POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRONIKI I TECHNIK INFORMACYJNYCH
INSTYTUT AUTOMATYKI I INFORMATYKI STOSOWANEJ

Rok akademicki 2012/2013

PRACA DYPLOMOWA INŻYNIERSKA

Paweł Rabiński

Vision based gesture-driven human-robot interface

Opiekun pracy:
dr inż. Wojciech Szynkiewicz

Ocena pracy: .....

.....

Data i podpis Przewodniczącego

Komisji Egzaminu Dyplomowego

Podziękowanie

Chciałbym podziękować wszystkim osobom, zaangażowanym w pomoc przy powstawaniu tej pracy. Szczególnie chciałbym podziękować mojemu promotorowi, Wojciechowi Szynkiewiczowi za jego cierpliwość i wyrozumiałość oraz pracownikom laboratorium 012 za cenny rady i pomoc w konfiguracji sprzętu do prowadzenia badań.

Streszczenie

Temat: *Interfejs człowiek-robot z użyciem gestów*

Celem niniejszej pracy inżynierskiej było zaprojektowanie i implementacja interfejsu do interakcji człowiek-robot z użyciem gestów. Dodatkowo należało zaimplementować wybrany algorytm podążania za człowiekiem w środowisku bez przeszkód. Zadania te zostały zrealizowane przez wykrywanie i śledzenia szkieletu człowieka za pomocą warstwy oprogramowania NITE uruchamianej na platformie programistycznej OpenNI. Do uzyskiwania obrazu 3D sceny został wykorzystany sensor RGB-D MS Kinect. System miał również informować użytkownika o swoim statusie poprzez komunikaty dźwiękowe generowane przez syntezator mowy. Całość oprogramowania została uruchomiona na platformie ROS zainstalowanej na robocie mobilnym z napędem różnicowym.

Słowa kluczowe: *robot mobilny, Kinect, HRI, ROS, NITE, OpenNI, gesty*

Abstract

The aim of the thesis was to design and implement vision based gesture-driven interface for human-robot interaction. It was also required to implement human following procedure in environment without obstacles. Human detection and tracking was realized by extraction of a skeleton model using NITE middleware, running in OpenNI framework. Analysis of the scene in 3D was performed by use of RGB-D MS Kinect sensor. The designed system must inform human about current status of operation by voice messages generated by speech synthesizer. The whole software was run on ROS framework installed on mobile robot with differential drive.

Keywords: *mobile robot, Kinect, HRI, ROS, NITE, OpenNI, gestures*

Contents

Chapter 1

Introduction

1.1 Motivation

Today we can observe rapid development in the field of mobile devices. Constantly increasing computing power and shock resistant flash memory allow migration of the majority of computer applications from PC to smartphone or tablet. The same technology trend occurs in robotics, where mobile robots are very common subject of research. Various advanced designs perform complex task like bomb detection and disposal, vacuum cleaning, playing football or guiding people in museum. Many of those and other scenarios frequently require contact with human beings, which led to another subject area: human-robot interaction, commonly referred by acronym HRI.

HRI is a multidisciplinary field of research involving robotics, artificial intelligence, natural language processing, psychology and other social sciences. The communication between a robot and human is a complex task and it is achieved by use of dedicated interfaces. We can distinguish three main types of such interfaces: GUIs, TUIs and vision-based ones. The first class, called graphical user interfaces is represented by common input devices like mice or keyboard. They were originally designed for PC computers and due to their static nature are inadequate for mobile robotics purposes. Tangible user interfaces, like Wii-Remote, are more suitable but require special skills and effort from the operator. Vision-based interfaces seem to be the best choice, because they provide hands free communication with much less effort from human side. Therefore, this type of interface has been chosen as a subject of this work.

1.2 Aim of the thesis

Purpose of the thesis is to design a human-robot interface based on vision and implement it on a mobile robotic platform. For a vision system in this work we use video sensor called Kinect, which will be described in details in Chapter 4. Because of working principal based on structured light we limited testing scenarios to indoor environment. The project assumes realization of the following tasks:

- human detection and tracking on camera 3D image
- robot navigation based on human displacement
- human-robot interaction via set of static arm gestures
- voice feedback from the robot based on speech generator

1.3 Thesis outline

This thesis is divided into seven chapters. In Chapter 2 we introduce main concepts related to this work, providing state of art. Chapter 3 presents requirements and main algorithm of the designed system. In Chapter 4 we describe hardware and software components used in this project. Next, in Chapter 5 we provide in details our contribution to the work and test results of the whole system in Chapter 6. Finally, in Chapter 7 we summarize our work and propose issues for possible future development.

Chapter 2

Background

2.1 Human-robot interaction

HRI is a multidisciplinary research field, which main aim is designing and testing robotic systems to be used by or with humans. For effective accomplishment of those goals we utilize various techniques, for example image processing, speech recognition and generation or finally gesture recognition. Considering, which method of perception to choose, we have to keep in mind that it supposed to be user-friendly and provide human with intuitive interface for comfortable interaction with the robot. Moreover, we can distinguish two main categories of interaction [1]:

- remote interaction
- proximate interaction

In the first one, interacting objects do not share the same physical workspace. It is commonly used for long-range navigation [2], where robots operations are seen in camera image stream via network or other medium. In the second group, in which robot and human cooperating with each other in a close proximity is a main subject of this thesis and will be discussed deeply in next chapters. In both groups operations have to assure safety and do not be potentially harmful for the human being. Also the machine should protect itself from damage by limiting workspace or applying obstacle avoidance algorithms. For testing purposes, various simulators are used to protect either robot and human during first test of interactive aplications. Afterwards, however, system parameters have to be tuned onece again, because of unexpected conditions, which are hard to predict in simulation.

Finally, very often social aspects of such a interaction, are considered [1], where human emotions and cognition are taken into acount. We design interfaces to be more intuitive and natural for usage by people in different age. Robotic devices are build in human-like form to pretend interaction with other human being.

2.2 Object detection and tracking

Effective tracking of different objects on 2D or 3D image, is extremely important issue in wider research area called computer vision [3]. In comparison to other methods of perception it provides significantly higher amount of information about surrounding environment. Therefore, it is frequently chosen as a main tool in mobile robot navigation [4]. For building 3D image, we use different types of sensors like stereo systems of cameras or Kinect-like devices, which are based on depth measurement technique based on structured light projection. The second one was chosen for the purpose of this work and it will be deeply discussed in Chapter 4. Both methods have their advantages and disadvantages, depending on functionality and environment, where system is going to be used [5]. First step in object tracking is to decide, which form of object representation is appropriate for our application. We can distinguish several, commonly used in literature [1]:

- point: located in the center of the object
- simple geometric model: like rectangular area
- complex geometric model: group of figures
- contour and silhouette: e.g. human body contour
- skeleton model: set of characteristic points connected by lines

In this work we focus on first and last from above mentioned models. Point tracking algorithm is effectively utilized in human following procedure, while gesture recognition part of the system based on relative position between joints in human skeleton.

2.3 Gesture-based interaction

Communication with gestures is an important topic in both language processing technology and computer sciences. Mapping of gestures on actions, which is going to be performed by the robot have to be done in intuitive manner. On the other side it have to be easy to detect by camera system, which suffer from noises coming from the environment. We can divide gesture recognition techniques into two types: static and dynamic gestures. First group can be detected by analysis of one frame from camera or to achieve better efficiency and noise cancellation, commonly we check sequence of consecutive images. Dynamic gestures are more frequent in natural language but require much more complex detection stateful algorithms. Nowadays, there exists many libraries, which make gesture detection easier [6]. In literature we can read about systems based on hand tracking [7] and application similar to our solution, where the full body gestures are analyzed [8] [9].

Chapter 3

Design

3.1 Task analysis

Our task was to design and implement human-robot interface, which allows communication via set of gestures and navigation of the robot by the change in human relative position. In this chapter we provide main requirments of the system, define gestures and present human following approach. After preliminary test with human tracking algorithm used in this work, we have to introduce the following assumptions:

- we assume wide space of operation without tall, human-like objects in the robot enviroment,
- because we are not equipped with the device, which allows independet horizontal movement of the video sensor and robot base, we assume no obstacles to avoid.

For safety reasons, in case of target lost or when communication with human is finnished via gesture, robot must immidiatly stop it's operation. In our robot interface we define two gestures for establishing and braking the connection and set of three gestures, which are expected during the navigation. In purpose to achive effective gesture recognition in every situation two simplifications have been done:

- the kinect sensor was mounted higher then in some constructions provided by the literature [10],
- used gestures were limited to static ones.

As will be presented in Chapter 6, this let us to avoid false recognition and target loses. We have been also decided to deliver feedback information via text-to-speech generator, which during interaction with human make our machine more human-like device. In Figure 3.1 we can see main structure of data processing system in the

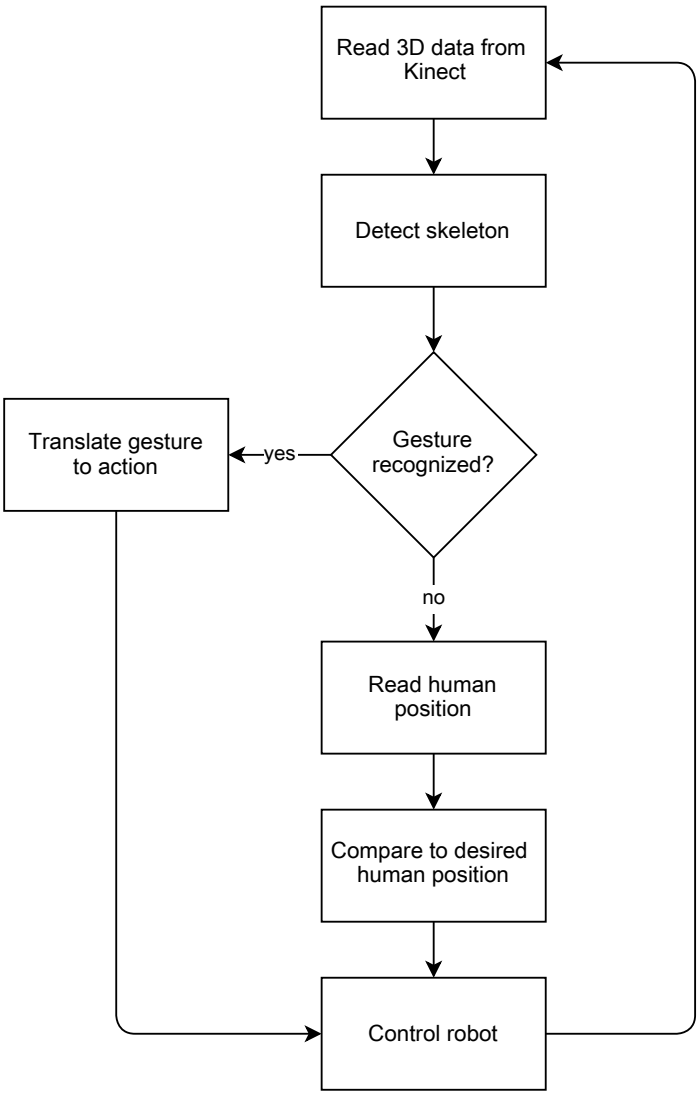
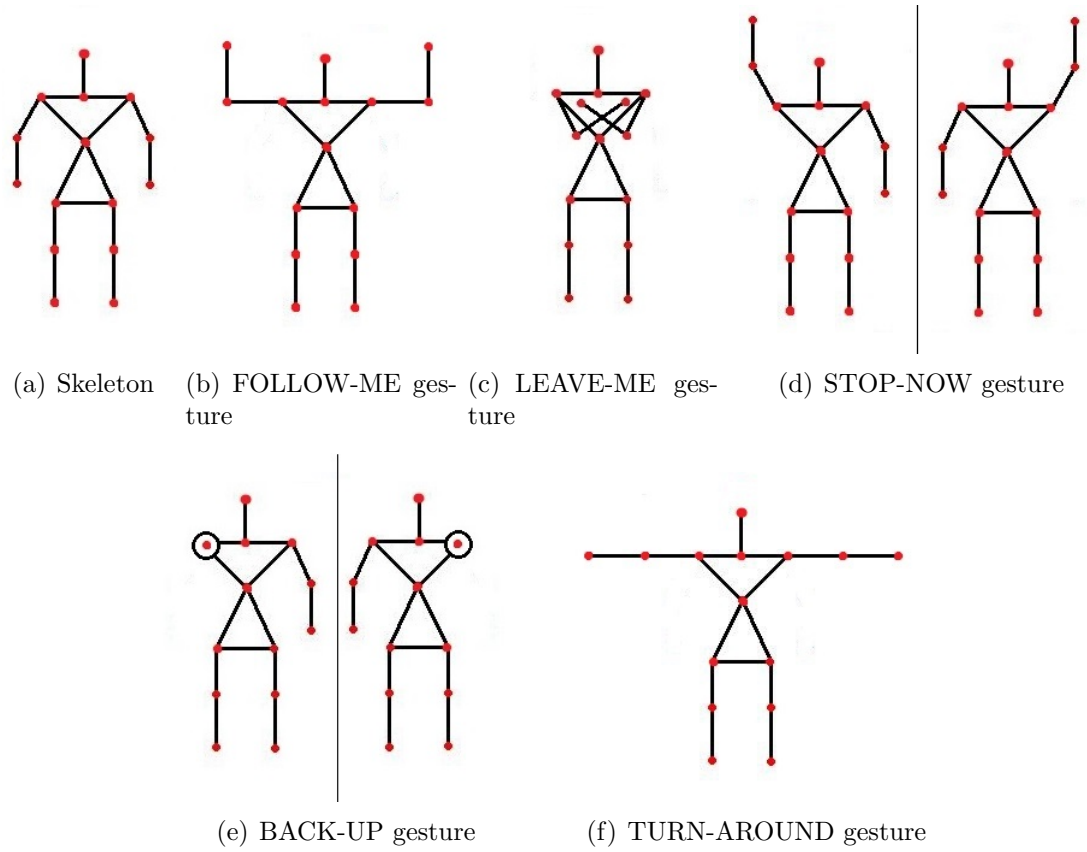


Figure 3.1: Data processing algorithm

designed software. At first a 3D image is captured and process by Kinect device. Next the human skeleton is extracted from point cloud and all the joints are detected. Relative position of joints is checked for gesture and in case of successful recognition an assigned action is performed by the robot. If no sign was presented by the user, human center of the mass position is estimated and compared to desired relative position to the machine. Finally, system use linear and angular velocity controllers for generation of robot base movement. The whole process is repeated in the loop with a frequency Kinect device operation, which value is equal to 30 fps.

3.2 Gestures

OpenNI library allows mapping human body, detected in 3D point cloud, onto skeleton model. By comparing joints(marked by red dots) position in following



(a) Skeleton (b) FOLLOW-ME gesture (c) LEAVE-ME gesture (d) STOP-NOW gesture

(e) BACK-UP gesture (f) TURN-AROUND gesture

Figure 3.2: Gestures

frames coming from Kinect, it is possible to read human predefined gestures. For the testing purposes of the system developed in this thesis, five static gestures were defined presented in Figures 3.2:

- FOLLOW-ME for establishing communication with the robot,
- LEAVE-ME for stoping the interaction,
- STOP-NOW for forcing robot to stop,
- BACK-UP for telling robot to move backward up to the certain boundary,
- TURN-AROUND for telling the robot that there is no way in this direction and it must turn around.

3.3 Human following

For smooth movement of the robot, human following procedure was realised by using P-controller presented in Figure 3.3. Due to the placement of the kinect sensor on the level close to human's body center of mass, we are able to make an approximation and assume world coordinatates equal to coordinates get from kinect point-cloud. Therefore, in our robot navigation part of the system, desired linear

and angular velocities for robot base are proportional to human displacement on horizontal plane. Y-axis controller is trying to keep human skeleton center of mass in proximity of vertical axis in camera, while x-axis controller regulate the distance between two object of our interaction.

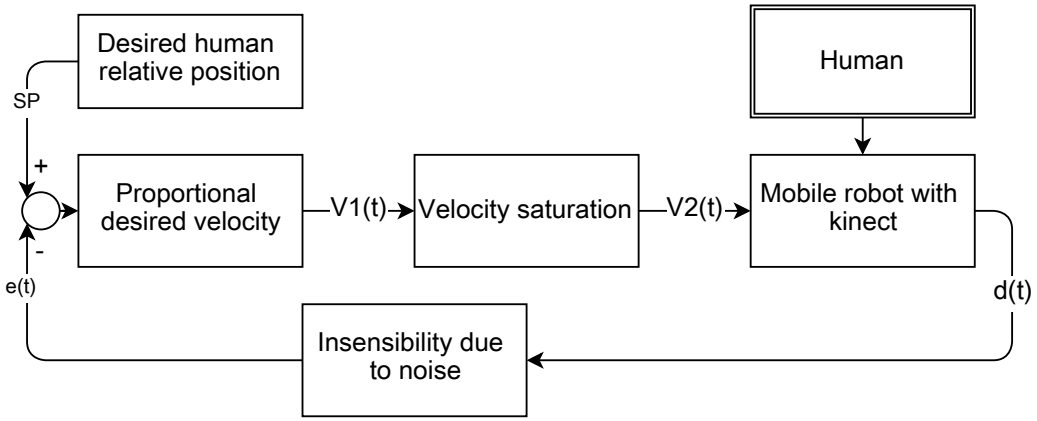


Figure 3.3: P-controller used in human following

Chapter 4

Tools

4.1 Hardware

4.1.1 Mobile Robot Elektron

Mobile Robot Elektron was designed in year 2007 at Warsaw University Of Technology by Institute of Control and Computation Engineering in cooperation with Institute of Control and Robotics [11, 12]. Since that time a few modifications have been made but the main body construction and wheel drive remains the same. The base of the robot can be seen in Figure 4.1. and sizes in that configuration are the following: 500 x 380 x 220mm (length, width, height). Elektron is based on six-wheel differential drive and was build as a mobile laboratory platform for testing robot control and navigation algorithms. It's modular construction allows using of many different sensor systems. Currently, robot is equipped with Microsoft Kinect and laser scanner SICK LMS 100. Odometry is realized by rotary encoders mounted on wheels on both sides of the machine. Intel Atom D525 (1.8 Ghz) dual core unit is used as an onboard central microcomputer and is placed between external overriding computer and microcontroller systems for sensors and actuators. Communication with external unit is done via Ethernet or USB port. Unit is also equipped with LCD display and a set of buttons, which are used for configuration and reading of work parameters.

4.1.2 Microsoft Kinect

Kinect is a motion sensing input device, which uses structured infrared light pattern to create depth map for every point in 2D camera image [13]. It was originally designed as a hands-free game controller for Microsoft Xbox 360 console but due to hardware data processing and open source libraries it has become very common tool for the scene analysis in robotics. It's relatively low price is achieved by mass

Figure 4.1: Mobile Robot Elektron

production for entertainment industry. In device body we can find RGB camera, IR (Infrared) projector and IR camera. All this elements can be seen on figure 4.2. The depth image has a 640x480 pixel resolution at 11 bits per pixel. The field of view is 57 degrees in horizontal and 43 degrees in vertical. The depth sensor realible range is 1,2-3,5 m. Kinect software makes possible to track skeleton image of one or two people moving in the front of the device, which allows developing gesture-driven applications. Therefore, it was chosen as a main sensor in this thesis.

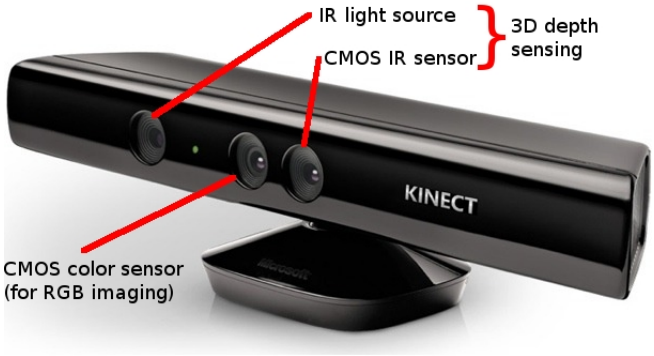


Figure 4.2: Microsoft Kinect

4.2 Software

4.2.1 ROS

ROS (Robot Operating System) is an open source framework created for development of robot applications [14]. It provides hardware abstraction for sensors and

actuators and supports distributed systems working on many machines of different types. ROS handles communication between processes, even running on not only the same machine. It is frequently chosen for mobile robot platforms with limited computation power, because it helps to use more powerful external machine for complex task like map creation or path planning. Moreover, it allows using many preprogrammed navigation algorithms and drivers nad utilize data read from Kinect sensor.

Part of the above mentioned system "tf", which stands for time frame is freqently use in our system. It lets the user keep track of multiple coordinate frames over time. Therefore, combined with openni framework it makes a perfect tool for joints tracking in human skeleton. Since, "tf" keeps track in time, we are able to analyze joint positions in sequence of frames, which is extremly useful for effective gesture recognition.

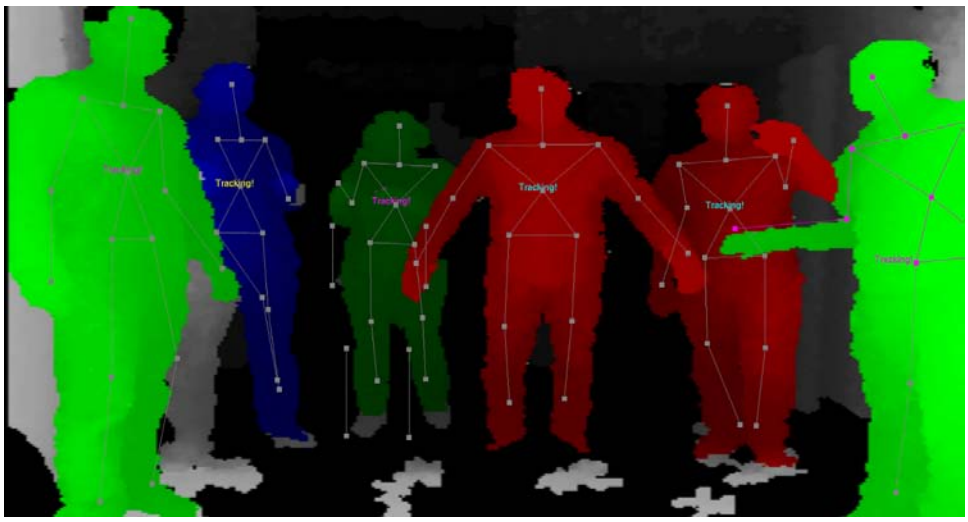


Figure 4.3: Multiple human skeletons tracking

Another component used in this work is sound-play packege, containt tool for text-to-speech translation and emition of human-like voice through speakers.

4.2.2 OpenNI

OpenNI (Open Natural Interaction) is a open-source framework which, allows integration of the Kinect sensor with ROS [6]. Combined with NITE middleware it allows to extract multiple skeletons from kinect point cloud,which can be seen in Figure 4.3. It is cross-platform and it was design for minimal CPU-load during full body and hand tracking. Although NITE contains functions for automatic dynamic and static gesture recognition, there are not directly available in ROS OpenNI interface. Instead in our system we rely on module called openni-tracker, from which we

are able to read position of 15 joints from multiple human bodies: head, neck, torso, left shoulder, left elbow, left hand, right shoulder, right elbow, right hand, left hip, left knee, left foot, right hip, right knee, right foot. The NITE skeleton with full set of 20 joints is presented in Figure 4.4.

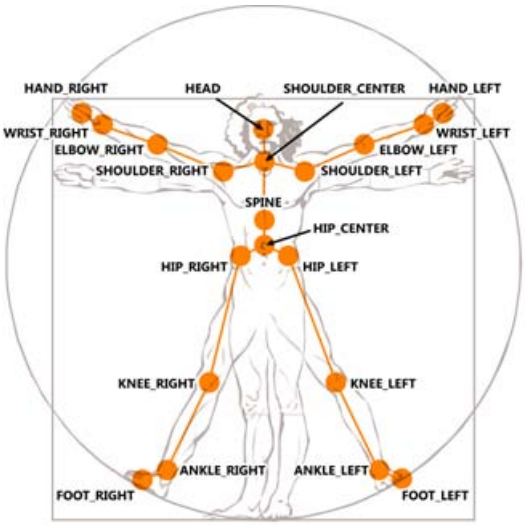


Figure 4.4: Skeleton with all the joints recognized by NITE middleware. Source: http://i.msdn.microsoft.com/dynimg/IC534688.png

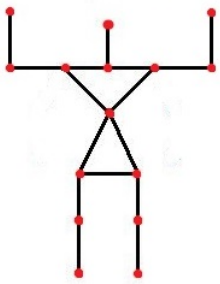


Figure 4.5: "Psi" pose

The tracker automatically register persons, who visible on the screen and after recognition of so called "psi pose", visible in Figure 4.5, perform skeleton calibration procedure. Afterwards by joints position accessible on tf topic with number according to order of detection. Finally, it has to be mentioned that position coordinates are calculated according to the center of RGB camera in kinect device.

Chapter 5

Implementation

5.1 Construction improvements

This section describes construction changes, visible on figure 5.1(a), which were made in mobile robotic platform described in Chapter 4. The kinect sensor used for human tracking was placed 104 cm above the robot base. After changes camera is located 126 cm over the ground level and provide better usability of arm gestures performed by human. Moreover, camera center is located closer to the level of human center of mass which is represented by "torso" joint in skeleton model. This fact reduce amount of calculation in human following procedure, assuming orientation of kinect coordinates system equal to global one. Also it makes the robot appearance more human-like, which is desired feature in social robot interaction.

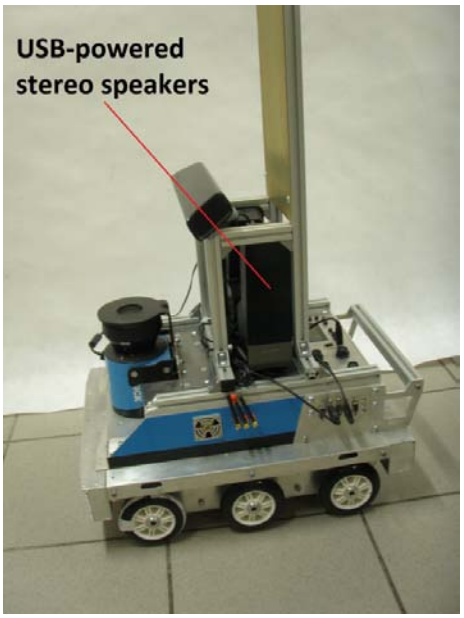
5.2 Software structure

In Figure 5.2 we present structure of the whole system,which was implemented in ROS framework. On the bottom, we can see representation of hardware layer, which is mobile robot with kinect sensor and USB-powered stereo speakers. Middle part consist of ROS nodes, which are responsible for kinect data processing, moving robot base and speech generation. Top layer, implemented by single node, called elektron-follower, is main part of the system and our contribution to this work. It is responsible for execution of the following procedures:

- analysis of multiple human skeletons delivered by openni-tracker,
- sending velocity commands to elektron-base node,
- transferring text, describing current status of the program, to speech generator.

Internal part of that module will be discussed later in next sections of current chapter. Since, ROS run each of the nodes as a different process, we can take an ad-

(a) Construction changes made in robot for more optimal use of Kinect sensor



(b) Speakers used in voice feedback

Figure 5.1: Mobile Robot Elektron with upgrades

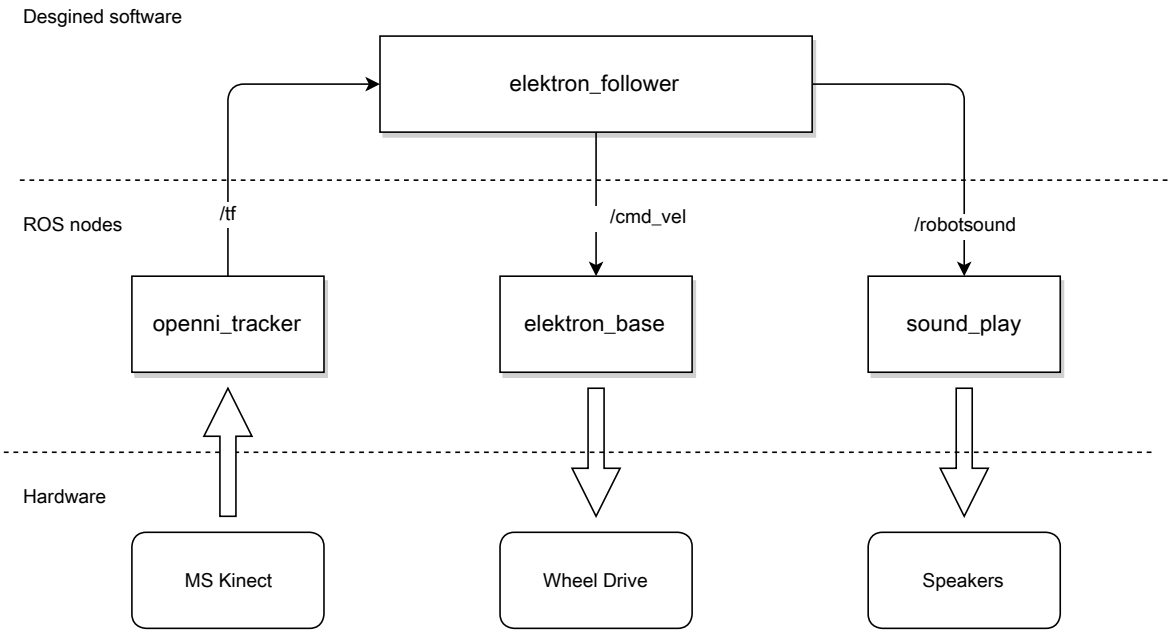


Figure 5.2: Nodes in ROS

vantage from multi core computing unit architecture and obtain better performance of the operation then on single-core machine.

5.3 Main algorihtm

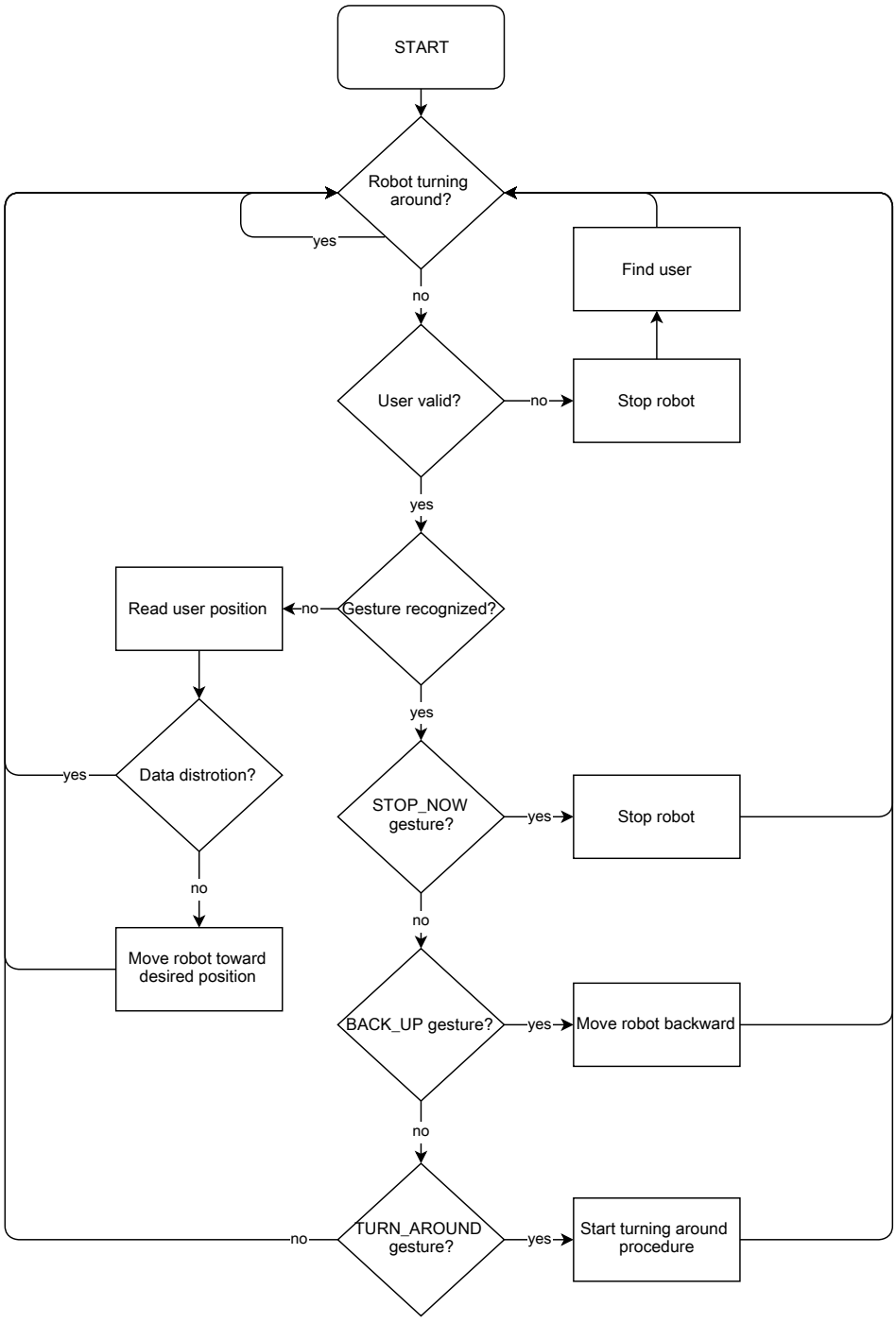


Figure 5.3: Main algorithm

The elektron-follower node, mentioned in previous section, is executed in main loop operating with frequency of 10Hz. Steps in every single execution can be seen in Figure 5.3 and consist of function calls from three libraries, which are detaily explained in subsequent sections of this chapter:

- tracker.h for detecting the user amoung people on the scene,

- gestures.h for gesture recognition,
- robotcontrol.h for controlling robot movement.

Going through the algorithm we can see that gestures are checked in sequence,which assign to them certain priorities. For safety reasons the most important among action gestures is STOP-NOW, next BACK-UP and TURN-AROUND. Moreover, in each loop we are check validity of the target and in case of lost, we stopping the robot and calling user search procedure. In human following procedure we always check for data distortion, which may be caused by tall human-like objects. Frames where data difference is greater then certain threshold value are not taken into account. Below we provide code implementation of the main loop in algorithm followed by short description of each fragment:

5.4 User detection and tracking

In Figure 5.4(a) we present how the user searching procedure is performed. Openni-tracker register person on the scene according to order of detection. It also trying to remember the user when contact lost, but this algorithm does not work perfectly. We cannot rely on user number when looking for a target, instead we going through all accessible obcject and checking for innitial FOLLOW-ME gesture.

Every 10ms the target validity is checked by algorithm visible in Figure 5.4(b). If the human skeleton is lost, it's data still exist in tf but are not changing at all. In case of lack of change the target person number is rejected and we start user searching procedure once again. In each call me are also check for LEAVE-ME gesture, asking if the user want to close the interaction.

5.5 Gesture recognition

For effective gesture recognition we implement set of functions in gesture.h library. To avoid false detection each gesture is checked in two frame sequence, in current frame and in 5. frames from the past. If both frames gives positive result then gesture is confirmed and corresponding action is performed. Below each Figure we provide short description of each gesture and conditions necessary to positive recognition. Functions: x(joint), y(joint), z(joint) returns x, y and z coordinates of joints in camera 3D image.

Gesture FOLLOW-ME visible in Figure 5.5(a) is used to establish connection between user and the robot. It was designed to be similar to aforementioned "psi" pose in purpose to force user to perform body calibration procedure in openni-tracker.

(a) User search algorithm (b) User valid algorithm

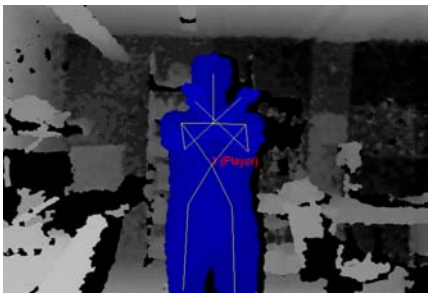
Figure 5.4: Algorithms used in interaction with human

Conditions:

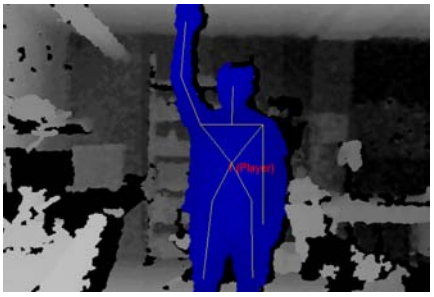
- (z(left hand) >z(head)) and
- (z(right hand) >z(head))

Figure 5.5(b) presents LEAVE-ME gesture used for closing connection with the robot. After positive gesture recognition, target become invalid and user search function is activeted.

(a) FOLLOW-ME gesture

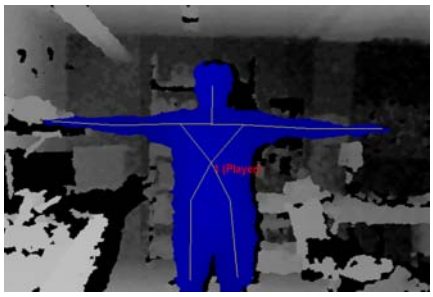


(b) LEAVE-ME gesture



(c) STOP-NOW gesture

(d) BACK-UP gesture



(e) TURN-AROUND gesture

Figure 5.5: Gestures

Conditions:

- (z(left hand) >z(left elbow)) and
- (z(right hand) >z(left elbow)) and
- (z(left hand) >z(right elbow)) and
- (z(right hand) >z(right elbow)) and
- (y(left hand) <y(right hand)) and
- (y(right elbow) <y(left elbow))

STOP-NOW gesture from Figure 5.5(c) provides safety feature to stop robot movement in any time during interaction with the robot. After positive recognition wheel drive is stopped and keep, as long as, user hold one of the hand above his head. Therefore, human performing this gesture can walk around and robot do not follow him.

Conditions:

- (z(left hand) >z(head)) and !(z(right hand) >z(head))) or
- (!(z(left hand) >z(head)) and (z(right hand) >z(head)))

BACK-UP gesture visible in Figure 5.5(d) is the only gesture, which used depth sensor in recognition. It force the robot to move backward until gesture is no longer present or distance from the human is no more then 4 meters. After reaching the boundary wheel drive is stopped.

Calculations:

- (MINIMUM ARM LENGTH):=0.42
- lHSDiff:= x(torso) - x(left hand)
- rHSDiff:= x(torso) - x(right hand)

Conditions:

- (lHSDiff >(MINIMUM ARM LENGTH)) or
- (rHSDiff >(MINIMUM ARM LENGTH))

Finally, in Figure 5.5(e) we present TURN-AROUND gesture used to rotate robot base by 180 degrees. It can be useful in narrow areas when there is no way to navigate the robot by human displacement. During turning procedure gesture recognition system is deactivated and user became invalid. Afterwards user search fictions is used to renew the connection.

Calculations:

- (TURN GESTURE ERROR):=0.1
- upperBound = (z(left shoulder) + z(right shoulder))/2 + (TURN GESTURE ERROR)
- lowerBound = (z(left shoulder) + z(right shoulder))/2 - (TURN GESTURE ERROR)

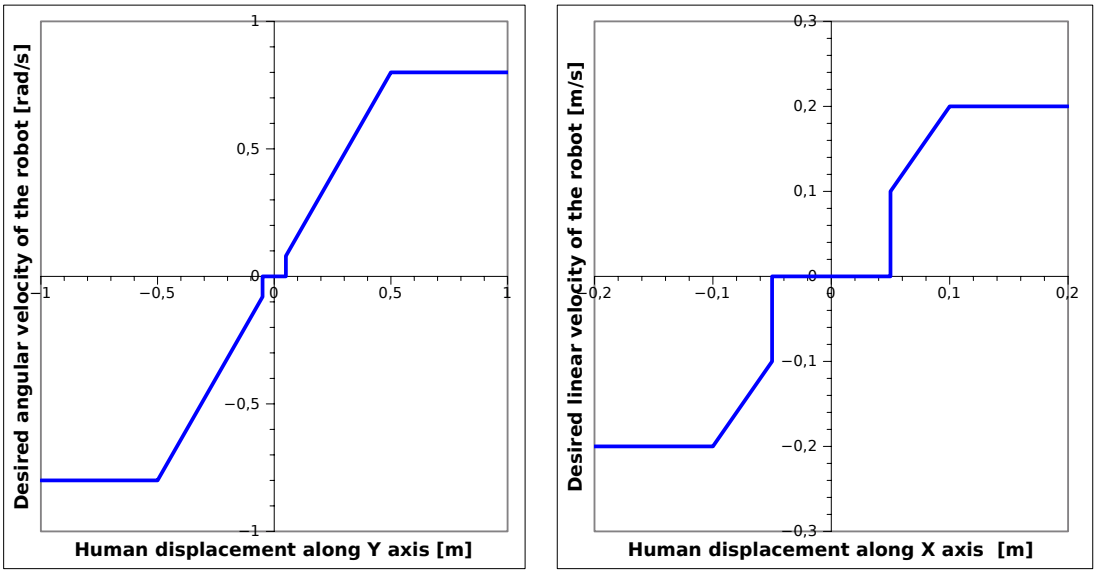
Conditions:

- (y(right hand) <y(right elbow)) and
- (y(left hand) >y(left elbow)) and
- (z(left hand) >lowerBound)) and
- (z(left elbow) >lowerBound)) and
- (z(right hand) >lowerBound)) and
- (z(right elbow) >lowerBound)) and
- (z(left shoulder) >lowerBound)) and

- (z(right shoulder) >lowerBound)) and
- (z(left hand) <upperBound)) and
- (z(left elbow) <upperBound)) and
- (z(right hand) <upperBound)) and
- (z(right elbow) <upperBound)) and
- (z(left shoulder) <upperBound)) and
- (z(right shoulder) <upperBound))

5.6 Robot navigation

Navigation of the robotic platform is implemented in robotcontrol.h library, which contains set of functions for controlling linear and angular velocity of the base. Due to lack of self-orientation of the robot in environment, turning around procedure is performed by calculating the time, which robot needs to perform an 180 degrees turn with certain angular velocity. Because of the wheel slip in real environment an extra coefficient of correction of that time have to be introduced and tuned.



(a) Angular velocity

(b) Linear velocity

Figure 5.6: Velocity curves

On figures 5.6(a) and 5.6(b) we present velocity curves used for linear and angular velocity control in human following procedure. The saturation comes from the limitation of the robot based velocities, while close to zero insensibility was introduced to protect from system instability caused by distortion in kinect data.

In case when the distance between the robot and human falls to 1 meter or rises above 3m, the voice feedback is used to ask user to slow down.

5.7 Voice feedback

On figure 5.1(b) we can seen speakers, which where installed on robot platform and are used for playing the voice messages describing current status of the system. For speech generation we use module from ROS framework called sound-play. The function from robotcontrol.h library translates current status of program to text, which is next send to text-to-speech generator. Below is presented a fragment of the code, which is used in translation of the program status into text messages. Afterwards in each case of switch statement speech is generated from each phrase and played through the speakers by ROS sound-play module. Line 2 and 25 protects module from playing the same sound twice and make it sensitive to only the change of status.

```
1  void RobotControl::voiceFeedback(int soundId){
2      if( soundId != lastSound ){
3          switch(soundId){
4              case SOUND_SLOW: sc.say("Please slow
                   down...");ROS_INFO("SLOW DOWN!!!"); break;
5              case SOUND_INTRO: sc.say("Hi. My name is Elektron
                   and i am social interactive mobile robot.... I
                   am waiting for your commands."); break;
6              case SOUND_SEARCHING: sc.say("Searching for
                   user."); break;
7              case SOUND_TURN_AROUND: sc.say("Gesture recognized.
                   TURN AROUND.... Turning around please wait.");
                   break;
8              case SOUND_STOP: sc.say("Gesture recognized.
                   STOP...."); break;
9              case SOUND_BACK_UP: sc.say("Gesture recognized.
                   BACK UP...."); break;
10             case SOUND_LEAVE_ME: sc.say("Gesture recognized.
                   LEAVE ME.... Searching for new user."); break;
11             case SOUND_FOLLOWING: sc.say("Following"); break;
12             case SOUND_BYE: sc.say("Goodbye."); break;
13             case SOUND_LOST: sc.say("User lost.... Searching
                   for new user."); break;
14             case SOUND_PERSON_1: sc.say("Person 1 set as a
                   user."); break;
15             case SOUND_PERSON_2: sc.say("Person 2 set as a
                   user."); break;
16             case SOUND_PERSON_3: sc.say("Person 3 set as a
                   user."); break;
```

```
17              case SOUND_PERSON_4: sc.say("Person 4 set as a
                    user."); break;
18              case SOUND_PERSON_5: sc.say("Person 5 set as a
                    user."); break;
19              case SOUND_PERSON_6: sc.say("Person 6 set as a
                    user."); break;
20              case SOUND_PERSON_7: sc.say("Person 7 set as a
                    user."); break;
21              case SOUND_PERSON_8: sc.say("Person 8 set as a
                    user."); break;
22              case SOUND_PERSON_9: sc.say("Person 9 set as a
                    user."); break;
23              case SOUND_PERSON_10: sc.say("Person 10 set as a
                    user."); break;
24          }
25          lastSound=soundId;
26      }
27  }
```

Chapter 6

Testing

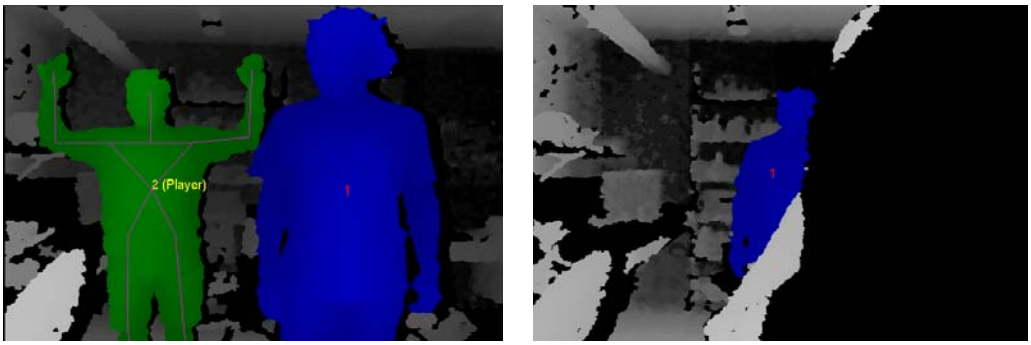
6.1 Evaluation of target tracking and gesture-based interaction

The designed gesture recognition system was tested separately under static conditions with robot engines switched off. The test assumes 10 attempts to perform each of gestures from the positions along x-axis: 1,2,3 and 4 meters from the robot. The results of the experiment are presented in table below. As we can observe, we obtained 100 percent success rate in distance 2 meter and greater. Even in 4 meter scope when according to Kinect specification data are not reliable, system achieved effective human-robot interaction. Only problems occurs in lower bound for Kinect range where, field of view is to narrow to capture the whole human skeleton.

Gesture:	1m	2m	3m	4m
FOLLOW-ME	0/10	10/10	10/10	10/10
LEAVE-ME	8/10	10/10	10/10	10/10
STOP-NOW	1/10	10/10	10/10	10/10
BACK-UP	0/10	10/10	10/10	10/10
TURN-AROUND	0/10	10/10	10/10	10/10

Next we tested user search algorithm with multiple targets on the scene and as we can seen in Figure 6.1(a), even if the person on the right hand side appears as a first user, the second person on right hand side was calibrated and set as the target for later intercation with the robot.

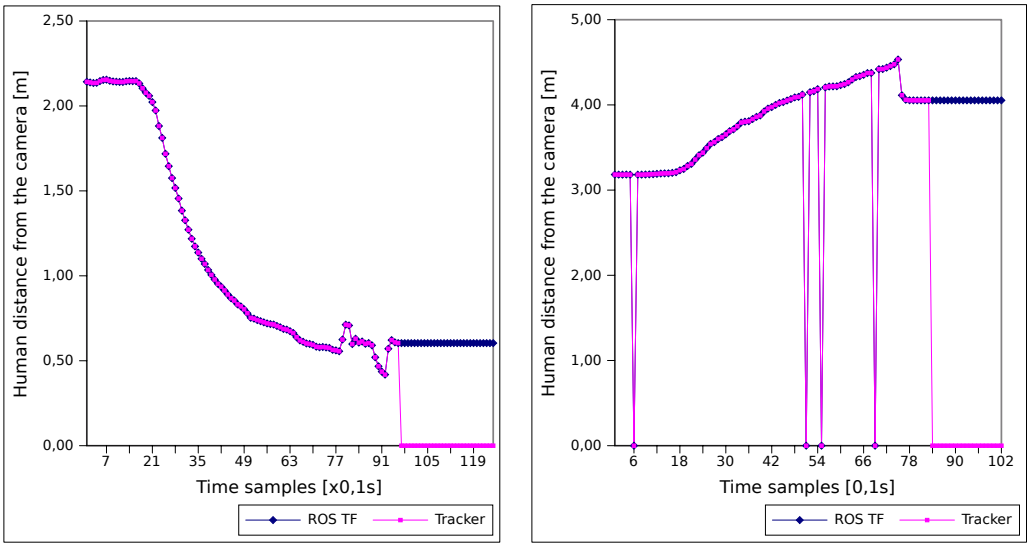
In the third test we check the system behavior, when human walking out of scope of the depth sensor in Kinect. Figure 6.2(a) present human walking towards the robot and Figure 6.2(b) we tracking movement in another direction. As we can observe, even the human is out of range and lost by skeleton tracking algorithm, the static frames still exist in ROS TF. For that reason in our tracker we check the



(a) Two users (b) Human obstacle

Figure 6.1: Multiple objects on the scene

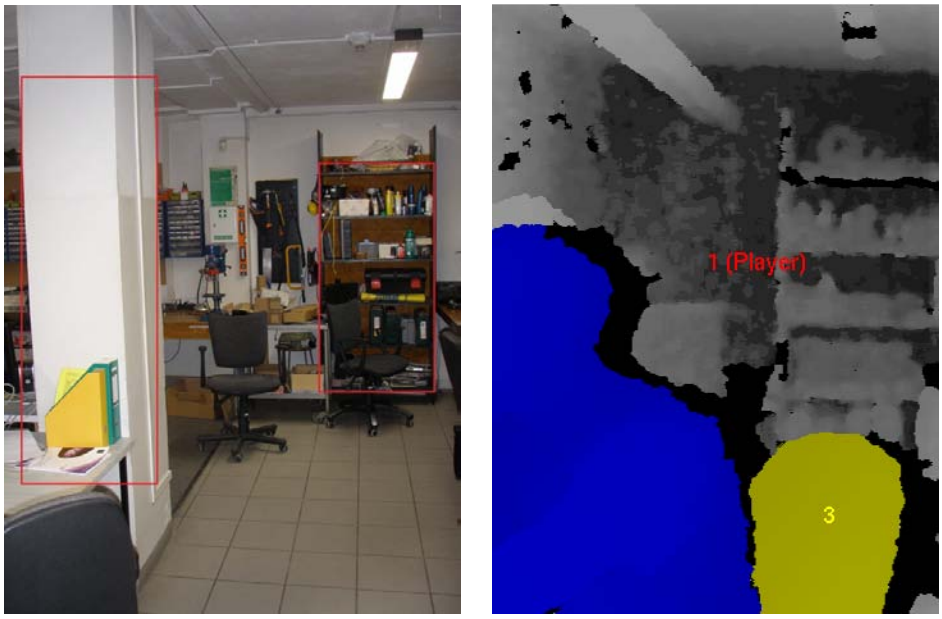
difference between the consecutive frames delivered by openni-tracker and in case of no change, we mark user as invalid.



(a) Human walking towards the Kinect (b) Human walking away from Kinect

Figure 6.2: Results of testing tracking range of the Kinect.

In Figure 6.3(a) we marked by red color, human-like object causing distortion in Kinect data that can be observed in Figure 6.2(b). Algorithm of noise cancellation was tested in environment presented above and it provides smooth motion of the robot with only few target lost situations. Figure 6.3(b) presents example of false recognition, where a chair is detected as human body and assign to user number three.



(a) Objects responsible for distortion in human tracking

(b) False recognition of human

Figure 6.3: Examples of objects causing distortions in human tracking

6.2 Robot navigation in real environment

The procedure of turning around,visible in Figure 6.4, which was tested in simulation provides expected inaccuracy in real environment. Attempt to turning platform by 180 degrees, result in about 33 degrees turn in real environment due to wheel slip. In order to tune that action, 10 tests were performed and the average value was calculated. We obtain result: 33 degrees and set the time correction coefficient to 5,5.

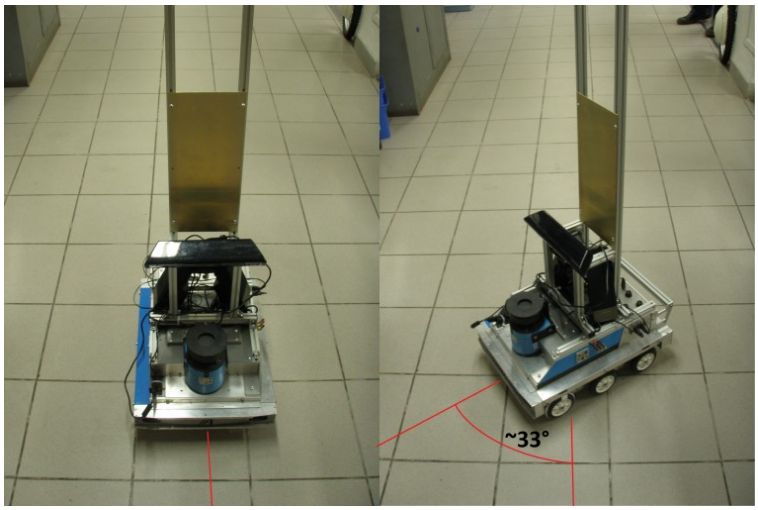


Figure 6.4: Turning in real eviroment

We also test a safety of our system placing a human obstacle in the front of robot during human following procedure. As it can be seen in Figure 6.1(b), because one of the sensor used for depth measurement is covered up, the data about user skeleton are no longer delivered to the system. As a result target will be marked as not valid and robot will be stopped.

In last test, we checked our assumption about insensitivity in human following procedure. In our system, we do not follow the target if the change in position is less then 0,05 meter. In Figure 6.5, we measure the error in the position of human center of the mass, which is represented by "torso" joint in skeleton model. As we can see the error in static human position is 10 times smaller then our bounds implemented in the system.

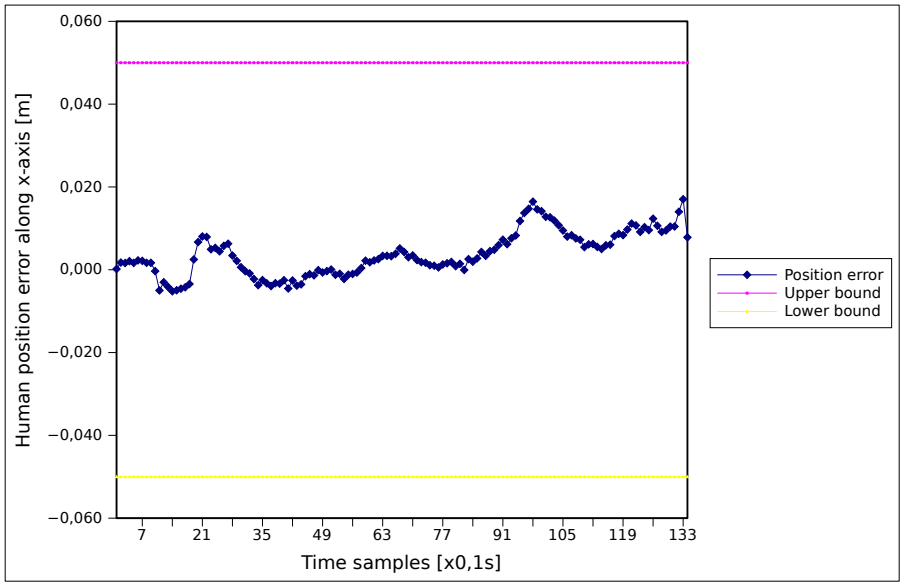


Figure 6.5: Error in tracking human center of the mass.

Chapter 7

Summary and conclusions

In our work, we designed and implemented vision based interface with successful gesture recognition mechanism. It provides full functionality for robot navigation and it realize safety features desired in human-robot interaction. The voice feedback and human-like appearance make a basis for future development of social interactive robot platform. Due to modular design of the software, it can be also installed on platform, which allows faster movement of the robot base, because current velocity limits are not comfortable for natural human walk. In order to create solution more suitable to use in unstructured environment, kinect supposed to be mounted on pan-tilt device, which will allow for independent human tracking and robot navigating. Afterwards, we can implement algorithms for obstacle avoidance and path planning. It is also suggested to add some dynamic gestures and use of microphone array in kinect sensor for interaction on voice commands.

Bibliography

[1] M. A. Goodrich and A. C. Schultz. Human–robot interaction: A survey. *Foundations and Trends in Human–Computer Interaction*, 1(3):203–275, 2007.

[2] M. Li and W. Pan. Gesture-based robot's long-range navigation. *The 7th International Conference on Computer Science and Education*, 2012.

[3] L. Spinello and K. O. Arras. People detection in RGB-D data. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[4] M. Stefańczyk. Wykorzystanie informacji z kamery 3D do nawigacji robota mobilnego(in polish). Master's thesis, Warsaw University of Technology, 2011.

[5] W. Jia, W. Yi, J. Saniie, and E. Oruklu. 3D image reconstruction and human body tracking using stereo vision and kinect technology. *IEEE*, 2012.

[6] OpenNI: The standard framework for 3D sensing. [Online: http://www.openni.org/], 2013.

[7] M. Van den Bergh, D. Carton, R. De Nijs, N. Mitsou, C. Landsiedel, K. Kuehnlenz, D. Wollherr, L. Van Gool, and M. Buss. Real-time 3D hand gesture interaction with a robot for understanding directions from humans. *20th IEEE International Symposium on Robot and Human Interactive Communication*, 2011.

[8] H. Yang, A. Park, and S. Lee. Gesture spotting and recognition for human–robot interaction. *IEEE transactions on robotics*, 23(2):256–270, 2007.

[9] L. Cheng, Q. Sun, H. Su, Y. Cong, and S. Zhao. Design and implementation of human-robot interactive demonstration system based on kinect. *IEEE*, 2012.

[10] T. Bonanni. Person-tracking and gesture-driven interaction with a mobile robot using the kinect sensor. Master's thesis, Sapienza University of Rome, 2011.

[11] Mobile Robot Elektron documentation. [Online: http://robotyka.ia.pw.edu.pl/twiki/bin/view/Bionik/Elektron/WebHome], 2013.

[12] W. Szynkiewicz, R. Chojecki, A. Rydzewski, M. Majchrowski, and P. Trojanek. Modułowy robot mobilny elektron(in polish). Technical report, Warsaw University of Technology, 2007.

[13] Kinect Wikipedia. [Online: http://en.wikipedia.org/wiki/Kinect], 2013.

[14] ROS: Robot Operating System. [Online: http://www.ros.org], 2013.