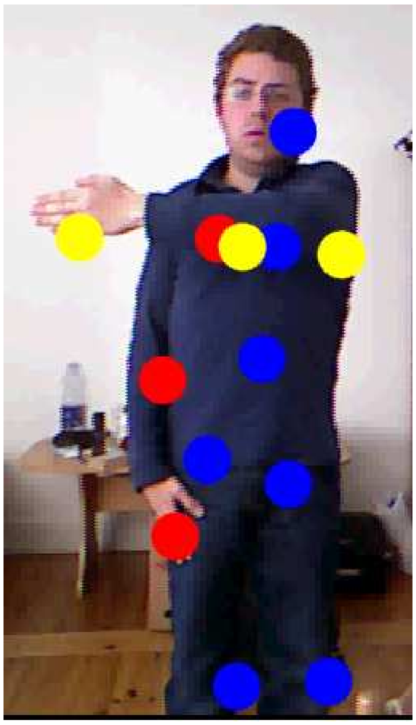


Figure 7: Turn Left pose

1.4.3 Gesture Definition

The gestures considered in this project will be called Base or Movement gestures. Each gesture will be described with:

- some pictures which will show the arm's pose during the gesture;
- the coordinates' variations during this gesture that means how the hand and elbow change their position during the gesture.

The coordinates variation will be used in the gesture recognition step to feed a FSA which will be used to identify in a proper way the gesture.

1.4.3.1 Base or Movement Gestures

There are five gestures in this category:

- Come here (go back),
- Turn left,
- Turn right,
- Go away (go ahead),
- Stop.

In this section the changes refer to the Coordinate System of this project, see Section 1.4.1 .

Come here (go back):

In this gesture the hand and the elbow go down, while the shoulder remains in the same position for all the gesture. In the coordinate system this means:

- $X_{shoulder}$, $Y_{shoulder}$ and $Z_{shoulder}$ do not change;
- X_{hand} and X_{elbow} do not change;
- Y_{elbow} and Y_{hand} increase;
- Z_{hand} and Z_{elbow} increase.

This means that the hand and the elbow start at the same X and Y coordinates but are closer (Z coordinate) than the shoulder to the camera and they end with the same X and Z coordinates but the Y coordinate of elbow and hand are greater than the shoulder one.

First Pose: Front, Second Pose: Lower

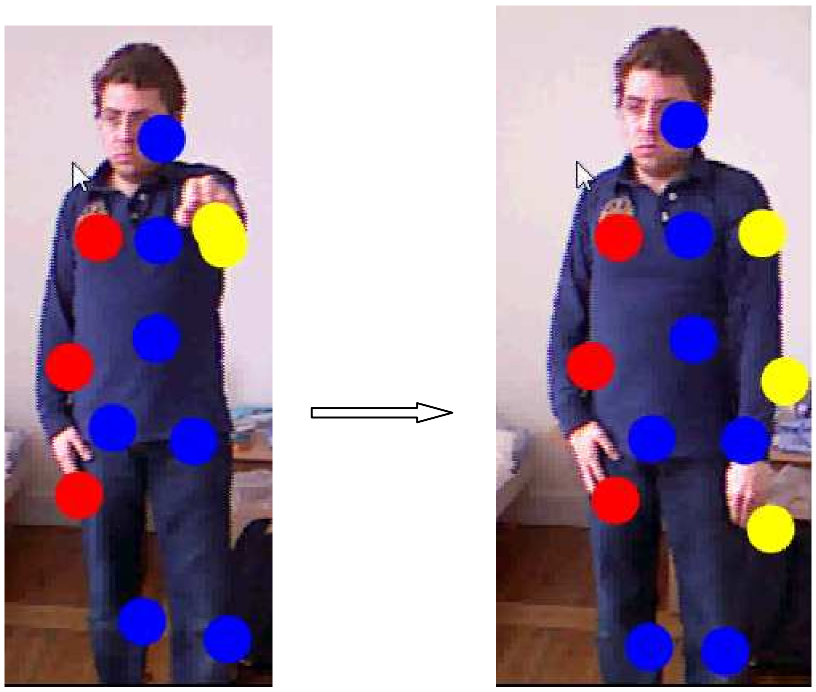


Figure 5: poses of the Come here (Go back) gesture

Turn left:

In this gesture the hand and the elbow go left, while the shoulder remains in the same position for all the gesture. In the coordinate system this means:

- $X_{shoulder}$, $Y_{shoulder}$ and $Z_{shoulder}$ do not change;
- Y_{hand} and Y_{elbow} do not change;
- X_{elbow} and X_{hand} decrease;
- Do not care Z_{hand} and Z_{elbow}.

This means that the hand and the elbow start at the same X and Y coordinates but are closer (Z coordinate) than the shoulder to the camera and they end with the same Y and Z coordinates but the X coordinate of elbow and hand are lesser than the shoulder one.

First Pose: Front, Second Pose: Turn Left

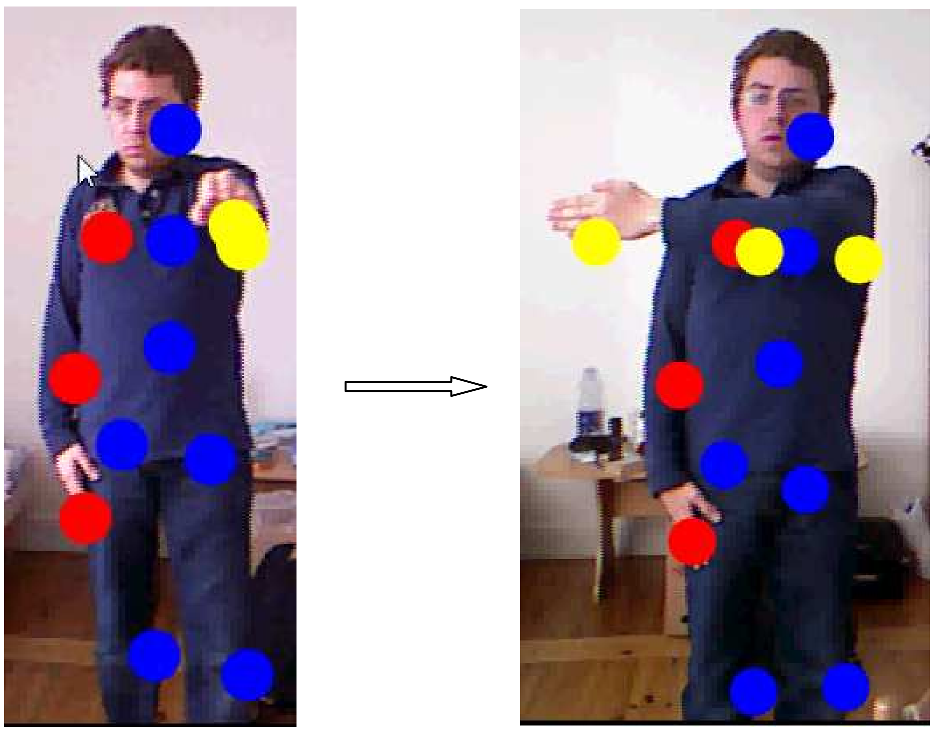


Figure 6: poses of the Turn Left gesture

Turn right:

In this gesture the hand and the elbow go right, while the shoulder remains in the same position for all the gesture. In the coordinate system this means:

- $X_{shoulder}$, $Y_{shoulder}$ and $Z_{shoulder}$ do not change;
- Y_{hand} and Y_{elbow} do not change;
- X_{elbow} and X_{hand} increase;
- Do not care Z_{hand} and Z_{elbow}.

This means that the hand and the elbow start at the same X and Y coordinates but are closer (Z coordinate) than the shoulder to the camera and they end with the same Y and Z coordinates but the X coordinate of the elbow and the hand are greater than the shoulder one.

First Pose: Front, Second Pose: Turn Right

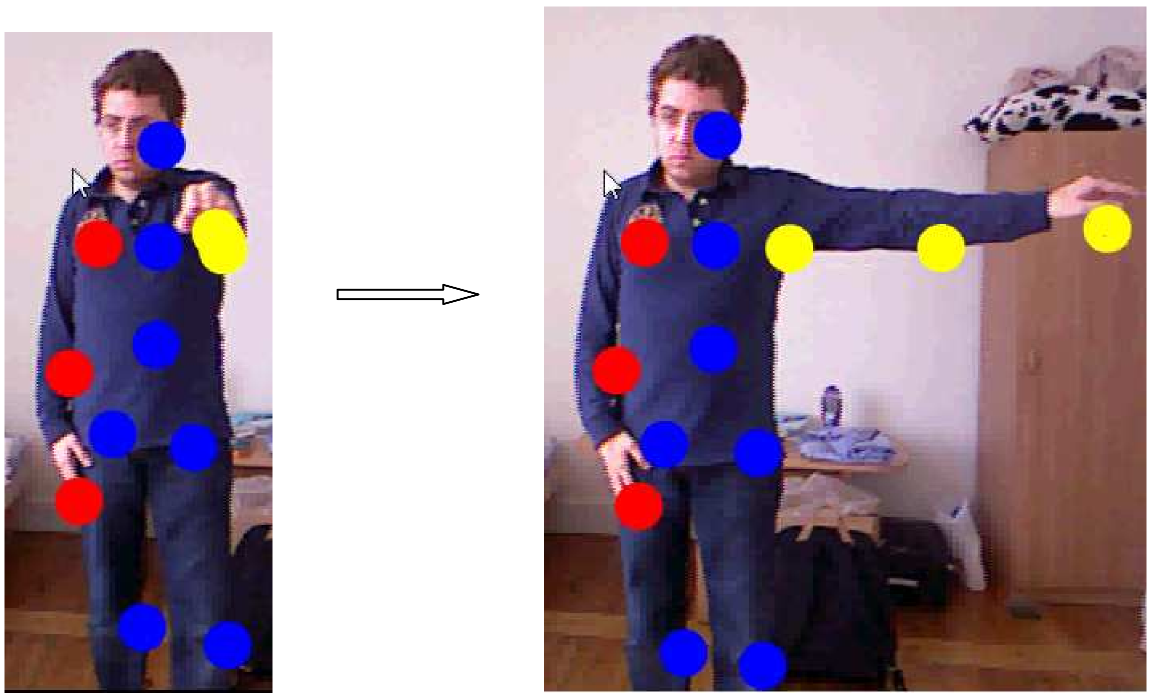


Figure 7: poses of the Turn Right gesture

Go away (go ahead):

In this gesture the hand and the elbow go up but they do not go beyond the height of the shoulder, while the shoulder remains in the same position for all the gesture. In the coordinate system this means:

- $X_{shoulder}$, $Y_{shoulder}$ and $Z_{shoulder}$ do not change;
- X_{hand} and X_{elbow} do not change;
- Y_{elbow} and Y_{hand} decrease;
- Z_{hand} and Z_{elbow} decrease;
- $Y_{shoulder} > Y_{elbow}$ and $Y_{shoulder} > Y_{hand}$.

This means that the hand and the elbow start at the same X and Z coordinates but the Y coordinate of elbow and hand are greater than the shoulder one and they end with the same X and Y coordinates but are closer (Z coordinate) than the shoulder to the camera.

First Pose: Lower, Second Pose: Front

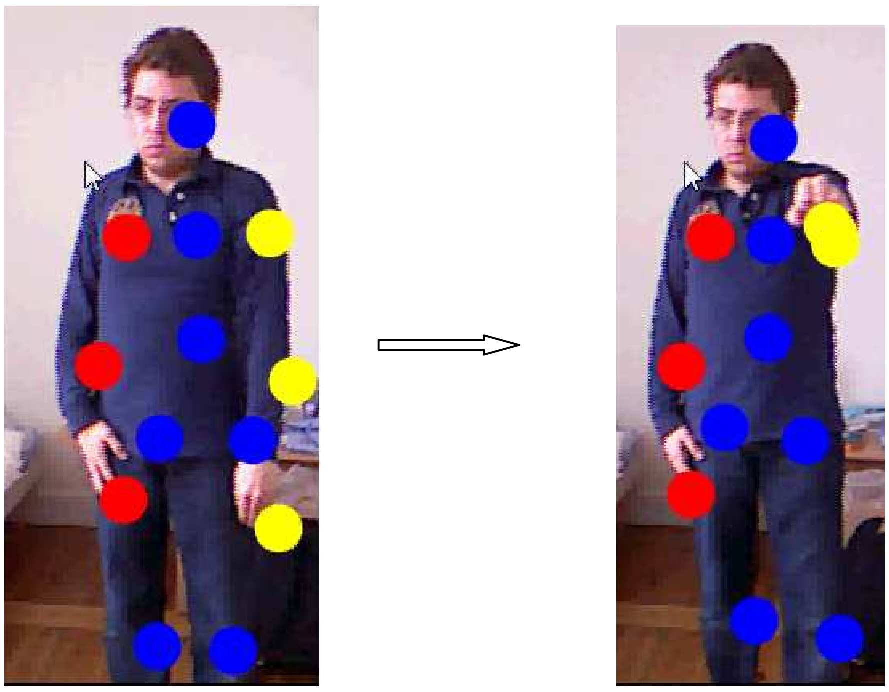


Figure 8: Poses of the Go away (go ahead) gesture

Stop:

In this gesture the hand and the elbow go up and hand goes beyond the height of the shoulder, while the shoulder remains in the same position for all the gesture. In the coordinate system this means:

- $X_{shoulder}$, Yshoulder and Zshoulder do not change;
- X_{hand} and X_{elbow} do not change;
- Y_{elbow} and Y_{hand} decrease;
- Z_{hand} and Z_{elbow} increase;
- $Y_{shoulder} > Y_{hand}$.

This means that the hand and the elbow start at the same X and Y coordinates but are closer (Z coordinate) than the shoulder to the camera and they end with the same X and Z coordinates but the Y coordinate of the elbow and the hand are lesser than the shoulder one.

First Pose: Front, Second Pose: Stop

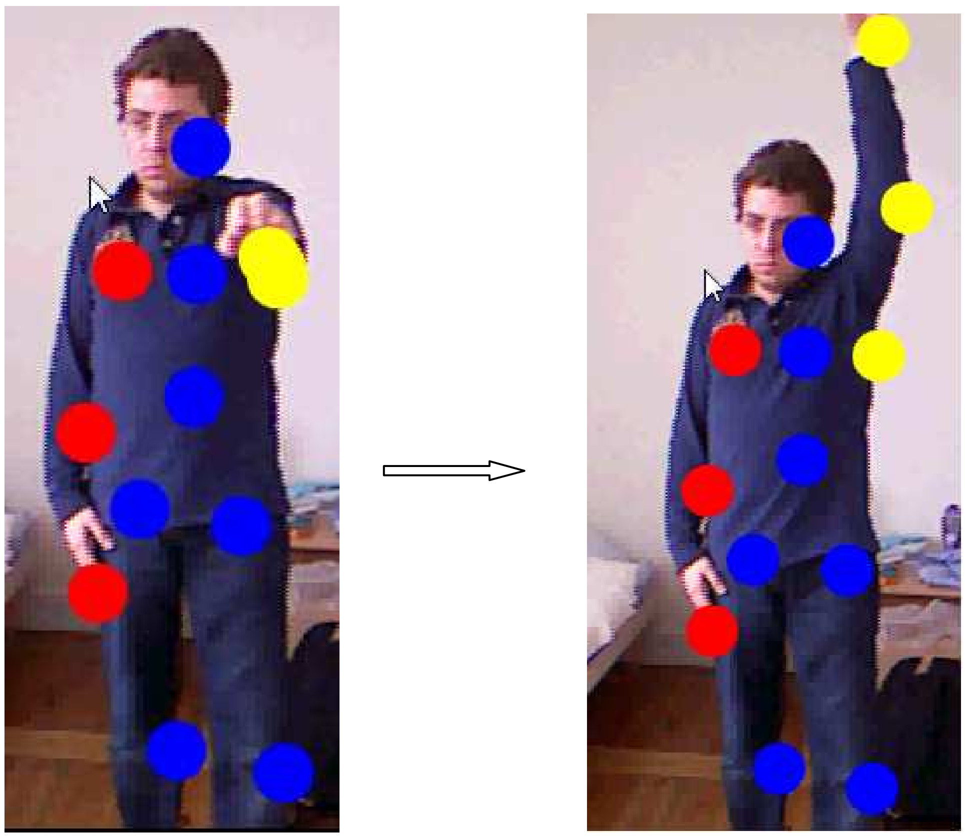


Figure 9: Poses of the Stop gesture

2 Related Works

The typical gesture recognition method is based on 4 steps:

1) Image acquisition,
2) Image processing;
3) Feature extraction;
4) Gesture recognition.

4.1 Image Acquisition

In paper [1] this step is done through a 3d and color camera, in paper [10] this step is based on Bumblebee2 camera which gives 2D image sequences and depth image sequences. The other paper considers this step as a tacit one.

4.2 Image Pre-Processing

This step is done in order to prepare the image for the next step, this means that some parts of the image are selected as interesting areas. In this step there are methods which use a particular kind of color space (RGB, YIQ, YCbCr and so on...) while other methods use the depth image to detect the areas of interest .

For paper [1] the arm segmentation is done through a clustering based on the depth image and the merging of these clusters until a specific number of clusters is reached. The hand-forearm segmentation is based on statistical modelling of the arm points in 3D space, using 3D Gaussian distribution and the hand pose estimation is done estimating the 3D orientation of the hand obtained by computing the principal direction of the 3D data and to compensate for the 3D pose, an orthonormal coordinate frame is defined, with center **m** (the center of mass), and its x, y and z.

The paper [2] uses OpenCV to get the skin colour and it makes the segmentation based on skin and motion information. The papers [3]and [6] use the Viola and Jones's feature set to detect the hand, [3] also uses Adaboost and the motion information.

Papers [5] ,[7] and [9] use the YCbCr colour space to the detect the areas with skin color, [5] binarizes the image and extracts the edges, while [7] only binarizes the image. Papers [4] and [10] use the depth image, and in particular [10] combines this image with a 2D image and then binarizes the image and extracts the edges.

In [8] we have a thresholding method to obtain interesting frames, which are grouped and binarized to obtain only the hand region and then the system erases the part not involved in the gesture.

[11] uses a combination between colour and motion detection to find the hand regions.

[13] and [16] use L*a*b* colour space to feed a RCE neural network to detect skin color region. [14] use the HLS colour space to detect skin colour region and to find the hands, one of the hans will be labelled as index hand that means that the hand show the action to be done.
[15] uses the HSV colour space but using only the hue (H) and saturation (S) components, then it uses the motion and color cue and merges them to obtain the area with the gesture. [17] use a conversion from RGB to rg with the follow formulas:

$$r = R/(R+G+B)$$

$$g = G/(R+G+B)$$

and uses them to detect the skin color, furthermore [17] uses the colour of the t-shirt to track the user.

4.3 Feature Extraction

This step is used to extract the information in order to recognize the gesture. There are a lot of features which can be extracted: shape, distance from the center of the hand, Hu moments.
In papers [1] and [4] the extracted features are the distance from the center of the hand, and they are extracted by sampling along some circles with center **c** and increasing radius. Paper [2] extracts the features by tracing the moving hand and the velocity of the moving hand and the distance between the previous hand and the current hand are calculated to extract a meaning gesture region. The relative coordinate between face and hand location is used as features to recognize a gesture. And also the combined features of statistical and structural features of moving hands are used as feature vector.
In [3] scale-space features detection is applied to find palm-like and finger-like structures.
In the paper [5] the features extracted in the image pre-processing step are Hu invariant moment, hand gesture region (arearatio, aspectratio, baryratio, handarea, handperimeter), Fourier descriptor; all this feature are jointly formed to a vector.
In the paper [6] the method extracts the Hu invariant moment features of the detected hand gesture. [7] extracts as features the trajectory and the shape of the hand. [8] uses as feature the image obtained in the previous step. [9] and [12] extract the features from the sequence of frames and collect them in 3 kind:

- Location;
- Velocity;
- Orientation.

Then this features are organized in a vector ([9]) or in discrete symbols ([12]) by using quantization.

In [10] Feature Extraction, three kinds of features are considered: Features for Hand Gesture, Features for Hand Posture divided into:

- Statistical Feature Vector: Hu-Moments are derived from basic moments and area, mean, variance, covariance and skewness are the properties which are extracted from these moments
- Geometrical Feature Vectors, such as: Circularity and Rectangularity used to exploit the hand shape
- Fingertip Detection

In [13] and [16] the features are the topological features of the hand, such as the number and positions of the extended fingers, which are extracted from the binary image of the segmented hand region. To find the number and positions of the extended fingers, the edge points of the fingers are the most useful features.

[11] extracts as features a feature vector Q_{jk} is extracted; Q_{jk} =(x_{jk}; y_{jk}; u_{jk}; v_{jk}), where the 2D position (x, y) is the pixel location of the region centroid and the 2Dvelocity (u; v) is the optical flow averaged over that region, where j is a frame of a sequence and K is one of the hand regions which have been found in the sequence.

In [14], [15] and [17] there is not feature extraction phase, they use the image obtained in the second step to recognize the gesture.

4.4 Gesture Recognition.

In [1] this step is based on the k-nearest-neighbour rule which classifies a new input image, and extracted feature vector a by assigning it the label most frequently represented among the k nearest training samples. The proximity of input vector to training sample vectors is computed using the normalized cross correlation measure. Paper [2] does not deal whit this step because it only talks about the first three steps. In [3] hand gesture type is determined by palm-finger configuration.

Paper [4] for the Hand gesture recognition it is used the patch-based local texture description based on the gray-level Local Binary Pattern (LBP) which will be partitioned into small patches, then the histogram is counted within each local patch and the final feature vector is formed by concatenating all the small histograms based on local patch, To perform the hand gesture recognition, it is selected the simple histogram intersection as the similarity measurement, and the nearest neighbour principle is used to give the final recognition result.

Paper [5] uses a multi-layer perceptron in order to make hand Gesture recognition that uses the feature vector. Paper [6] uses the features to train a SVMs (*Support Vector Machines*) classifier which can realize recognition.

The paper [7] uses a dynamic recognition by using the features and selecting the key frame to reduce the amount of data and it uses a method of recognition static hand gesture based on Fourier Descriptor- BP neural network, for shape based algorithms.

In order to determine gesture [8] makes a pattern matching using PCA (Principal Component Analysis) algorithm, which is suitable for pattern matching as the human hand is used for gesture expression and components present in the hand.

In order to make this step papers [9], [10], [12] use a HMM (Hidden Markov Model),which is fed by a vector which contains all the extracted features, besides that [10] use a SVM (Support Vector Machine).

In [11] by using the extracted features the system makes a spatiotemporal matching with which the system tries to find the hand region that fits best the model of a gesture, in order to do that a function D is used, that minimizes the matching cost of a warping path W (defined as the alignment between model M^g and a subsequence (Q_1,.. ,Q_n) ending at the frame n).

The system proposed in [13] and [16] recognizes the gesture using and analyzing the number and positions of the extended fingers

In [14] this step is based on the pre-processed image:

- Recognition with Edge detection Morphological and Sobel. By analyzing the curves, recognition rules for gesture actions can be taken such that all the gestures can be correctly recognized.
- Template Matching, it is used a set of templates taken from the rectangle blocks after hand labeling. The recognition is realized by finding the action relative to that of the training template which gives out the minimum matching error.
- Recognition by skeletonizing: After similar processing, the simplified square templates are obtained and the skeletonizing is done by Zhang-suen transform, obtaining a 2 or 3 vectors. By inspecting the angles of the vectors, the gesture actions can be found in the direction of the fingers. So the recognition rules can be found

In the paper [15] The Hand Posture Classification step starts when analyzed and its precise three-dimensional position is computed. For this purpose, regions of interest of 256^2 pixels are selected around the points yielded by the tracking for the two cameras. Greyscale images of these regions are downsampled by a factor of two and serve as the input for the hand posture classification. The posture recognition is based on elastic graph matching. Processing is done on greyscale images and works in a person-independent way and in the presence of varying complex backgrounds. In elastic graph matching, objects are represented as labeled graphs, where the nodes carry local image information and the edges contain information about the geometry. One model graph is created for each posture. A graph representing a particular posture is matched to an image by moving it across the image until it fits best to the regions in the image it comes to lie on. During the matching process certain geometrical transformations of the graph are allowed by scaling and making rotation in plane. For the purpose of posture recognition the graph for each posture is matched in both images sequentially and then the similarities of the same posture are added for both images. The posture with the highest total is chosen as the recognition result. This part of the method takes a lot of time since it has to match every posture in the database with the two images.

In the paper [17] this step is based on:

- Pose Analysis is based on two different methods operating on a color-filtered sub-region of the image which contains the person's right side as determined by the tracking module. This two methods are:
    - Graphical Template Matcher (also called: pose template matcher) compares images to a set of pre-recorded templates of arm poses, called pose templates. More specifically, the color-filtered image is correlated with a set of R pre-recorded templates of arm poses. Each pose template T consists of two regions with a fixed value. The result of correlating the image segment with R pre-recorded templates is a correlation vector, or feature vector, of R components, a feature vector is extracted for every frame. These feature vectors form the basis of the temporal template matching.
    - The Neural Network-Based Method is the other way to recognize poses. The neural network-based approach predicts the angles of the two arm segments relative to the person's right side from the image segment. The output corresponds to the angles of the arm segments, encoded using multi-unit Gaussian representations. If the outputs are close to the targets, the network predicts both angles with high accuracy.

3 Problem delimitation

This section describes which kind of constrains this system has. For example some systems have constrains about the background, the user, and the clothes and so on i.e. some systems need:

- a simple background, such as a monocolor background
- Some special clothes to be worn by the user.

This project can be used with any kind of background:

- simple, like a mono-colored wall;
- Complex, like an office or a lab or any other.

The user does not need to wear particular clothes such as colored gloves or a colored shirt.

But in this project there are these constrains:

- Who wants to use the system must be the first person standing in front of the camera;
- The user must stand for a few seconds in a particular pose, called psi-like calibration pose. This particular pose is required by the system itself in order to track the user; the system continuously searches for an user which stands in this position;
- In this moment only one person can use the system, standing in psi-like calibration pose, being tracked by the system and make gestures in front of the camera, this is because the system must recognize only one user at time, because if there are more than one user in the range of the camera that assume the psi-like calibration pose and are tracked by system, it does not realize which one of them must be recognized as the real user of the system and it is possible a mistake.
- A particular ID (=1) will be assigned to the first user which stands in psi-like calibration pose, this is the only ID that can be used to make the system works. If other persons stand in the psi-like calibration pose, another ID (from 2 to 6) will be assigned but the person will not be tracked as user of this system and will not be able to drive the robot, and his gestures will be ignored;
- In order to use the system it needs to select which body's side will be used to make gesture, the default side is the right one, but it is possible to switch the used side by pressing a button which is present in the User Interface of the system.
- The User must wait for a few seconds (1 or 2) before making another gesture, this is made to avoid some mistakes, since the Come Here and Go Away gestures are reversed.
- The User has 5 seconds to make a gesture (also invalid) instead to stand in one of the 5 poses that the system recognized.

4 System Design

This section deals with the system in each of its aspects. First of all we will describe the hardware components of the system such as the camera, the PC and the robot which were used during the project and the test. After this we will talk about the software components such as the framework, the library and the User Interface developed for this project. The last sub-section will describe step by step how the system works.

4.1 Hardware components

The hardware components, which have been used in this project, are the following:

- Kinect Camera
- A laptop pc

4.1.1 Kinect camera

Kinect is a Microsoft device developed for XBOX360 console, this camera has:

- Two 3D depth sensors, which calculate the distance from the sensor;
- Rgb Camera;
- Motorized Tilt;
- Multi-array microphones.

The Kinect’s parts involved in this project are: the RGB camera and the two 3D depth sensors. These parts are used to:

- acquire the user’s image;
- track him;
- acquire the useful data such as the X,Y,Z coordinates of the user.

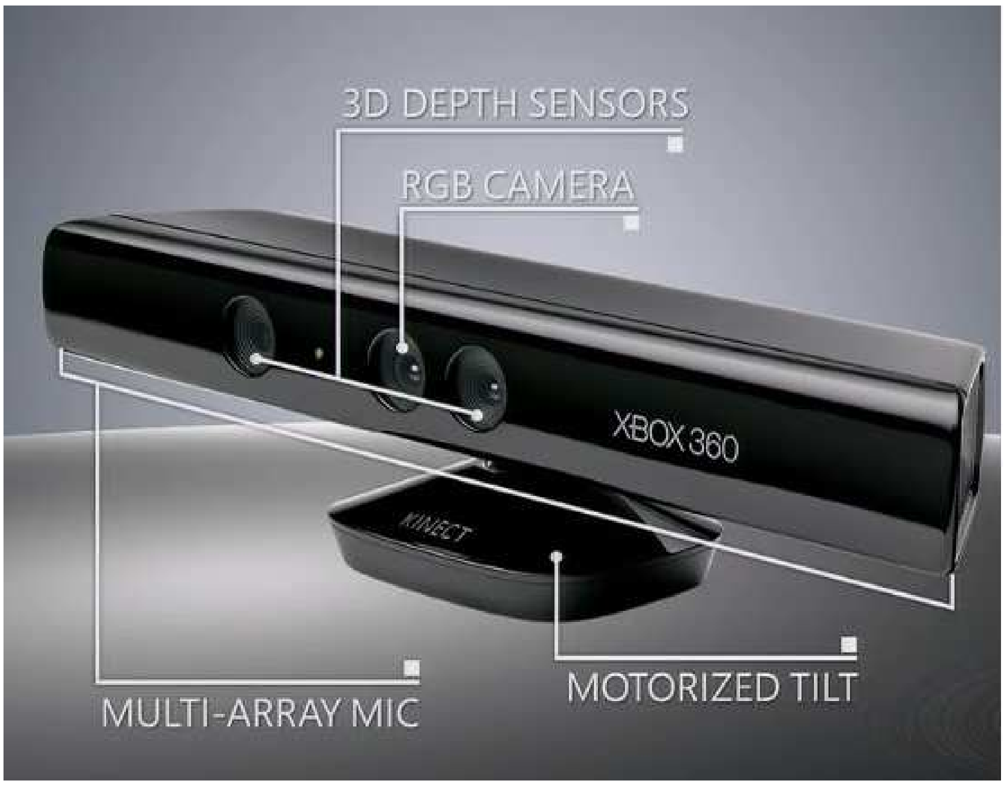


Figure 10: The Kinect camera and its components

4.1.2 Laptop Pc

This Laptop Pc has:

- A Intel Core Duo T6400 @ 2GHz
- 3 Gb ram
- ATI HD 3650 with 512MB
- Windows 7
- Visual Studio 2010

4.2 Software components

This section deals with the Software components which are used in this project:

- OpenNI
- Nui.vision library
- User Interface
- FSA

4.2.1 OpenNI

OpenNI or *Open Natural Interaction* is an industry-led, not-for-profit organisation focused on certifying and improving interoperability of natural interaction devices, applications that use those devices and middleware that facilitates access and use of such devices.

The OpenNI framework provides a set of open source APIs. These APIs are intended to become a standard for applications to access natural interaction devices. The API framework itself is also sometimes referred to by the name *OpenNI*.

The APIs provide support for:

- Voice and voice command recognition
- Hand gestures
- Body Motion Tracking

The organisation has been created on November 2010, with the website going public on December 8th. One of the main members is PrimeSense, the company behind the technology used in the Kinect.

4.2.2 Nui.Vision Library

It is a library developed by Vangos Pterneas (vangos.eu) licensed under The Microsoft Public License (Ms-PL), with this library it is possible to make image acquisition and user tracking by using the classes defined in the library itself.

4.2.3 User interface

The user interface has been developed by using WPF and C#. This UI can be divided in 2 parts:

- On the left we can see the image which is acquired by the Kinect, with some spots on the user's body, these spots represent the position of some body parts such us: Head, Neck, LeftShoulder, LeftElbow, LeftHand, RightShoulder, RightElbow, RightHand, Torso, LeftKnee, LeftHip, LeftFoot, RightKnee, RightHip, RightFoot.
- On the top-right there are the X, Y, Z coordinates of the interesting body parts: Shoulder, Elbow and Hand; these coordinates are shown in real-time.
- In the middle right there are :
  - The user ID;
  - A button to select the side to use for making gesture;
  - The status of the system (enable, disable);

- In the middle left some label are shown:
    - The actual pose that has been recognized
    - The final gesture which has been recognized by a combination of two poses;
- In the bottom there are the features which have been extracted in real time by the system.

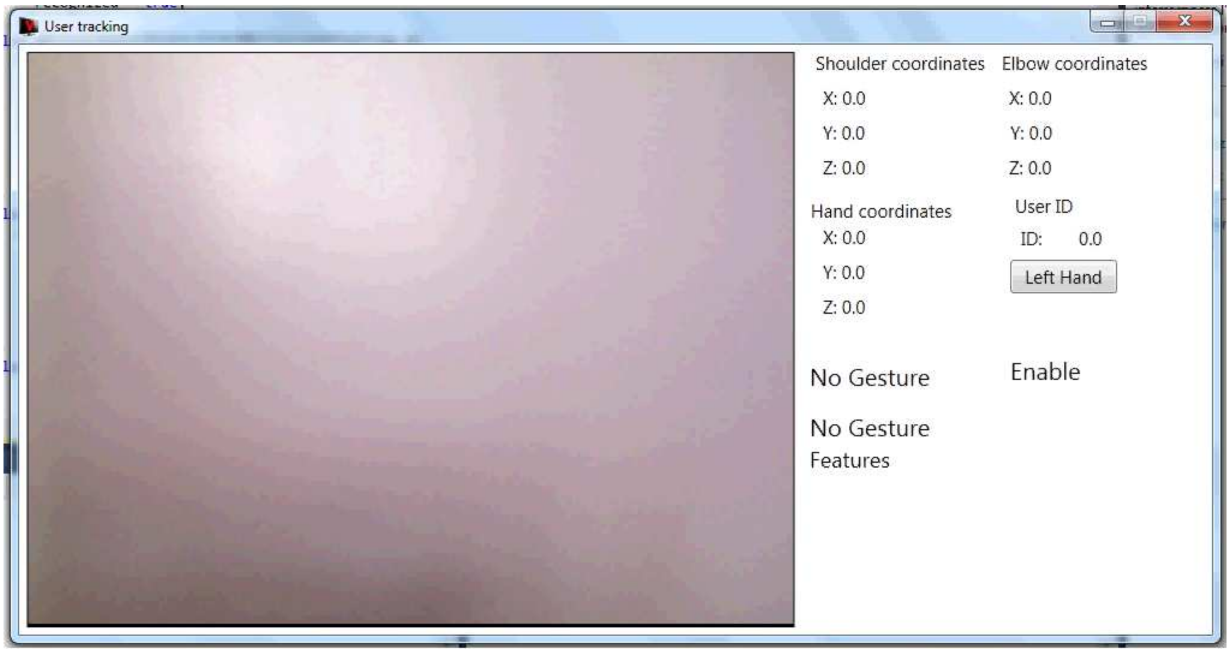


Figure 11: User Interface

4.2.4 FSA

In order to recognize the gestures a FSA was implemented. A finite-state machine (FSM) or finite-state automaton (plural: *automata*), or simply a state machine, is a mathematical abstraction sometimes used to design digital logic or computer programs. It is a behavior model composed of a finite number of states, transitions between those states, and actions, similar to a flow graph in which one can inspect the way logic runs when certain conditions are met. It has finite internal memory, an input feature that reads symbols in a sequence, one at a time without going backward; and an output feature, which may be in the form of a user interface, once the model is implemented. The operation of an FSM begins from one of the states (called a *start state*), goes through transitions depending on input to different states and can end in any of those available, however only a certain set of states mark a successful flow of operation (called *accept states*). Finite-state machines can solve a large number of problems, among which are electronic design automation, communication protocol design, parsing and other engineering applications. In biology and artificial intelligence research, state machines or hierarchies of state machines are sometimes used to describe neurological systems and in linguistics—to describe the grammars of natural languages.

A current *state* is determined by past states of the system. As such, it can be said to record information about the past, i.e., it reflects the input changes from the system start to the present moment. The number and names of the states typically depend on the different possible states of the memory, e.g. if the memory is three bits long, there are 8 possible states. A *transition* indicates a state change and is described by a condition that would need to be fulfilled to enable the transition. An *action* is a description of an activity that is to be performed at a given moment. There are several action types:

- Entry action which is performed *when entering* the state;
- Exit action which is performed *when exiting* the state;
- Input action which is performed depending on present state and input conditions;
- Transition action which is performed when performing a certain transition.

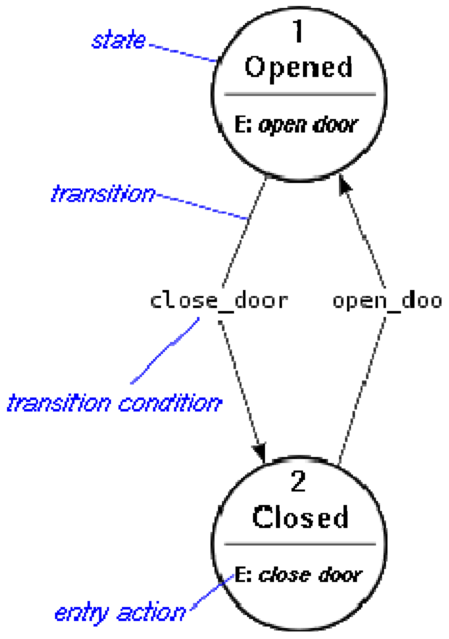


Figure 12: an example of FSA

An FSM can be represented using a state diagram (or state transition diagram) as in figure 1 above. In addition to their use in modelling reactive systems, finite state automata are significant in many different areas, including electrical engineering, linguistics, computer science, philosophy, biology, mathematics, and logic. Finite state machines are a class of automata studied in automata theory and the theory of computation. In computer science, finite state machines are widely used in modelling of application behaviour, design of hardware digital systems, software engineering, compilers, network protocols, and the study of computation and languages.

4.3 The Proposed Algorithm

These are the main steps of this system:

- Image acquisition;
- Pre-Processing; You need to explain how they find the control points.
- Feature extraction;
- Gesture recognition.

4.3.1 Image acquisition (to be extended)

This step is done by using:

- A Kinect camera
- OpenNI
- And a library Nui.vision, developed by Vangos Pterneas.

The image acquisition is done by taking the image from the Kinect camera and using the OpenNi framework and Nui.Vision library to bring it on a pc screen.

There are two kind of image which are used:

- the RAW image and
- the depth image.

The first image is used to show in real time the RGB visual of the camera and the X,Y coordinates of the interesting object (in this case the human body), the second is used to extract the data about the distance (Z coordinate) of the object, this second kind of image is taken by using the tow depth sensors.

This step uses the OpenNi framework and its classes to take the data from the Kinect, there are special classes (ImageGenerator, DepthGenerator) that make this part, this classes take the data and pass them to the screen by using a C# class (Image) to show them.

4.3.2 Pre-Processing

The acquired image must be processed in a proper way in order to take the data, this step is called Pre-processing. The Pre-Processing step is based on the tracking of the user’s body obtained by using the Nui.Vision library and OpenNI which allows the user to be tracked by Kinect, after he stands in the calibration pose (psi-like pose). This particular pose is required by OpenNi itself in order to make the tracking of the user. Here is how this step works:

- The application wants to track the motion of a human figure in a 3D scene. This requires a special class called **User Generator**. This specific user generator obtains its data from a **Depth Generator**. A depth generator is a class that is implemented by a sensor (in our case the kinect), which takes raw sensory data from a depth sensor (for example, a stream of X frames per second) and outputs a depth map, in our case the raw data from the depth sensors are merged with the raw data from the RGB sensor. This Classes use some internal capabilities that allow them to get the most important data for our system: the position of some parts of the body. This Capabilities are:
    - **Pose Detection:** Enables a user generator to recognize when the user is posed in a specific position, in our case this specific position is the calibration pose (psi-like pose). When the system recognizes this pose it uses the next capability;
    - **Skeleton:** Enables a user generator to output the skeletal data of the user. This data includes the location of the skeletal joints (such as neck, head, feet, hands, elbows and so on), the ability to track skeleton positions and the user calibration capabilities.
    - **User Position:** Enables a Depth Generator to optimize the output depth map that is generated for a specific area of the scene.

After the calibration, some spots appear on the user's body which is tracked on the pc screen; these spots represent the position of some body parts of the user, and they are obtained from the skeletal joints which have been tracked by using the Skeleton Capabilities. This post are: Head, Neck, LeftShoulder, LeftElbow, LeftHand, RightShoulder, RightElbow, RightHand, Torso, LeftKnee, LeftHip, LeftFoot, RightKnee, RightHip, and RightFoot.

In this project the interesting body parts are:

- Left/RightShoulder
- Left/RightElbow,
- Left/RightHand.

The spots have three different colors:

- Blue for the body parts which are not involved in the gesture recognition such as: Neck, Head, Torso Torso, LeftKnee, LeftHip, LeftFoot, RightKnee, RightHip, and RightFoot.
- Yellow for the part of the right arm.
- Red for the part of the left arm.

These spots have as a center the real time X,Y coordinates of the body parts involved in the gesture recognition. This step prepares the data to be extracted as features will be used in the next steps.

4.3.3 Feature Extraction

The purpose of this step is to extract the features and group them in a proper way that, The Feature are extracted to be processed for the next step; the extracted features are:

- The Position (X, Y, Z) of the hand and elbow compared to a fixed point, the shoulder.

The way to extract the features is made frame by frame, this way to extract the features is more accurate than other such as sampling with a fixed rate or using a threshold.

In order to extract the position features there are some threshold used to compare each coordinate of the elbow and hand with the shoulder's one, these threshold are used to get a good approximation in comparison between the positions.

Here is a description of these features:

- Position compare to the shoulder:
  - X coordinate (elbowLR, handLR): 0 means same position, 1 at the right side, -1 at the left side;
  - Y coordinate (elbowOver, handover) : 0 means same position, 1 lower , -1 upper;
  - Z coordinate (elbowCF, HandCF): 0 means same position, 1 farther to the camera, -1 closer to the camera.

Position						
Elbow/Hand	**value**	**meaning**	**value**	**meaning**	**value**	**meaning**
X	**-1**	**left**	**0**	**same position**	**+1**	**right**
Y	**-1**	**upper**	**0**	**same position**	**+1**	**lower**
Z	**-1**	**closer**	**0**	**same position**	**+1**	**farther**

Table 1: Description of the features' values and their meanings

4.3.4 Gesture recognition

This step is done using a FSA, we have chosen this tool, instead of other similar tools such as HMM or Neural Network, because we have a deterministic situation since we use as a reference of our system the user's shoulder, this implies that we know the position of the elbow and the hand and the shoulder frame by frame, without any problem. In this section we will describe the FSA and the State Transitions involved in it. The extracted features of the previous step are used to feed this FSA which is composed by 6 state:

- No gesture;
- Come Here;
- Go Away;
- Stop;
- Turn Left;
- Turn Right.

It is possible to go from a state to another without any constrains, it is necessary that the State Transition of a specific state is found that the system will recognize the gesture. The State Transitions are described in the follow table.

Poses → *Output State ↓*	*First pose*	*Second pose*
Go Away	Lower	Front
Stop	Front	Stop Pose
Turn Right	Front	Turn Right Pose
Turn Left	Front	Turn Left Pose
Come Here	Front	Lower
No Gesture	otherwise	otherwise

Table 2: Transition table

The corresponding gesture will be recognized, if the system finds one of the first five couple of poses, this means that the system finds the corresponding State Transition, if the system finds another couple of poses the system will not recognize any gesture (No Gesture state).

Now we will describe the FSA for every gesture that we have implemented.

Stop:

From one of the five states the user can go to the Stop State, if he makes the Front Pose and then the Upper pose. Of course the user can make also 2 or more consecutive Stop Gestures.

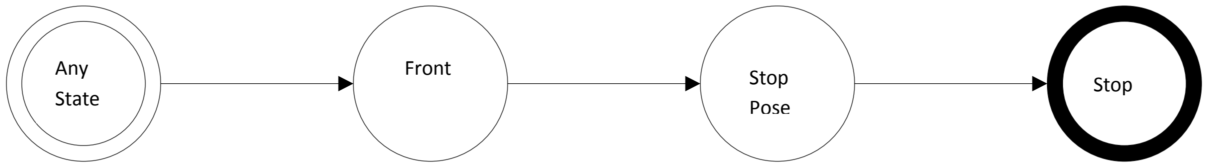


Figure 13: FSA for the Stop Gesture

Turn Right:

From one of the five states the user can go to the Turn Right State, if he makes the Front Pose and then the Right pose. Of course the user can make also 2 or more consecutive Turn Right Gestures.

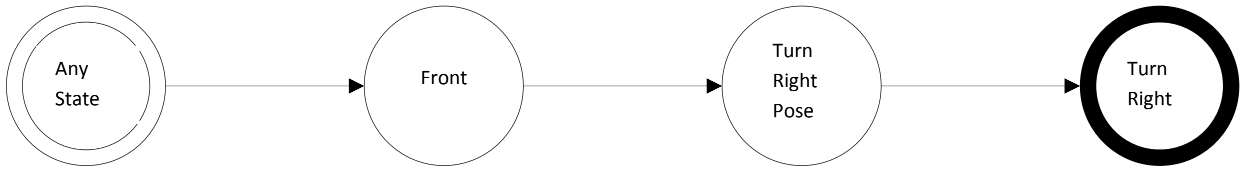


Figure 14: FSA for the Turn Right Gesture

Turn Left:

From one of the five states the user can go to the Turn Left State, if he makes the Front Pose and then the Left pose. Of course the user can make also 2 or more consecutive Turn Left Gestures.

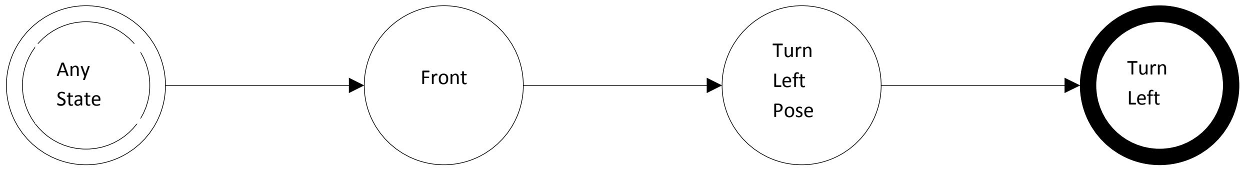


Figure 14: FSA for the Turn Left Gesture

Come Here:

From one of the five states the user can go to the Come Here State, if he makes the Front Pose and then the Lower pose. Of course the user can make also 2 consecutive Come Here Gestures.

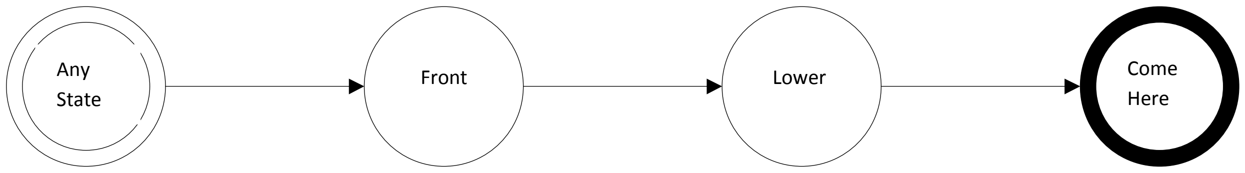


Figure 156: FSA for the Come Here Gesture

Go Away:

From one of the five states the user can go to the Go Away State, if he makes the Lower Pose and then the Front pose. Of course the user can make also 2 or more consecutive Go Away Gestures.

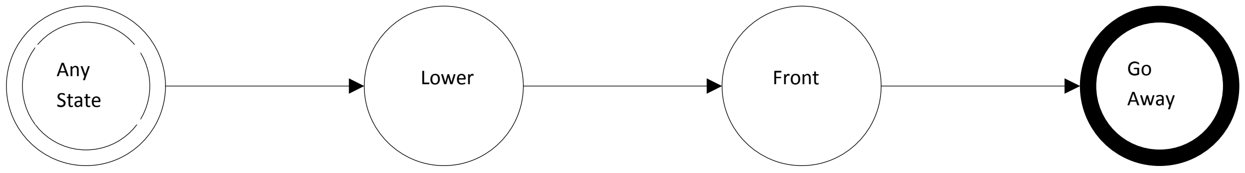


Figure 17: FSA for the Go Away Gesture

No gesture:

From one of the five state the user can go to the No Gesture State, if he makes as X a pose like Right, Left or Upper and then every pose he wants or if he make as X the Lower Pose and as Y a pose like Right, Left or Upper.

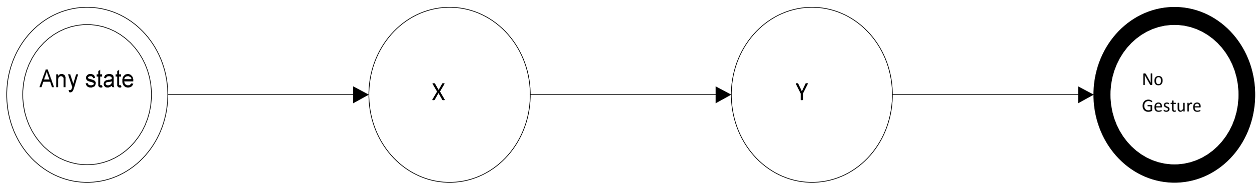


Figure 18: FSA for the No Gesture Gesture

5 Experimental Results

In this section we will talk about the experimental results of this project, we will divided it in three kinds of test:

- Result of pose recognition
- Result of gesture recognition
- Test in different condition such as:
  - Lightning,
  - Speed of the arm movement
  - Recognition if the User moves the entire body while is making the gesture.

5.1 Results of Pose Recognition Test

This Section describes the test which have been done to check the pose recognition rate of the system.

Gesture →	**Stop Pose**		**Turn Left Pose**		**Turn Right Pose**		**Front Pose**		**Lower Pose**	
Person↓	**Done**	**Recognized**	**Done**	**Recognized**	**Done**	**Recognized**	**Done**	**Recognized**	**Done**	**Recognized**
P1	10	10	10	10	10	9	60	60	20	20
P2	10	9	10	7	10	6	56	56	24	21
P3	10	8	10	10	10	10	48	45	18	17
P4	10	10	10	10	10	10	50	50	20	20
P5*	17	13	11	11	12	12	63	61	23	23
P6	6	6	5	4	5	5	25	25	9	9
Recognition Rate	63	56	56	52	57	52	302	297	114	110
%Recognition	89%		93%		91%		98%		96%	

Table 3: Pose Recognition Test T table (* low light condition)

Since we used the same videos to test the Gesture Recognition rate and the Pose Recognition rate we counted the number of each pose. This means that for the Front pose we have a huge number since it is involved in all the gesture, while the Lower Pose is involved in only two gesture (so in this pose we have a smaller number than the Front Pose), and the Stop, Turn Left and Turn Right pose are involved in only one gesture.

5.2 Results of Gesture Recognition Test

This Section describes the test which have been done to check the gesture recognition rate of the system. We used the same video of the Pose Recognition Tests.

Gesture →	**Stop**		**Turn Left**		**Turn Right**		**Come Here**		**Go Away**	
Person↓	**Done**	**Recognized**	**Done**	**Recognized**	**Done**	**Recognized**	**Done**	**Recognized**	**Done**	**Recognized**
P1	10	10	10	10	10	9	10	10	10	10
P2	10	9	10	7	10	6	13	10	11	11
P3	10	8	10	10	10	10	10	10	8	7
P4	10	10	10	10	10	10	10	10	10	10
P5*	17	13	11	11	12	12	15	15	8	8
P6	6	6	5	4	5	5	5	5	4	4
Recognition Rate	63	56	56	52	57	52	63	60	51	50
%Recognition	89%		93%		91%		95%		98%	

Table 3: Gesture Recognition Test table (* low light condition)

5.3 Results of different condition test

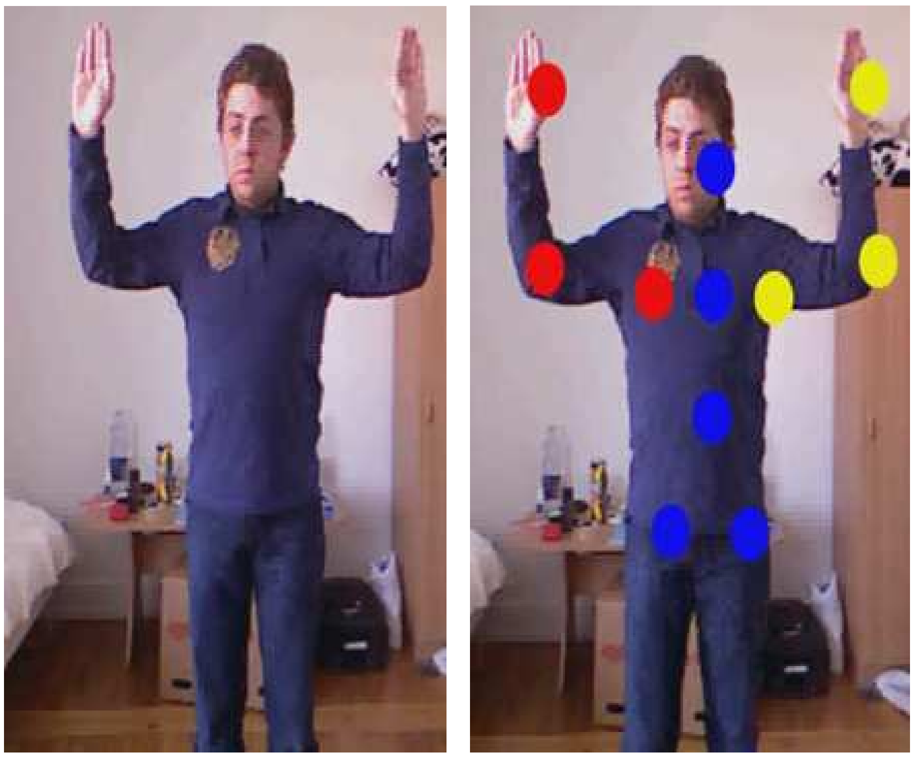
In this section we test the system in different conditions:

- Speed of the Arm's Movement: in this case the system does not present any problem to recognize the gestures and the poses even if the user moves the arm fast or slowly.
- Light condition: the system has been tested with:
    - Natural light (during the day and in a room),
    - Low light, in this case the body results almost black but the system is able to detect, track the body and recognize the gesture with a good rate.

  The system is not affected by any problem regarding the light conditions.

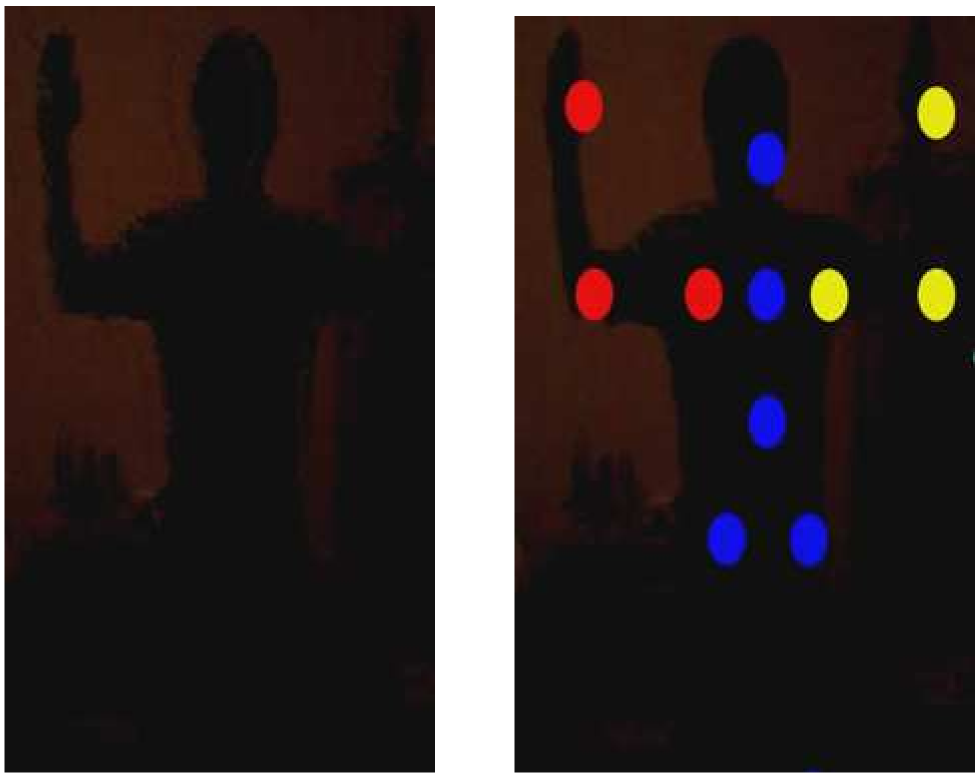
In the next figures we will show the light conditions with which we have tested the system.

As we can see in the figures below the system works fine even in low light conditions, this is because it works with the depth image and not with the RGB image, so the system is robust to light condition.



(a) (b)

Figure 20: Normal light condition: the user is standing in front of the camera (a), the user has been tracked by the system



(a) (b)

Figure 20: Low light condition: the user is standing in front of the camera (a), the user has been tracked by the system

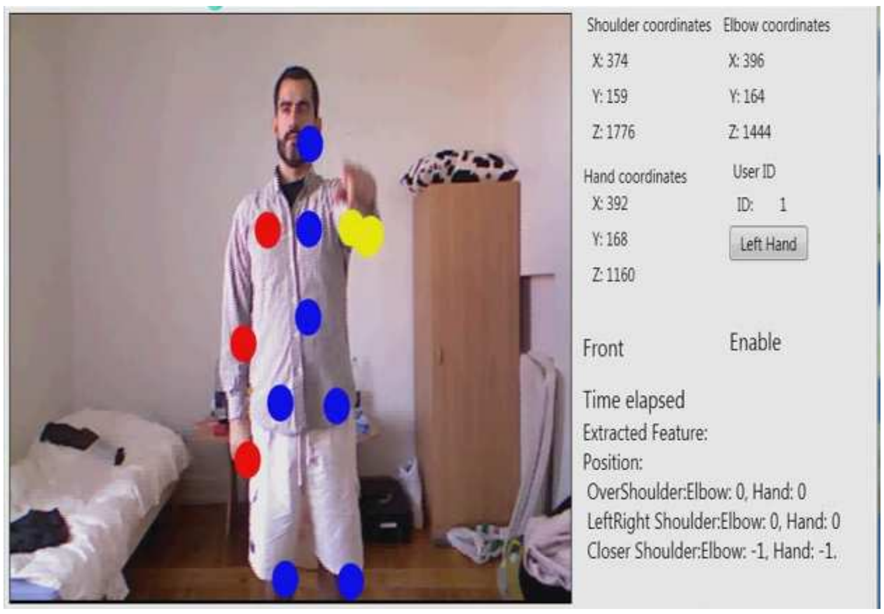
5.4 Reasons of the failure and working of the system

In this section we will describe the reasons of failure and working. For the first case we have identified the main reason of failure is that the system fails to recognize a pose; this situation can be due to:

- The user does not respect the timeout period, after that the system recognizes a gesture the user must wait a few seconds to make another one (the user interface shows a label "Disable" during this period), so if he begins to make a new gesture after having completed another one the system will not recognize the correct gesture.
- After a gesture recognition the user stands in a pose like Stop Pose, Turn Right Pose and Turn Left Pose to respect the timeout period but these poses are invalid to start a new gesture, there are only two start poses are: Front and Lower.
- The user is too close to the camera so the hand can go out of screen and some pose cannot be recognized (for example the Stop pose because the hand is too high or the Turn Right/Left pose because the hand is near to the border of the screen) .

Now we will show some figures about these failures and some figures when the system is working fine.

The figures 21(a) and 21(b) show that the system is working, it recognizes the first (Front) and the second (Lower) pose this means that the Come Here Gesture has been Recognized.

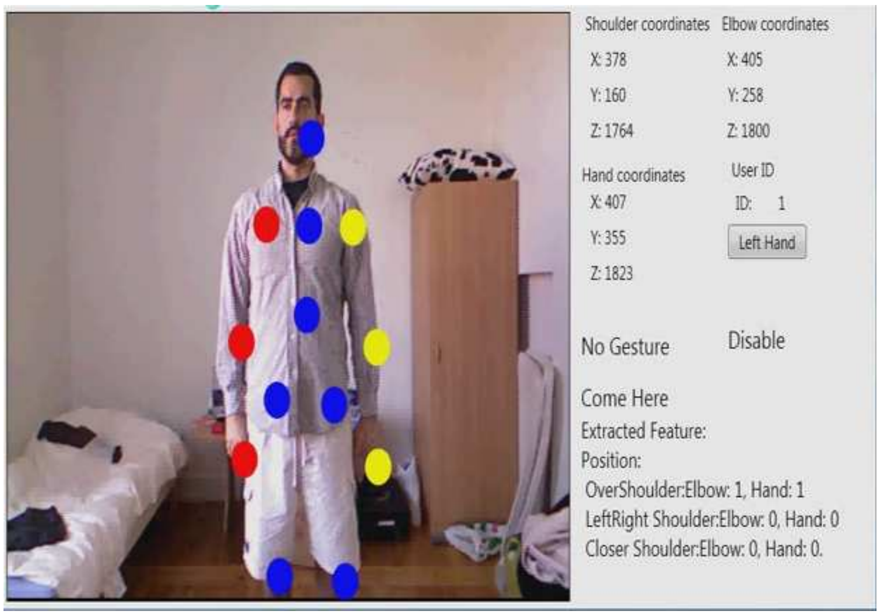


```
Shoulder coordinates   Elbow coordinates
X: 374                 X: 396
Y: 159                 Y: 164
Z: 1776                Z: 1444
Hand coordinates       User ID
X: 392                 ID:    1
Y: 168                 Left Hand
Z: 1160

Front                  Enable

Time elapsed
Extracted Feature:
Position:
 OverShoulder:Elbow: 0, Hand: 0
 LeftRight Shoulder:Elbow: 0, Hand: 0
 Closer Shoulder:Elbow: -1, Hand: -1.
```

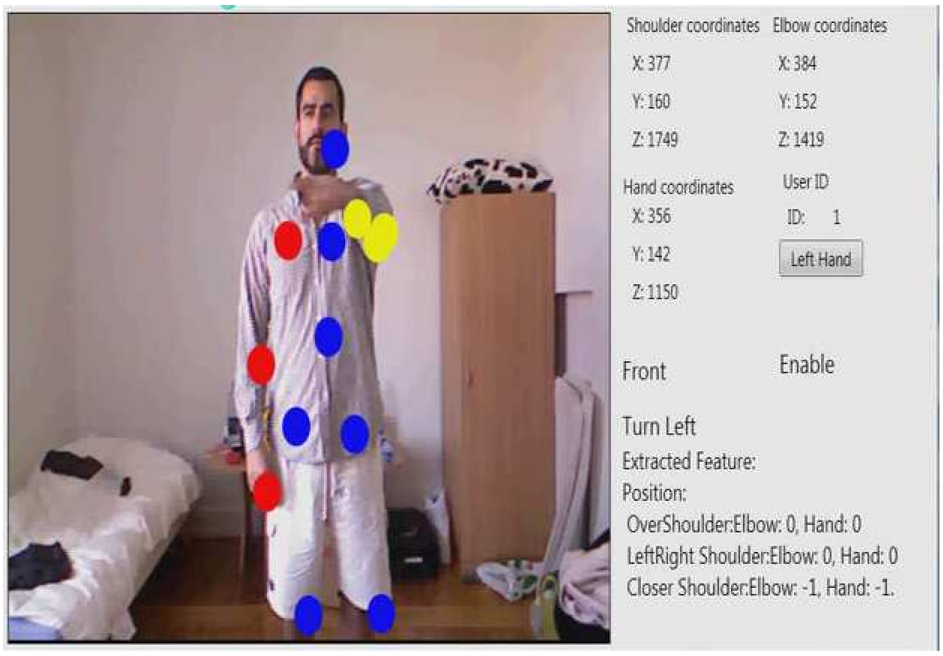
Figure 21(a) : the system is working


Shoulder coordinates
X: 378
Y: 160
Z: 1764
Elbow coordinates
X: 405
Y: 258
Z: 1800
Hand coordinates
X: 407
Y: 355
Z: 1823
User ID
ID: 1

No Gesture
Disable
Come Here
Extracted Feature:
Position:
OverShoulder:Elbow: 1, Hand: 1
LeftRight Shoulder:Elbow: 0, Hand: 0
Closer Shoulder:Elbow: 0, Hand: 0.

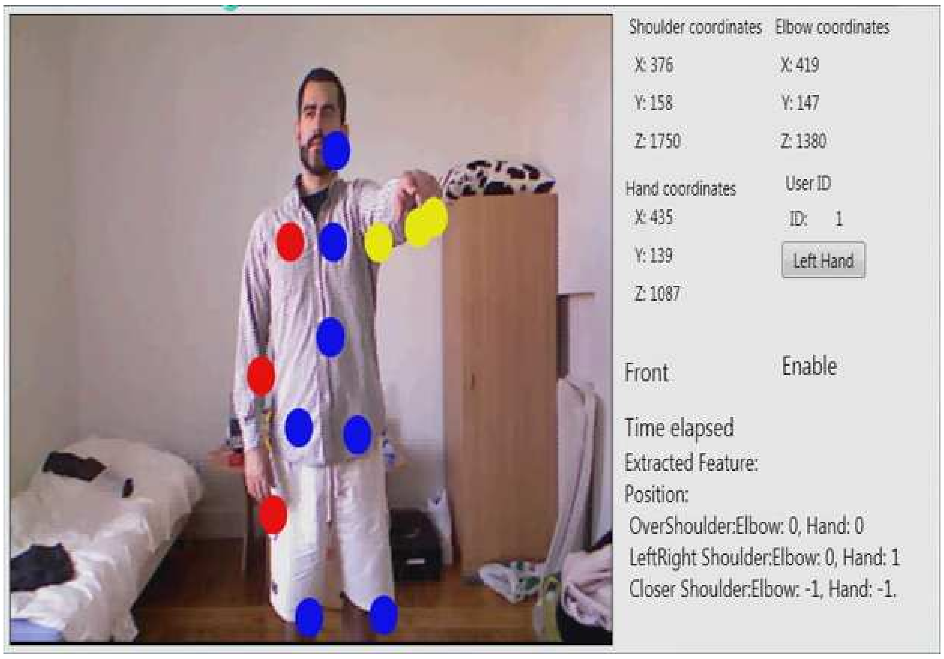
Figure 21(b): the system is working

Here is an example of failure (Figures 22(a) and (b)), the user stands in the Front Pose for too much time and the system goes in time out.


Shoulder coordinates
X: 377
Y: 160
Z: 1749
Elbow coordinates
X: 384
Y: 152
Z: 1419
Hand coordinates
X: 356
Y: 142
Z: 1150
User ID
ID: 1

Front
Enable
Turn Left
Extracted Feature:
Position:
OverShoulder:Elbow: 0, Hand: 0
LeftRight Shoulder:Elbow: 0, Hand: 0
Closer Shoulder:Elbow: -1, Hand: -1.

Figure 22(a): Time Elapsed Failure



Shoulder coordinates
X: 376
Y: 158
Z: 1750

Elbow coordinates
X: 419
Y: 147
Z: 1380

Hand coordinates
X: 435
Y: 139
Z: 1087

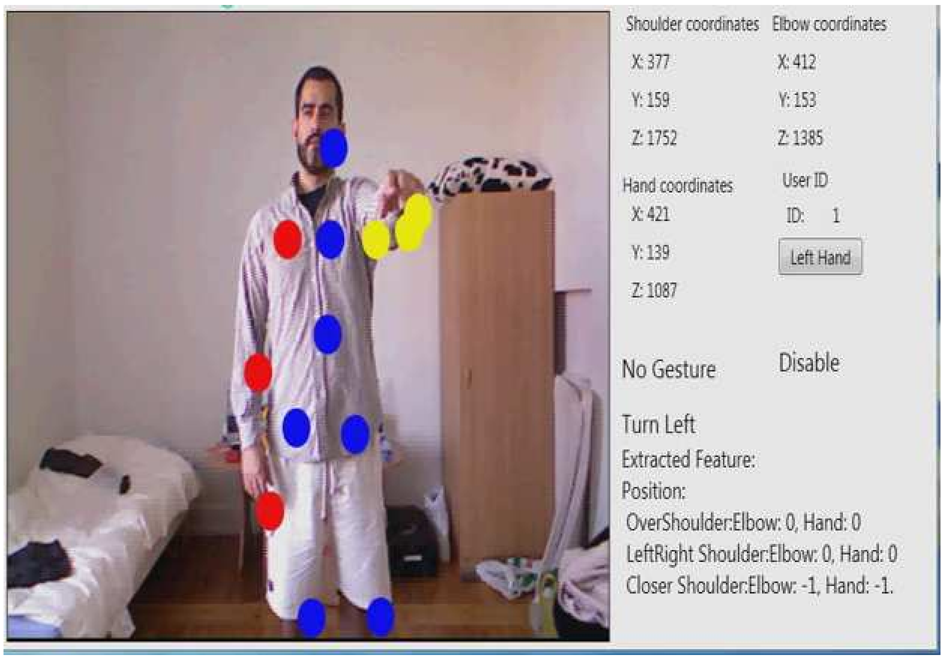
User ID
ID: 1

Front   Enable

Time elapsed
Extracted Feature:
Position:
OverShoulder:Elbow: 0, Hand: 0
LeftRight Shoulder:Elbow: 0, Hand: 1
Closer Shoulder:Elbow: -1, Hand: -1.

Figure 22(b): Time Elapsed Failure

Figures 23(a) and (b) show an user too close to the camera so wheh he tries to make the Stop Pose the hand goes out of the screen and the system cannot extract the right features of the hand.



Shoulder coordinates
X: 377
Y: 159
Z: 1752

Elbow coordinates
X: 412
Y: 153
Z: 1385

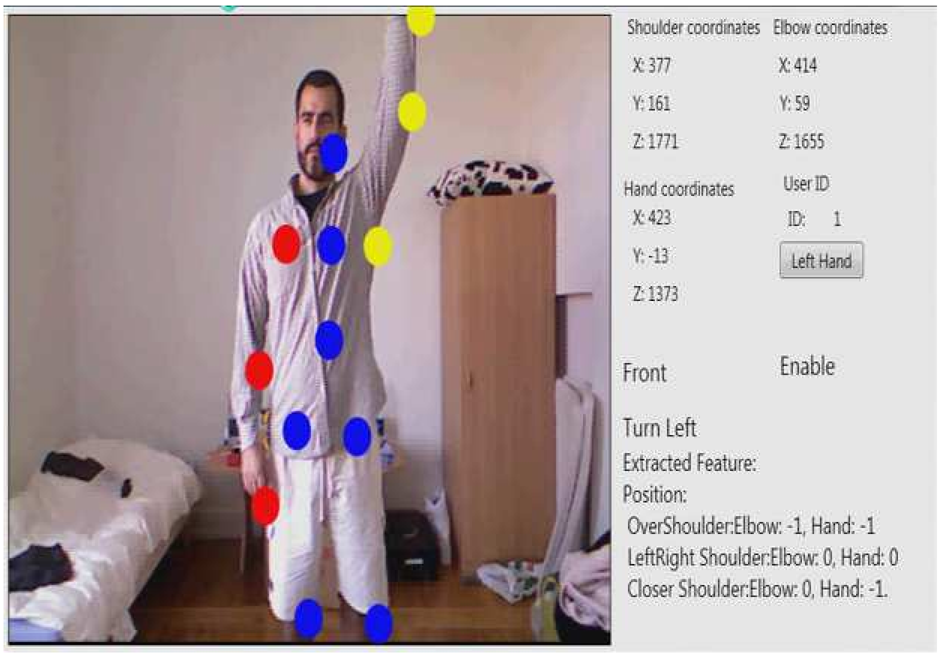
Hand coordinates
X: 421
Y: 139
Z: 1087

User ID
ID: 1

No Gesture   Disable

Turn Left
Extracted Feature:
Position:
OverShoulder:Elbow: 0, Hand: 0
LeftRight Shoulder:Elbow: 0, Hand: 0
Closer Shoulder:Elbow: -1, Hand: -1.

Figure 23(a): User too close to the camera


Shoulder coordinates
X: 377
Y: 161
Z: 1771

Elbow coordinates
X: 414
Y: 59
Z: 1655

Hand coordinates
X: 423
Y: -13
Z: 1373

User ID
ID: 1

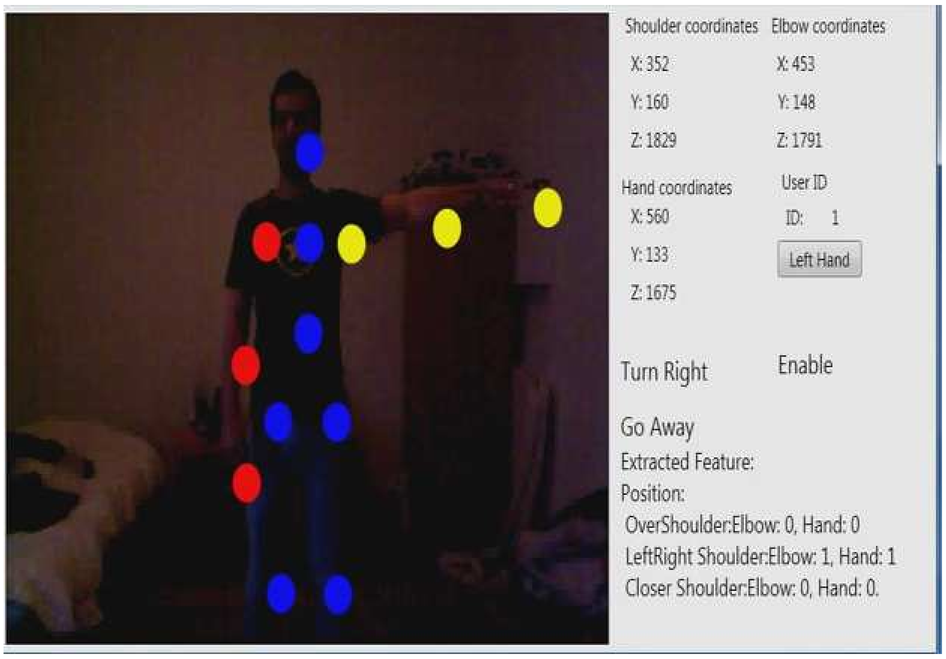
Front

Enable

Turn Left
Extracted Feature:
Position:
OverShoulder:Elbow: -1, Hand: -1
LeftRight Shoulder:Elbow: 0, Hand: 0
Closer Shoulder:Elbow: 0, Hand: -1.

Figure 23(b): User too close to the camera

Here (Figures 24(a) and (b)) the users is standing in the Turn Right Pose for too much time the system recognizes this pose as a first pose, but this pose are invalid to start a new gesture, this situation leads to the No Gesture state.


Shoulder coordinates
X: 352
Y: 160
Z: 1829

Elbow coordinates
X: 453
Y: 148
Z: 1791

Hand coordinates
X: 560
Y: 133
Z: 1675

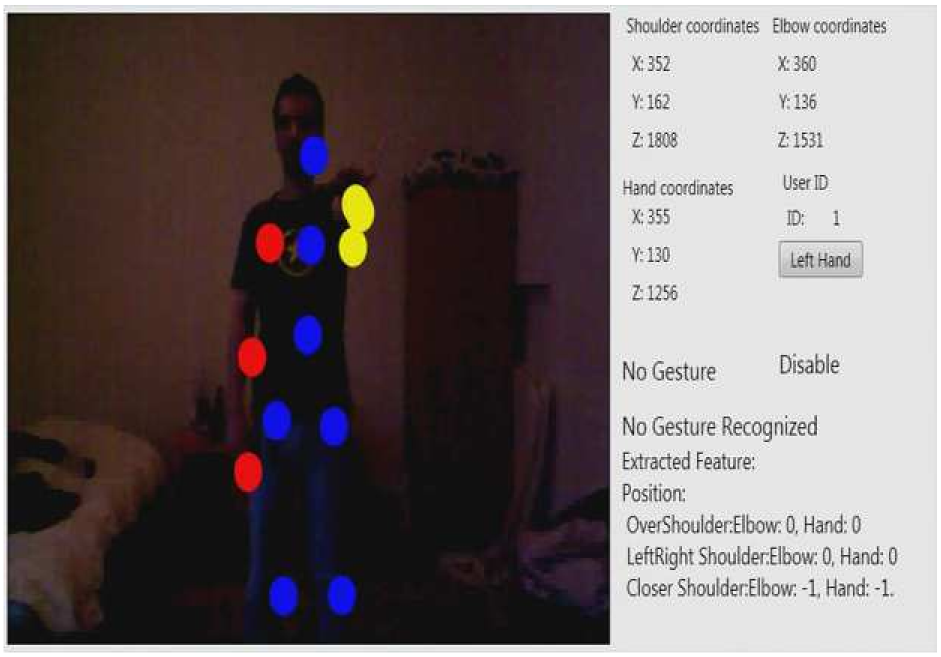
User ID
ID: 1

Turn Right

Enable

Go Away
Extracted Feature:
Position:
OverShoulder:Elbow: 0, Hand: 0
LeftRight Shoulder:Elbow: 1, Hand: 1
Closer Shoulder:Elbow: 0, Hand: 0.

Figure 24(a): User is standing in a wrong first pose



Shoulder coordinates
X: 352
Y: 162
Z: 1808

Elbow coordinates
X: 360
Y: 136
Z: 1531

Hand coordinates
X: 355
Y: 130
Z: 1256

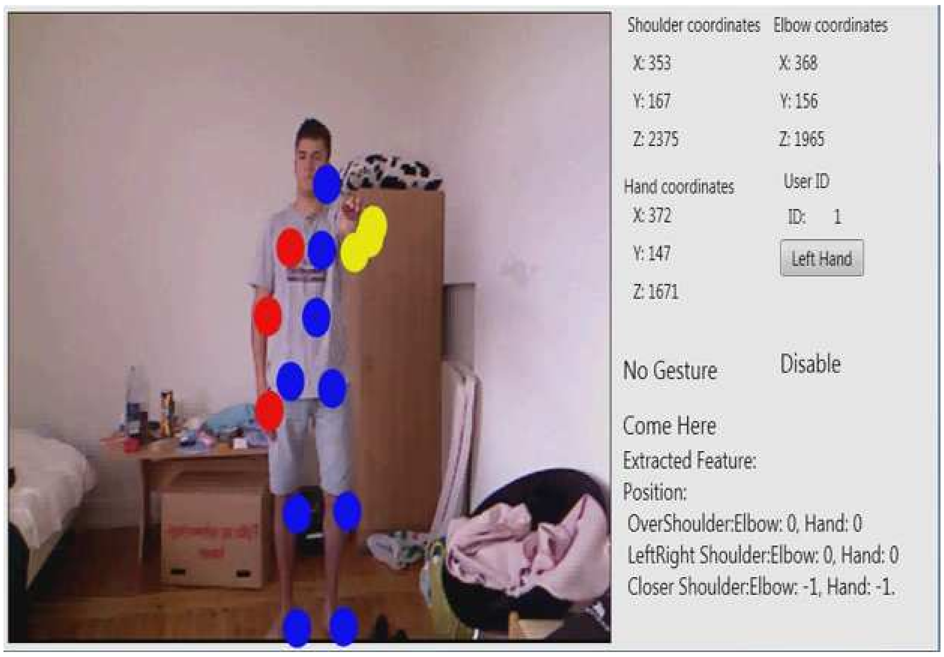
User ID
ID: 1

No Gesture
Disable

No Gesture Recognized
Extracted Feature:
Position:
OverShoulder:Elbow: 0, Hand: 0
LeftRight Shoulder:Elbow: 0, Hand: 0
Closer Shoulder:Elbow: -1, Hand: -1.

Figure 24(b): User is standing in a wrong first pose

Finally the last case of failure (Figure 25 (a),(b),(c)) the users does not respect the timeout period, so the system will recognize the second pose made by the user as First Pose and the follow pose that the user maks like the Second Pose of a gesture, this can lead not to recognize a gesture.



Shoulder coordinates
X: 353
Y: 167
Z: 2375

Elbow coordinates
X: 368
Y: 156
Z: 1965

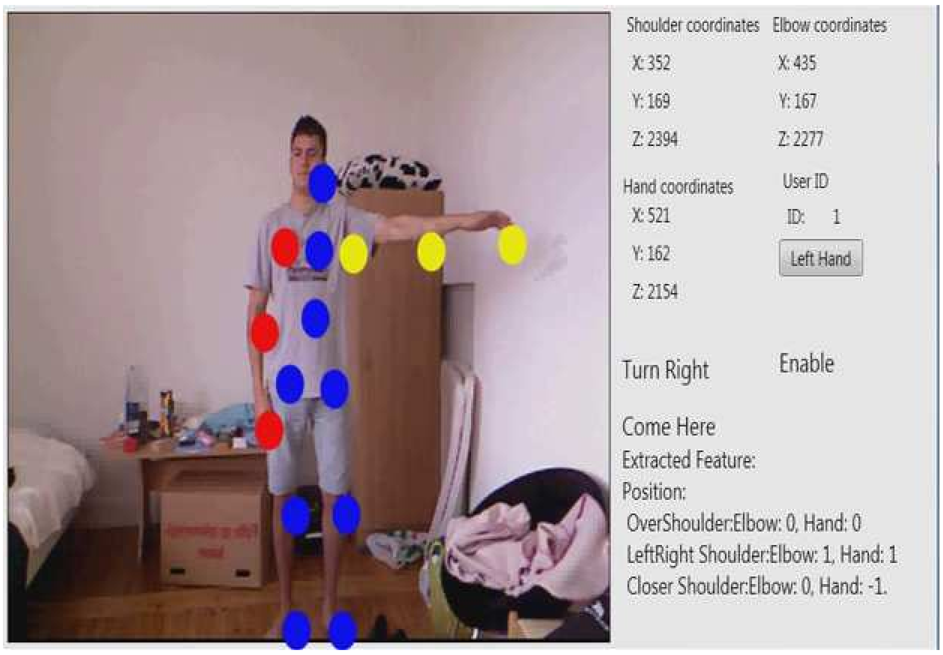
Hand coordinates
X: 372
Y: 147
Z: 1671

User ID
ID: 1

No Gesture
Disable

Come Here
Extracted Feature:
Position:
OverShoulder:Elbow: 0, Hand: 0
LeftRight Shoulder:Elbow: 0, Hand: 0
Closer Shoulder:Elbow: -1, Hand: -1.

Figure 25(a): the user does not wait the timeout and make a pose (Front) which will not have been recognized by the system.



Turn Right Enable

Come Here

Extracted Feature:

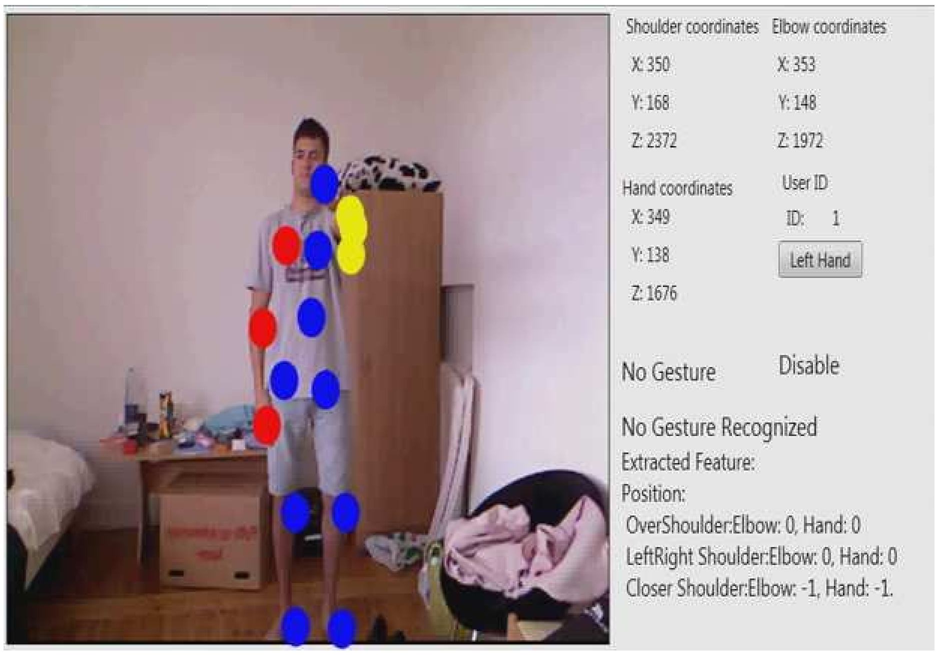
Position:

OverShoulder:Elbow: 0, Hand: 0

LeftRight Shoulder:Elbow: 1, Hand: 1

Closer Shoulder:Elbow: 0, Hand: -1.

Figure 25(b): This should be the second Pose but the system recognize it as the First Pose...



No Gesture Disable

No Gesture Recognized

Extracted Feature:

Position:

OverShoulder:Elbow: 0, Hand: 0

LeftRight Shoulder:Elbow: 0, Hand: 0

Closer Shoulder:Elbow: -1, Hand: -1.

Figure 25(c): ... here the user make again the Front Pose but the system does not recognize any gesture in this combination : Turn right/Front

6 Future Works

As we say above the aim of this project is to drive a robot by making gesture in front of the kinect, in the future, we will test the system to drive a remote robot to check if the system works properly. For this future tests the robot will be the 3MO.R.D.U.C., “3rd version of the MObile Robot DIEES University of Catania”, shown in Figure 13, is a wheeled mobile robot in differential drive configuration.

Here are some of its features:

- the movement is accomplished by two 40W DC motors, Maxon F2260, and the motor axes is linked with a gear box (gear ratio 1:19).;
- two rubber wheels are linked with the gear box axis and a third castor wheel is free to rotate, facilitating so the execution of the curves.
- the robot structure has three shelves linked together. On the lower shelf there are two lead batteries (12V/18Ah), which provide the power supply. The robot autonomy is about 30 min. for continuous working.
- an on board electronic rack controls the modules of the robot (motion, sensors and communication). On the robot there are several sensors monitoring the workspace and the robot state (bumper, encoder, laser, sonar and stereocam). The electronic part and sensors on board are analyzed in the following section.
- Finally, on the top shelf there is a laptop where the robot control application is runs.

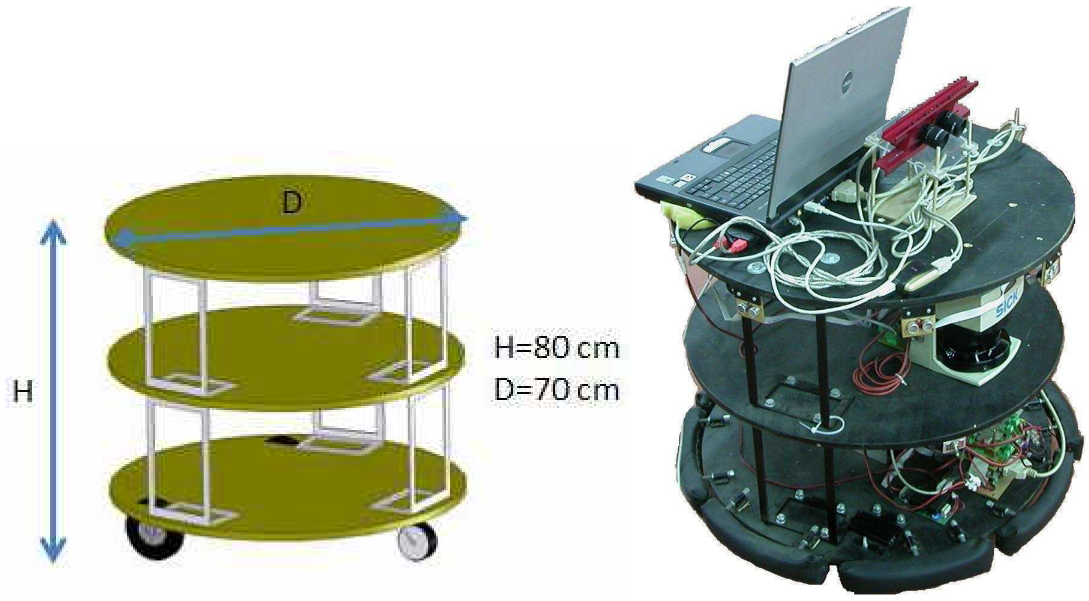


Figure 169: 3 MO.R.D.U.C. robot

7 Conclusions

In this report we have explained how we developed a system to recognize the gestures using the Kinect camera, and we have described which kind of gesture we can use and how the system recognized them.

The system was developed by using the OpenNI Framework to get the image and the Library Nui.Vision to track the user's body and extract the feature, but in order to develope the true part of the gesture recognition a FSA was used, this FSA is fully connected so we can go from a state to the others without any other condition except the ones that we have explained in the section 4.3.4. This kind of algorithm takes inspiration from some papers that we read but they used the HMM, a probabilistic kind a FSA, in our case since we have pre-fixed type of pose with their specific feature we can use a deterministic FSA.

After developing the system we have tested it with different people and different condition, the results of these test have been good both in low-light conditions, if the user is moving while making gestures, and with different users.

In the future the system can be developed by including definition of new gestures for more specific action, and by extending the use of the system to more than one person and using that for controlling a remote robot.

References

[1] S. Malassiotis, N. Aifanti and M. G. Strintzis, A Gesture Recognition System Using 3D Data, 3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on, Publication Year: 2002 , Page(s): 190 - 193

[2] Kye Kyung Kim, Keun Chang Kwak and Su Young Chi, Gesture Analysis for Human-Robot Interaction, Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference, Volume: 3 Publication Year: 2006 , Page(s): 4 pp. - 1827

[3] Yikai Fang, Kongqiao Wang, Jian Cheng and Hanqing Lu, Real-Time Hand Gesture Recognition Method, Multimedia and Expo, 2007 IEEE International Conference on, Publication Year: 2007 , Page(s): 995 - 998

[4] Xiujuan Chai, Yikai Fang and Kongqiao Wang, Robust Hand Gesture Analysis And Application in Gallery Browsing, Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on, Publication Year: 2009 , Page(s): 938 - 941

[5] Chenglong Yu, Xuan Wang, Hejiao Huang, Jianping Shen and Kun Wu, Vision-Based Hand Gesture Recognition Using Combinational Features, Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2010 Sixth International Conference on, Publication Year: 2010 , Page(s): 543 - 546

[6] Liu Yun and Zhang Peng, An Automatic Hand Gesture Recognition System Based on Viola-Jones Method and SVMs, Computer Science and Engineering, 2009. WCSE '09. Second International Workshop on, Publication Year: 2009 , Page(s): 72 - 76

[7] Tan Wenjun, Wu Chengdong Zhao Shuying Jiang Li, Dynamic Hand Gesture Recognition Using Motion Trajectories and Key Frames Advanced Computer Control (ICACC), 2010 2nd International Conference on, Publication Year: 2010 , Page(s): 163 - 167

[8]Jagdish Lal Raheja, Radhey Shyam, Umesh Kumar, P. Bhanu Prasad, Real-Time Robotic Hand Control using Hand Gestures, Machine Learning and Computing (ICMLC), 2010 Second International Conference on, Publication Year: 2010 , Page(s): 12 - 16

[9] Mahmoud Elmezain, Ayoub Al-Hamadi and Bernd Michaelis, Improving Hand Gesture Recognition Using 3D Combined Features, Machine Vision, 2009. ICMV '09. Second International Conference on, Publication Year: 2009 , Page(s): 128 - 132

[10] Omer Rashid, Ayoub Al-Hamadi and Bernd Michaelis, A Framework for the Integration of Gesture and Posture Recognition Using HMM and SVM, Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on, Publication Year: 2009 , Page(s): 572 - 577

[11] Jonathan Alon, Vassilis Athitsos, Quan Yuan, and Stan Sclaroff, A Unified Framework for Gesture Recognition and Spatiotemporal Gesture Segmentation, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Publication Year: 2009 , Page(s): 1685 - 1699

[12] Ho-Sub Yoon, Byung-Woo Min, Jung Soh, Young-lae Bae and Hyun Seung Yang, Human Computer Interface for Gesture-Based Editing System, Image Analysis and Processing, 1999. Proceedings. International Conference on, Publication Year: 1999, Page(s): 969 - 974

[13] Xiaoming Yin and Ming Xie, Hand Gesture Segmentation, Recognition and Application, Computational Intelligence in Robotics and Automation, 2001. Proceedings 2001 IEEE International Symposium on, Publication Year: 2001 , Page(s): 438 - 443

[14] Chao Hu, Max Qinghu Meng, Peter Xiaoping Liu and Xiang Wang, Visual Gesture Recognition for Human-Machine Interface of Robot Teleoperation, Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, Publication Year: 2003 , Page(s): 1560 - 1565 vol.2

[15] Jochen Triesch and Christoph von der Malsburg, A Gesture Interface for Human-Robot-Interaction, Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on, Publication Year: 1998 , Page(s): 546 - 551

[16] Xiaoming Yin and Xing Zhu, Hand Posture Recognition in Gesture-Based Human-Robot Interaction, Industrial Electronics and Applications, 2006 1ST IEEE Conference on, Publication Year: 2006 , Page(s): 1 - 6

[17] Stefan Waldherr, Roseli Romero and Sebastian Thrun, A Gesture Based Interface for Human-Robot Interaction, AUTONOMOUS ROBOTS, Publication Year: 2011 Page(s) 151-173