Aalto University
School of Electrical Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

Tesfamichael Agidie Getahun

Ceilbot Development and Integration

School of Electrical Engineering

Department of Electrical Engineering and Automation

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

18.08.2014

Instructor: Tomi Ylikorpi Lic.Sc.(Tech.)
Aalto University
School of Electrical Engineering

Supervisors: Professor Emeritus Aarne Halme
Aalto University
School of Electrical Engineering

Professor Thomas Gustafsson
Luleå University of Technology

Aalto University

School of Electrical Engineering Abstract of the Master's Thesis

Author:	Tesfamichael Agidie Getahun		
Title of the thesis:	Ceilbot Development and Integration		
Date:	August 18, 2014	**Number of pages:**	60
Department:	Electrical Engineering and Automation		
Program:	Master's Degree Program in Space Science and Technology		
Professorship:	Automation Technology (AS-84)		
Supervisors:	Professor Emeritus Aarne Halme (Aalto)		
	Professor Thomas Gustafsson (LTU)		
Instructor:	Tomi Ylikorpi (Aalto)		

The mobile robots that are present today are struggling to deal with challenges related to localization, power supply, mobility in the real world with all sorts of obstacles and other issues. At the same time, the demand for service robots for domestic applications has been growing and predictions show that the demand will continue to grow in the future. To meet the demands and to fulfill the expectations, those challenges need to be addressed. This thesis presents the development of a ceiling mounted robot known as Ceilbot. It is a type of mobile service robot except that it works on a track attached to the ceiling. This implies that the robot operates in a structured environment with continuous power supply simplifying some of the issues mentioned above. The development of the Ceilbot includes a simplified DC motor controller development, object recognition development and an easy-to-use graphical user interface design. The developed motor controller provides flexibility for the user to change the control parameters and produces deterministic output with high repeatability when compared to a regular proportional and integral (PI) controller. The designed user interface simplifies the interaction between the user and the Ceilbot by allowing the user to send commands to the Ceilbot and displaying some status parameters for monitoring. In order to have a complete robot system for demonstration purposes, a simple manipulator using two servomotors is also developed.

Keywords: Ceilbot, Ceiling mounted robot, color based object recognition, position determination using Kinect camera/sensor, GUI development

Acknowledgments

First, I would like to thank my instructor Tomi Ylikorpi for his continuous advices and recommendations to make the thesis content organized and complete. I also thank Professor Emeritus Aarne Halme for the comments to clarify the report.

Thank you to Laura Mendoza from the language center for the invaluable comments to make the first three sections of this report clear and readable.

I am also grateful to Jaime Hernandez, SpaceMaster round 7, for his brief introductions to the Ceilbot hardware and the necessary tools to be used for the Ceilbot development. Thank you to the SpaceMasters round 8 at Aalto University - Franz Steinmetz, Michael Strohmeier, Nemanja Jovanovic and Christina Sherly - for providing me answers, suggestions and clues for all my annoying questions - that helped me considerably.

Finally, a special thank you to the European Commission, Education, Audiovisual and Culture Executive Agency for granting me the scholarship to attend the joint European Master in Space Science and Technology - SpaceMaster (round 8).

Contents

List of Figures

List of Tables

1. Introduction

1.1. Brief overview of robotics

The first robot was installed in 1961 by the General Motors in USA. Since then robot uses in industry (industrial robot) have developed enormously. As a result, in today's modern industries manufacturing cannot be imagined without the support of robots [1]. Industrial robots are, however, fixed on their shoulders and cannot move from place to place to perform other tasks that require mobility such as transporting goods from one station to another. They only work a repetitive task in a very well controlled environment and with close assistance of the people [2]. Therefore, other types of robots are obviously needed to accomplish tasks that require mobility. These types of robots are commonly referred to as service robots. The main difference between the industrial and service robots lies on mobility capability either in a well-structured static environment or in unstructured and dynamic environment. Either case, the robot needs to have some level of intelligence to navigate in the environment and accomplish the required tasks [1], [2], [3].

Information about the environment in which the robot is expected to work is collected by a variety of sensors which serve as the eyes and ears of the robot. The sensor information is then processed and a control technique/algorithm decides how the robot should react to that given environment. The decision making process may not be entirely carried out by the robot itself. There is always some level of interference from humans due to the fact that the robot cannot perform as desired by its own. These group of robots are categorized as semi-autonomous robots. However, the advances in artificial intelligence, sophisticated sensor technology and high performance computing platforms are paving a way to the improvement of the intelligence level of robots and hence its autonomy [3]. For example, in the 1960s and 1970s an autonomous vehicle that can drive at a speed of one meter every 10 to 15 minutes was created by researchers at Stanford, Carnegie Mellon Universities and Jet Propulsion Lab (JPL). In late 1980s, another autonomous vehicle that could follow a road at speed of 96 km/h was created [2]. An example of fully autonomous and popular service robot is the iRobot which is capable of cleaning a room without interference from humans [4].

The potential application areas of service robots are very diverse and the demand in the market has been showing a significant growth. According to the World Robotics statistics report released in 2013, the demand for robots in general will increase by 6% on average [5]. The majority of the percentage is taken by industrial robots. Nevertheless, the forecast for the next two years shows that service robots for domestic applications will grow considerably as shown in Figure 1.1.

The desire to use robots for domestic applications has started about 1000 years ago [1]. However, the development of such kind of robots has been very slow and only few robots are deployed and are in use today. Those robots already in use are usually specialized to perform a single task in one kind of environment [2], [6]. The crude sensory data and lack of cognitive abilities are the main reasons that affected the flexibility of this kind of robots. The development cost is also another factor for the sluggish progress of the robots for some domestic applications such as healthcare [6],[7].

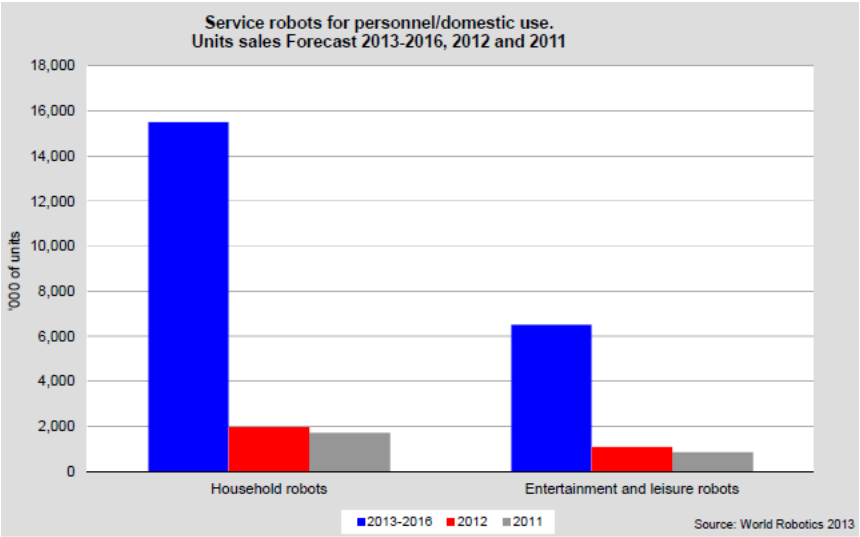


Figure 1.1 Forecast of Personal/Domestic use robots by World Robotics

1.2. Ceiling mounted robots

One of the major challenges that service robots in general encounter is obstacle detection and avoidance. The obstacles faced by the robot can be of static or dynamic in nature. If, for example, a wheeled robot is designed for domestic applications such as working as an assistant for the disabled, the robot has to avoid dynamic obstacles like humans and static

obstacles like tables and chairs. However, if there are staircases in the house, then the robot will fail to perform its task. Although there are some research-robots that are able to climb staircases, they are mostly preprogrammed to do so for that particular situation.

The other issue for mobile robots is power supply. The primary power source for mobile robots is batteries and these batteries get exhausted in some time interval. To return the robot to operation, the batteries need to be replaced or recharged. Thus, there is the issue of non-continuous power supply for the robot.

One possible approach to cope with the challenges related to obstacles and power supply is using a special kind of robot that operates under the ceiling instead of on the ground [8]. This approach, of course, applies to robots that are required to work in indoor environments. This kind of service robot is designed to move on a track/rail attached to the ceiling to execute its tasks. The space under the ceiling is often free of any obstacles and the robot can be connected to the power outlets for a continuous power supply. Therefore, transferring the robot from the ground to the ceiling simply eliminates the critical problems that mobile robots often encounter when they are on the ground. The added advantage of ceiling mounted robot is that the vision system installed on the robot provides a better view of the environment in which it is operating when compared to the vision system of the mobile robots on the ground.

In this report, a ceiling mounted robot, which will be referred hereafter as Ceilbot, is presented and its organization is as follows. The practical application of ceiling mounted robots that are in use today by different industries are reviewed in section 2. Although these robots are dedicated to perform a (single) specific task, they clearly demonstrate how ceiling mounted robots can be adapted for domestic applications as well as other industrial applications by extending their capabilities such as multitasking.

In terms of the construction of the Ceilbot, the hardware requirements for the ceiling mounted robot, in general, is mostly the same as any other mobile robot except the locomotion system. The issues of the locomotion system and the possible solutions proposed by different researchers are briefly presented in section 3. The building blocks of the Ceilbot system, the role of each block in the system, the algorithms used for integrating the blocks to form the robot system and the implementation methods and details are presented in section 4. Results of the implementation and their analysis is covered in section 5. Finally, the conclusion is presented in section 6.

2. Applications of the Ceilbot

The potential of Ceilbot for indoor applications is very high, including both domestic and industrial applications. One of the key advantages of using these kinds of robots is that it makes use of the space under the ceiling. In addition, it eliminates the complexity of obstacle detection and avoidance problems in the implementation as there are not many obstacles on the way of the robot. In this section, some of the practical applications of ceiling-mounted mobile robots that are currently in use are presented and potential application areas are also suggested.

2.1. Commercialized applications

2.1.1. Surveillance System (Security)

The common way of monitoring activities in big stores is by using large number of security cameras installed at different locations. The problems associated with the use of large number of cameras are huge cost and difficulty of managing each camera from the operator room. For example, the company Boots UK has a store building of size 413,000 square feet and a large number of employees in it. This company uses a track-based camera system, which is equivalent to ceiling mounted robot, for the surveillance. The ceiling mounted robot is developed by a company called Sentry [9], [10] (see Figure 2.1). The loss prevention division of the Boots company estimated that they would have needed 1,600 independent cameras to cover the area if they had chosen to use the traditional method.

The camera tilt, pan and zoom operations can be controlled remotely by the operator or it can be programmed to operate autonomously. The added use of this system is to monitor people whether they are following the strict onsite safety rules or not, especially those people working near dangerous machineries. A similar robotic surveillance system is also developed by the Revolutionary Robotics Company based in Slovenia [11] as shown in Figure 2.2. Their robot is called Advanced IP CamTrack (AIPT) and it is IP-based system which can be accessed through the internet from anywhere.

2.1.2. Studio applications

In this application, the robot is being used to carry a professional video camera and the movement of the robot is usually controlled by the operator. Autonomous operation is also possible. Camera tilt, pan and zoom operations are executed by the manipulator which is of a single arm suspended from the motorized trolley. One of the popular designers of this robotic system is *Telemetrics Inc*. [12].

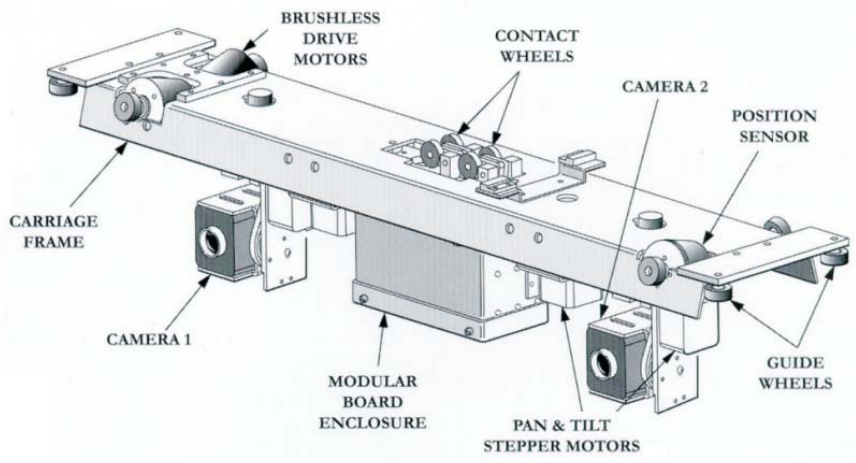


Figure 2.1 Sentry's smart track camera carriage [9].

Figure 2.2 Advanced IP CamTrack (AIPT) developed by the *Revolutionary Robotics* [11].

2.1.3. Health care in Hospitals

To increase the mobility of patients by providing dynamic body weight support, a robotic trolley is being used in hospitals since recently. The robot, known as Vector Gait & Safety

System, is used for the first time at the University of Maryland Rehabilitation and Orthopaedic Institute (see Figure2.3). Using such kind of system is said to have improved the rehabilitation process of patients with spinal cord injury, stroke and other problems that affect balance. The system avoids the psychological effects of falling [13]. The system also monitors the patient's movements and walked distance. A similar product, called *ZeroG,* is also designed by *ARETECH* Company with additional features such as recording performance data for each training session, thus allowing the therapist to easily keep track of the patient's progress [14].

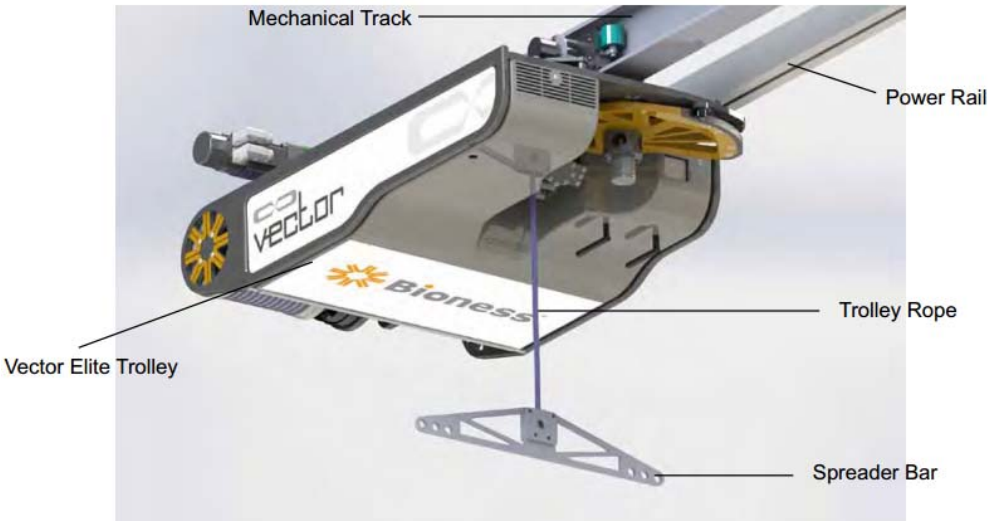


Figure 2.3 Vector Gait & Safety System robot by *Bioness* [13].

2.2. Other potential applications

The requirement analysis studied in [15] shows that ceiling-mounted mobile robots have the potential to be used in many applications. However, the prototype robot demonstrated clearly shows that a considerable amount of improvement is needed to use the robot practically. In this section, possible applications of the Ceilbot are briefly presented.

2.2.1. Domestic Applications

Cleaning of windows, organizing children's rooms, locating and recovering misplaced devices are some of common activities in everybody's home. Some of these activities are

repetitive, tedious and demanding tasks. These kinds of tasks can be effectively handled by the Ceilbot which it can perform the tasks without interfering with the personal space of the owner. This can be achieved by scheduling the tasks.

Another possibility of using the Ceilbot is as a personal care assistant. The assistance could be for everyone at home who desires to use it such as the elderly, the young, the disabled. The assistance that the robot can provide are picking objects, carrying (heavy) objects, monitoring the person using sensors and assistant for sport exercises. Moreover, the Ceilbot can be used to monitor the safety and security of the house in a day-to-day basis.

2.2.2. Industrial Applications

Industrial robots in use today are fixed at one point and perform repetitive tasks in a defined workspace. However, studies such as in [16] indicate that future industries require service robots with more capabilities than the ones in use today. According to the study, the features that will be needed from industrial robots in the future are mobility, versatility, cooperative, vision-enabled, and flexibility. For that, Ceilbots can have a role to fulfill the demands. Example, supervising or monitoring the activities of different sections of the industry by the Ceilbot can be of great assistance to the supervisor running the industry.

2.2.3. Greenhouse Applications

Collecting ready flowers, ripped fruits and vegetables, spreading fertilizers, watering the plants based on sensory data and other similar tasks can be performed by the Ceilbot. This operations could be performed by the Ceilbot as an assistant or autonomously.

2.2.4. Museum

There have been some tour-guide interactive robots in use in some museums as mentioned in [17]. The main challenges the researchers encountered in the design of the robot for this particular application are safe-navigation of the robot in the dynamic environment and obstacle avoidance. These challenges can be very well simplified or completely eliminated by using Ceilbots.

2.2.5. Library

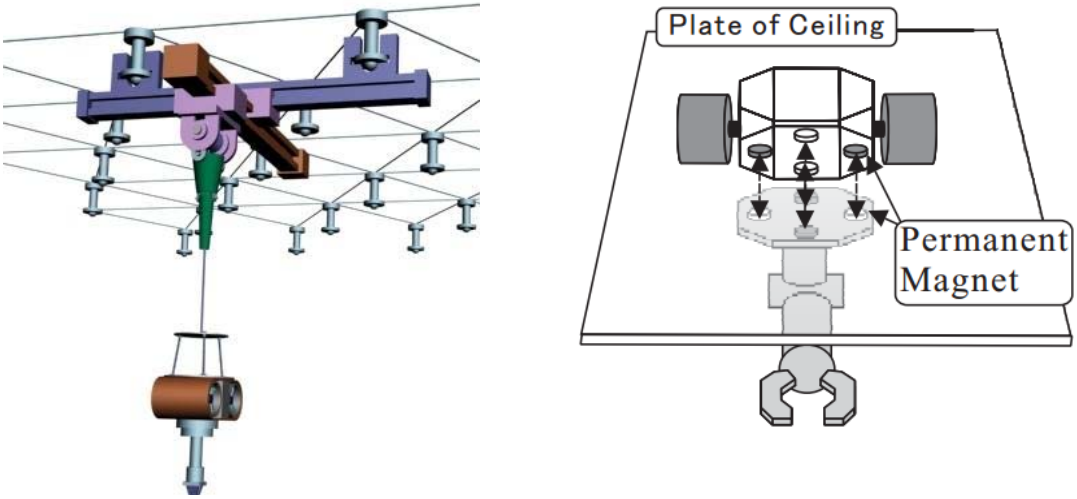
In libraries, finding books, magazines, and other resources can be automated by using Ceilbots. In this application, the Ceilbot can be ordered to bring a book or magazine to the reader/user that is in the reading room. The added advantage of this system is that the book shelves can be made taller thus saving floor space.

3. Review of Related Work

Transferring the robot from the ground to the ceiling poses its own challenges. Locomotion and manipulation are common ones that have attracted the attention of several researchers. Selection of locomotion and manipulation methods for the robot is dependent on the application for which the robot is designed to be used. In this regard, little research has been done and a few are still in development. In this section, previous works related to Ceiling mobile robots are presented.

The ACROBOTER presented in [15] is designed to crawl using the anchor points installed on the ceiling. For the manipulation purpose a swinging unit equipped with ducted fan actuators is used. This swinging unit that includes tool-changer mechanism is suspended from the climber unit (crawling unit) through a cable. To adjust the vertical position of the swinging unit, a winding mechanism is also added to the climber unit as shown in Figure 3.1 (a). This robot design is claimed to be highly innovative and novel considering its capability to operate autonomously and cover all volume of space. In addition, it can move in any direction. This particular locomotion mechanism is proposed to allow cooperation between similar robots in a given workspace. However, examining the way of crawling and the manipulation mechanism from the demonstration released by the authors, it seems that the construction is very complex and the whole operation is slow. The stability of the manipulator (the swinging unit) is also the problem they faced despite using the ducted fans for this purpose.

Another ceiling-mounted mobile robot proposed in [18] uses permanent magnets for suspending the robot on the ceiling. They used a mobile unit that contains permanent magnets on the top of a ceiling-plate. On the bottom side of the ceiling-plate another unit with permanent magnet is placed as indicated in Figure 3.1(b). The manipulator is attached to the bottom unit. Locomotion is achieved by driving the wheeled mobile unit. The lower unit is then dragged by the force developed by the magnets both on the top and lower units. The advantage of using the permanent magnets is that no electric power is needed to hold the lower unit with the upper unit even during blackouts. The ceiling-plate is also specially designed material that includes a 2D-matrix code to help the mobile unit estimate its pose. To read the matrix code, which is encoded to indicate the coordinate of that point with respect to a reference point, a CCD camera is employed. The demonstration shows that the movement is smooth at a relatively good speed (30mm/s).



(a) (b)

Fig 3.1 (a) ACROBOTER [15] (b) Permanent magnet traction mechanism [18]

The work presented in [19] demonstrates another alternative method of implementing the locomotion system to the effort discussed in [15]. The mechanism consists of two parts: inner body and outer body. The ceiling is made from a metal sheet with uniformly distributed holes on it (perforated-metal). The lower part of the outer body is connected to the manipulator. The movement of the robot is executed by both the inner and outer-bodies which both have mechanisms to hang on the perforated-metal. This is done by inserting the mechanisms on the perforated-metal alternatively i.e. the outer body attaches itself on the ceiling, then the inner body moves horizontally and hooks itself (see Figure 3.2) while the outer-body is still attached to the ceiling to carry the whole robot system. This process continues until the required pose is reached. The authors claim that their robot design (called HangBot) simplifies the complexity of control system and ceiling architecture of the ACROBOTER described above.

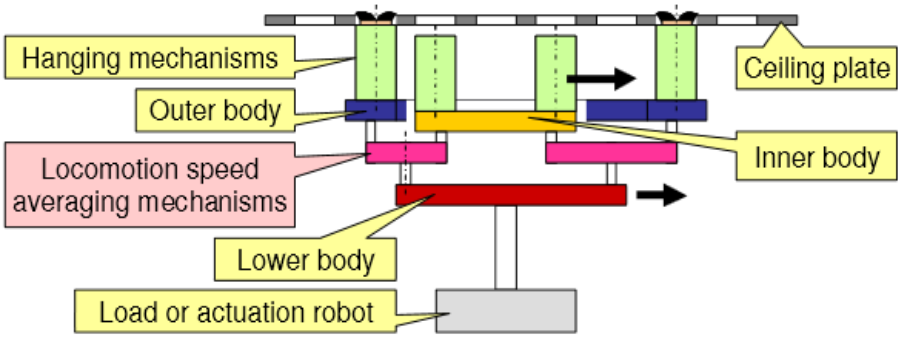


Figure 3.2 Locomotion mechanism of HangBot [19].

An undergoing research at Eindhoven University of Technology focuses on contactless energy transfer between moving and stationary part of the ceiling robot system as shown in Figure 3.3 [20]. The stationary part of the system (stator) is on the ceiling and the movable part (mover) is to be suspended by a passive vertical force generated by permanent magnets. The principle applied for the system is electromagnetic induction (transformer/induction motor action), where the primary coil is distributed on the ceiling and a single secondary coil is on the mover. Communication between the stator and the mover is also wireless. The design aims to transfer constant amount of power from the stator to the mover regardless of the position of the mover, thus avoiding cables for power transfer to the mover. That is the ultimate objective of the research. Therefore, the technique is expected to improve the acceleration of the mover and accuracy of the mover pose.

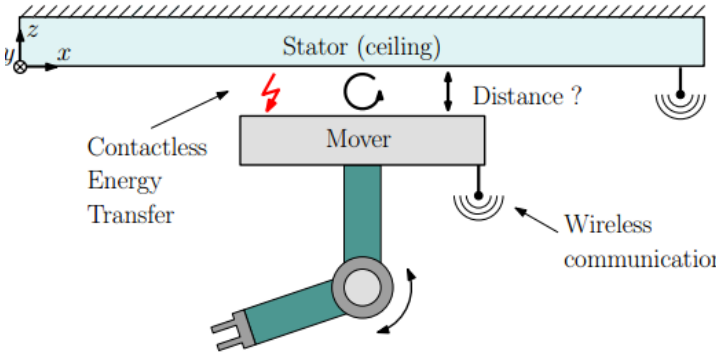


Figure 3.3 Contactless energy transfer using induction principle

At Aalto University, different types of actuator designs and methods of locomotion for the Ceilbot have been studied and simulated, but only few of them are implemented and tested. The following is a brief description of some of the works done at Aalto University [21].

A ceiling-walking robot by Juhana Leiwo and Marco Torti (2009).

The robot uses four arms with claws to grip anchors distributed uniformly on the ceiling (Figure 3.4 (a)). The robot uses one of the arms at a time in some defined sequence to move from one location to another. They also proposed another method of locomotion shown in Figure 3.4 (b). In this design the robot is attached to two actuators on each side using a pair of chains. The actuators are mounted on rails. The actuators then position the robot in three

dimensions: two horizontal movements and one vertical motion (by providing sag for the chains).

(a) (b)

Figure 3.4 (a) Ceiling walking robot (b) Robot suspended by chains between actuators

Fire Robot by Vasek Brabec and Julio Cordon Munoz (2009).

They proposed a conceptual robot whose task is firefighting. The locomotion mechanism operates by using two parallel beams installed on the walls close to the ceiling. The robot is placed on another transverse beam that connects the parallel beams. Therefore, the robot can move in a horizontal plane defined by the beams as shown in Figure 3.5.

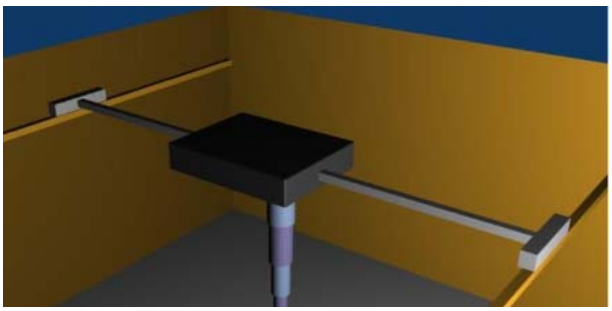


Figure 3.5 Fire robot mounted across parallel beams

Ceilbot for Sports Hall by Gillaume Mercier and Massinissa AIT-Gherbi.

The design proposed for locomotion of the robot is by pulling and releasing the four cables that hangs the robot from four corners of the building. As illustrated in Figure 3.6, four motors with a pulley system are used to pull and release the cables, thus adjusting the pose of the robot. Stability of the robot is maintained by a stabilizer fitted on top of the robot.

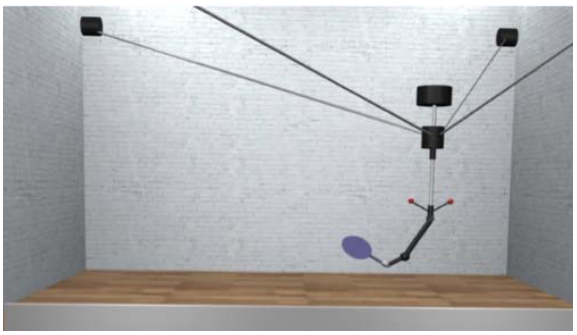


Figure 3.6 Ceilbot suspension systems for sports hall application

Ceilbot for home environment by Juuso Kinnunen and Ville Rahikka (2009).

In their work, a robot for general purpose applications for home environment is presented. The proposed robot uses small wheels to move on a network of rails or tracks laid on the ceiling. One of the problems of using rails (track) for locomotion is that the robot cannot reach all parts of the area under the ceiling. However, this problem can be circumvented by using a manipulator as shown in Figure 3.7. The other known problem of using rails is that it does not allow multirobot systems. The proposed system, however, attempts to solve the first problem by using rail segments (turntables) that can be adjusted to create different routes for the robot (See Figure 3.7). This same technique could also be used to support multirobot system.

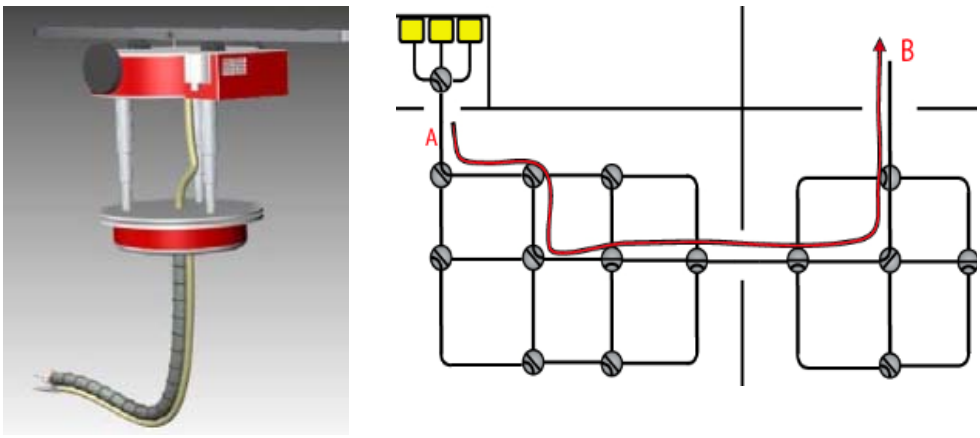


Figure 3.7 Ceilbot and the rail network

In general, the manipulator designs that are used in two of the conceptual designs are elephant trunk-like structure called continuum manipulator. The other two robots use slightly different type of manipulators.

4. Ceilbot System

4.1. System components

The overall system integration of the Ceilbot is shown in Figure 4.1. As illustrated in the figure, the Ceilbot is composed of seven components/modules each contributing a role in the system control except the manipulator component.

Computer

This module is the central part of the system. It runs the algorithms for object recognition and motor control. For the object recognition algorithm, input image comes from the Kinect camera and for the motor control algorithm, feedback signal comes from the encoder through the microcontroller module. In addition, orientation of camera, manipulator angular position, switch status and other control signals are monitored and processed in this module. This computer has Intel Dual core ATOM N330, 1.6 GHz processor and NVIDIA ION graphics processor.

Kinect Camera

This module is used to take colored images of the scene and its corresponding 3D information. The camera has a servo motor attached at its bottom for tilt angle adjustment and it can be controlled manually by the user and by the object recognition algorithm that runs in the computer (module).

Switch

This is a microswitch in the normally open configuration. It is used to detect the end positions of the track on which the Ceilbot is working. In addition, it is used for calibration of the Ceilbot position measurement. At the end points of the track, a bracket is attached to trigger the switch.

Manipulator

This module is a two degree-of-freedom manipulator constructed using two servo motors. A laser pointer is used as the end-effecter. It should be noted that this manipulator is mimicking

a non-existing manipulator arm just for demonstration purposes. Detailed description of the manipulator is presented in section 4.4.

Motor and motor driver

The motor used is a DC brushed type. The motor driver circuit uses the L9904 H-bridge controller chip for the motor control and positive voltage regulator (78100A chip) to power the controller chip. AEDA-3300 encoder is also attached to the shaft of the motor to get angular position feedback.

Microcontroller

This is AVR's ATmega2560 microcontroller (Arduino Mega 2560 board). It is used as an interface between the computer and the other modules. In addition, the microcontroller is used to process the raw data from the switch and the encoder.

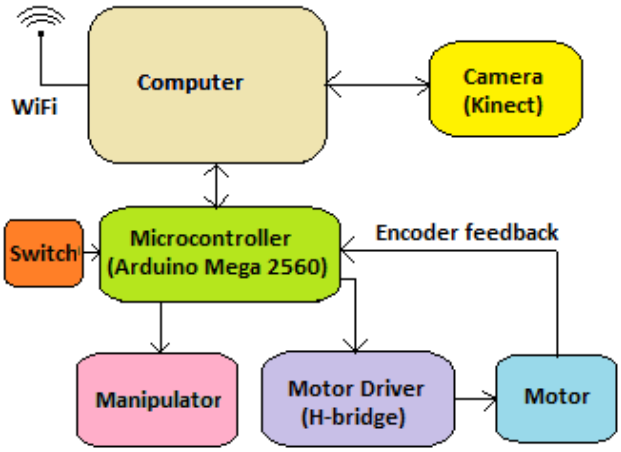


Figure 4.1 Ceilbot system modules

4.2. Software Libraries and Tools

To build the Ceilbot system from the modules described in section 4.1, the following open source software libraries and tools are used.

4.2.1. Open source Computer Vision (OpenCV) and Point Cloud Library (PCL)

OpenCV and PCL are libraries of functions implementing a variety of algorithms mainly for robotics, machine learning and computer vision applications. The libraries are currently developed and supported by a robotics research lab, Willow Garage [22]. In the Ceilbot development, the OpenCV libraries and the PCL are used for image and point cloud data processing respectively to identify the presence of the required object in the scene and extract its 3D location information.

4.2.2. Robot Operating System (ROS)

ROS is a software framework that gives structured communications layer on top of the host operating system (mostly Ubuntu and other Linux systems) [23]. It provides infrastructures such as hardware abstraction, message passing between two or more processes running on a single machine or in a networked systems and other services similar to what a standard operating system offers. Due to its rich features and functionalities, the framework is also called a meta-operating system on top of which one can develop a robotic application software.

ROS offers the message passing service between processes using publisher-subscriber model. The software modules/processes are considered as *nodes* and each of these nodes can *publish* the message on a *topic* and the other nodes must *subscribe* to the topic to have access to the published messages. A single node can publish and subscribe to multiple topics. This makes the communication between nodes asynchronous. ROS also provides synchronous mode of communication referred to as *service* [23] [24].

4.2.3. Qt integrated development environment (IDE)

Qt is a cross platform IDE used for the development of application software including graphical user interface (GUI) in C++ programming language. This open-source IDE is currently developed by the *Qt project* [25]. This IDE has been used for the development of the GUI for the Ceilbot application. In addition, the Qt framework allows linking and integration of the openCV library, the PCL and the ROS packages making application development more convenient.

4.3. Motor Controller

One of the major modules of the Ceilbot is the motorized trolley. It consists of a DC motor and wheels. This module fits into a track attached to the ceiling to form the locomotion system of the Ceilbot. Although there are considerable number of DC motor control methods/algorithms, proportional-integral-derivative (PID) controller is the most popular control algorithm used in industry [26]. For the Ceilbot motor control, the proportional term is only considered and the integral and differential terms are ignored. The structure of the control system is shown in Figure 4.2. Feedback signal about the current position of the robot/motor is obtained from the encoder mounted on the shaft of the motor. The desired position of the robot can be set manually by the user or if semi-autonomous operation is used, the value of position can be set by the Ceilbot itself.

4.3.1. PID control introduction

The PID controller is a closed loop feedback control system that takes the difference between the set point value and the measured value as an input. The difference value is called the error signal and it is denoted by $e(t)$. The main objective of the controller is to make the error signal zero or as close to zero as possible. To achieve that the error signal is treated in three separate ways and added to produce the final controller output. The first treatment is scaling of the error by a proportionality gain K_p to produce the proportional term $K_p.e(t)$. However, the proportional term alone is often not enough to produce the desired effect on the system under control. This is due to the steady-state error left by the proportional term. This is where the second treatment of the error signal comes into play to improve the previous controller output. In this case, the instantaneous error of the system, which can be positive or negative in sign at different time intervals, is added and scaled by a constant called integral gain K_i to produce the term given by $K_i.\int_0^t e(\tau)d\tau$. To improve performance of the controller, the third component called the derivative term $K_d.\frac{d}{dt}e(t)$ can be considered to enhance the controller output. Therefore, the final output of the controller is obtained by adding the three terms [26]

$$Controller\ output = K_p.e(t) + K_i.\int_0^t e(\tau)d\tau + K_d.\frac{d}{dt}e(t) \quad (1)$$

4.3.2. Position Control

The position of the robot is controlled by the controller block shown in figure 4.2. The technique used for this purpose is simple error minimization, i.e. the controller simply drives the motor at some speed until the error is in the tolerance range. The error tolerance is predefined and it is less than 3 mm (it can be modified by the user). The direction of driving is determined by the sign of the error signal.

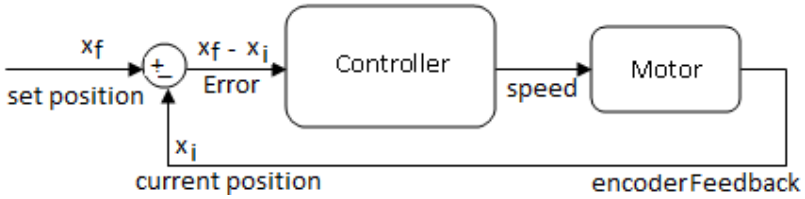


Figure 4.2 Structure for the motor control

4.3.3. Speed control

The speed at which the Ceilbot moves over the track is determined by the magnitude of the error signal. The controller shown in Figure 4.2 takes error signal as an input from which it determines the direction and the distance the robot has to travel. It also calculates the speed at which the robot should move to reach the set-point (desired position). To reach the desired position the speed curve could be either a straight line (constant speed) for short distances or a trapezium shaped curve for longer distances. The advantage of using a trapezium shaped speed curve is to avoid abrupt starting and breaking of the motor. In addition, the smooth stop operation allows more accurate positioning of the robot. More detailed explanation of the speed control mechanism is presented in the next sections.

4.3.4. Speed control method

As briefly introduced in the previous section, the speed of the motor is determined by the magnitude of the position error signal. For short set-point distances ($\leq 0.4\,m$) driving the motor at high speed would not be efficient. For longer set-point distances ($> 0.4\,m$) the robot has to get faster to the desired position which implies that the motor has to be driven faster. However, driving the motor faster has its own problems. The first problem is the difficulty of accurate positioning of the robot. This is due to the fact that the feedback obtained from encoder changes very fast given its quadrature configuration for better resolution. Thus, this

makes it difficult to put the robot in the error tolerance limit. In addition, stopping the motor suddenly from the maximum speed can cause mechanical stress on the motor parts such as gears. The second problem is due to starting of the robot at maximum speed. This will have side effects on the power supply. The characteristic of the motor indicates that during starting, the motor draws high amount of current from the power supply. This current, commonly referred to as inrush current, is the peak current the motor draws for its operation. For our application, the Ceilbot can be frequently stopped and started. This frequent on-off operation might result in loading on the power source.

The proposed solutions for the two problems aforementioned are the following. The motor should be driven at the minimum speed while starting and increase the speed linearly until it reaches the maximum speed. This eliminates the demand for maximum current. Driving slow when robot approaches the desired position also avoids the problem of inaccurate positioning as well as possible mechanical problems. This slow driving makes sure that the rate of change of position feedback from the encoder to be slow which in turn allows the controller to stop the robot at the desired position with tolerable error.

The problems described are well known in servo control systems in industrial automation and robotics fields and the solutions proposed are also known in the same fields except that the techniques are adapted to the Ceilbot.

4.3.5. Speed control method analysis

The track used for the Ceilbot development is 3 m long as shown in Figure 4.3. Here, the left end of the track is considered as the reference or starting point of the robot. The position of the robot is always measured with respect to this reference point. Suppose the robot is now at x_i distance from the reference point and the robot is required to go to the x_f position ($x_f > x_i$). For this scenario, there are two options for the robot to reach the desired position. The first option would be moving the robot at a constant speed regardless of how fast or slow the speed is. However, as discussed above, higher constant speed will be problematic in terms of accurate positioning and potential mechanical issues. Low speed driving may not also be desirable especially if the desired location is significantly large (say $> 0.4\,m$) as it takes longer time to get to the final position. The best option is, therefore, to use variable speed depending on the magnitude of the error signal. This method takes care of all the issues discussed earlier. The error magnitude simply shows that the desired position is far from the

current position of the Ceilbot and vice versa. Therefore, for short distances ($< 0.4\ m$), the robot drives slowly at a predefined constant speed and for larger distances ($> 0.4\ m$) the robot follows the trapezium shaped speed curve as illustrated in Figure 4.3.

The speed curve has three parts; acceleration, constant maximum speed and deceleration. In the acceleration/deceleration part, the robot increases/decreases speed linearly until it reaches the defined maximum/minimum speed. The slope of the acceleration/deceleration line can be adjusted by choosing distance d. The smaller the value of d the faster the robot reaches the maximum speed and vice versa. In the constant speed curve, the robot moves at the predefined maximum speed until it is d distances away from the final position.
The slope of the acceleration line is given by

$$slope = \frac{1-p_o}{x'-x_i}, \tag{2}$$

where x' is position of the robot after traveling d distance, x_i is the initial position and p_o is the initial speed at position x_i. Here, the speed is scaled by a factor of 255. Position x' can be expressed in terms of other parameters that do not change until the robot reaches the final position.

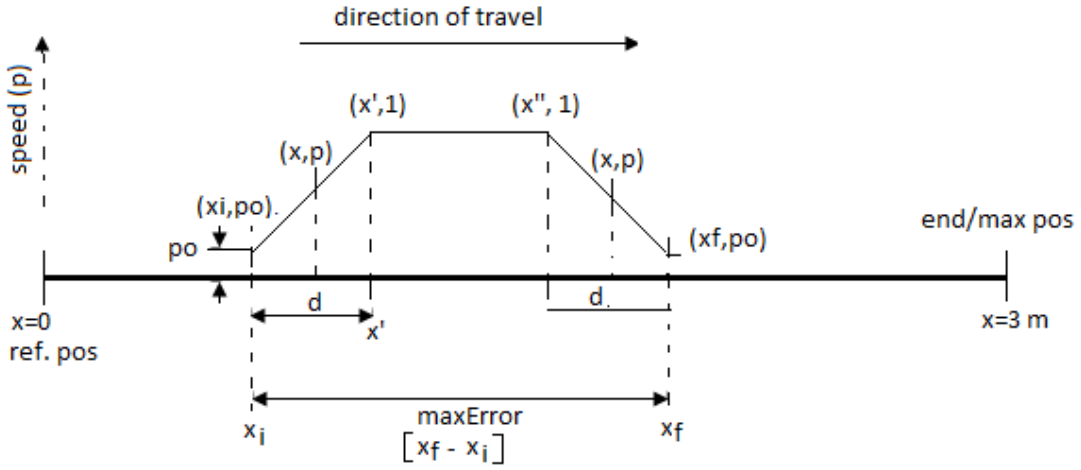


Figure 4.3 Position vs. speed when $x_f > x_i$

$$x' = x_f - (maxError - d) = x_f - maxError + d, \tag{3}$$

Similarly, for x_i,

$$x_i = x_f - maxError \tag{4}$$

where x_f is the final position and $maxError$ is the absolute value of difference between the final and initial positions. The error is calculated once only when a new robot position is given/known.

From Figure 4.3, $x' - x_i = d$. Therefore, the slope of the line is given by

$$slope = \frac{p-p_o}{x-x_i} = \frac{1-p_o}{x'-x_i} = \frac{1-p_o}{d} \tag{5}$$

Substituting (4) into (5) gives,

$$\frac{p-p_o}{x-x_f+maxError} = \frac{1-p_o}{d}$$

$$p = p_o + \frac{1-p_o}{d}\,(x - x_f + maxError) \tag{6}$$

The equation shown in (6) shows that the speed p is dependent on the position x. The rest of the parameters are constants except that the $maxError$ value always changes whenever x_f changes. Once the robot arrives at the x' position, its speed stays at the maximum. When the robot arrives x'', which is d distance away from the x_f position, it starts decelerating and follows a similar speed curve as in (6) except that the slop is negative. The modified equation for the deceleration curve is as follows.

$$slope = \frac{1-p_o}{x''-x_f}, \tag{7}$$

but $x'' = x_f - d \Rightarrow x'' - x_f = -d$. Substituting in to the (7) gives,

$$slope = -\frac{1-p_o}{d} \tag{8}$$

Therefore, the equation of the line is given by,

$$\frac{p-p_o}{x-x_f} = -\frac{1-p_o}{d}$$

$$p = p_o - \frac{1-p_o}{d}\,(x - x_f) \tag{9}$$

As a summary, if the robot is to travel from x_i to x_f, where $x_f > x_i$, then speed of robot is defined by the three equations given in (10)

$$p = 255 * \begin{cases} \left[p_o + \frac{1-p_o}{d}\,\left(x - x_f + maxError\right)\right], & if(|error| > (maxError - d)) \\ \left[p_o - \frac{1-p_o}{d}\,\left(x - x_f\right)\right], & if(|error| < d) \\ [1\,], & otherwise \end{cases} \tag{10}$$

where *error* is the instantaneous position difference between x_f and x_i which reduces as the robot advances to the x_f position.

If the robot is required to move from x_i to x_f, for $x_f < x_i$, a similar analysis as above is followed based on Figure 4.4. For the first ramp (acceleration curve) the line equation is given by

$$slope = \frac{p-p_o}{x-x_i} = \frac{p_o-1}{x_i-x'}, \tag{11}$$

where $x' = x_f + maxError - d$ and $x_i = x_f + maxError$. Substituting x' and x_i values into (11) and solving for p yields

$$p = p_o - \frac{1-p_o}{d}(x - x_f - maxError) \tag{12}$$

Similarly, for the second ramp (deceleration curve), the line slope is given by

$$slope = \frac{p-p_o}{x-x_f} = \frac{1-p_o}{x''-x_f}\ , \tag{13}$$

where $x'' = x_f + d$. Solving for p after inserting x'' into (13) gives the equation of the line,

$$p = p_o + \frac{1-p_o}{d}\,(x - x_f) \tag{14}$$

To summarize, when $x_f < x_i$, the governing equations of speed of the robot are

$$p = -255 * \begin{cases} \left[p_o - \frac{1-p_o}{d}\,(x - x_f - maxError)\right], & if(|error| > (maxError - d)) \\ \left[p_o + \frac{1-p_o}{d}\,(x - x_f)\right], & if(|error| < d) \\ [1\,], & otherwise \end{cases} \tag{15}$$

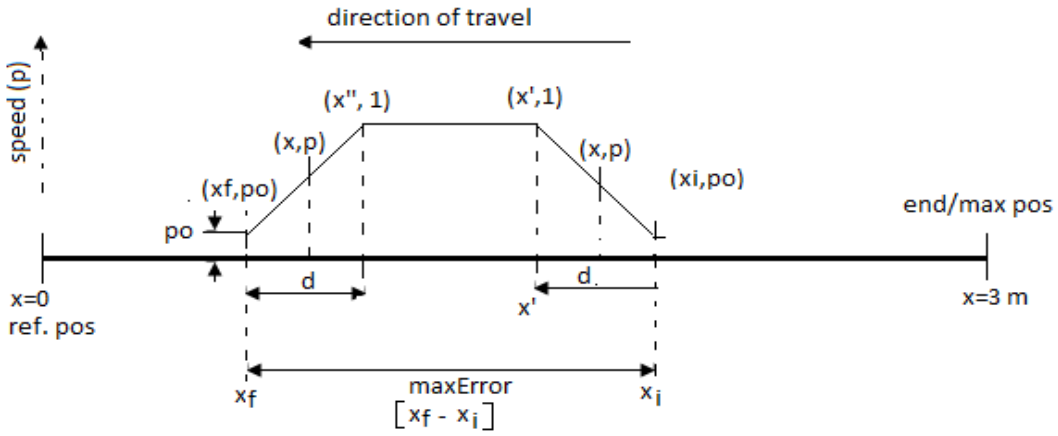


Figure 4.4 Position vs. speed when $x_f < x_i$

The pseudocode for the speed control system of the Ceilbot is shown in algorithm 4.1. The robot follows the trapezium speed curve only when the position error is greater than $0.4\,m$. This value can be adjusted by the user. Otherwise, it moves at a selected constant speed. This constant speed is fast enough to travel the $0.4\,m$ and slow enough to decode the encoder feedback signal for accurate positioning. It should be noted that the speed value has a range of 0 to 255. The output of the speed control is fed to an 8 bit microcontroller which in turn drives the motor through an H-bridge motor driver. Therefore, this speed output is the pulse width modulation (PWM) required to drive the motor.

4.3.6. Comparison with PI controller

The implementation discussed in the previous section is similar to proportional controller (P controller) in terms of the basic principle. The difference between the standard *P* controller

and the proposed controller lies on the way the proportional gain, K_p value is used. In the case of the P controller, carefully selected proportional gain K_p is used all the time. The product of the gain and the error signal produces the control output. This implementation works for the Ceilbot with little modifications. However, it lacks flexibility and during starting the controller output will be high as the error is always the highest every time the robot starts and it signals the motor to run at the highest speed. This can have potential problems on the motor and the power supply as discussed in the previous section. The proposed controller, however, provides flexibility to the user to adjust parameters such as maximum speed determination and the time to reach the maximum speed.

Algorithm 4.1 Speed and Position control algorithm pseudocode

$error = x_f - x$
$if(|error| > maxError)$
    $maxError = |error|$

$if(x_f > x \;\&\&\; !(|error| < 0.003))$ // error tolerance 0.3 cm
    $if(maxError > 0.4)$
        $if(|error| > (maxError - d))$
            $p = 255 * \left[[\, p_0 + \frac{(1-po)}{d} * (x - x_f + maxError)]\right]$
        $else\ if(|error| < d)$
            $p = 255 * \left[p_0 - \frac{(1-p_0)}{d} * (x - x_f)\right]$
        $else$
            $p = 255$
    $else$
        $p = 42$ // for error <0.4 m, robot moves at constant speed 42

$else\ if(x_f < x \;\&\&\; !(|error| < 0.003))$
    $if(maxError > 0.4)$ //0.4 meter
        $if(|error| > (maxError - d))$
            $p = -255 * \left[p_0 - \frac{(1-p_0)}{d} * (x - x_f - maxError)\right]$
        $else\ if(|error| < d)$
            $p = -255 * \left[p_0 + \frac{(1-p_0)}{d} * (x - x_f)\right]$
        $else$
            $p = -255$
    $else$
        $p = -42$
$else$
    $p = 0$

A pure P controller is known to have a steady-state error. To avoid this error an integral term is usually added with the P term. The controller becomes the standard PI controller. Again, carefully selected integral gain K_i is used. However, the use of the integral term can cause a prolonged overshoot that can potentially lead to undesired response. This problem is formally known as integral windup and it happens when a new desired position is provided to the controller and the error is large. In this case the integral term accumulates more error while the robot is moving towards the desired position. When the overshoot happens, the error will change sign and its magnitude is less than the previously accumulated error just before the overshoot starts. As a result the overshoot stays for an extended period until it offsets the error. Unlike the standard PI controller, the proposed controller do not directly use integral term or it does not add errors, which means that it is not prone to the integral windup problem. Instead the controller tries to minimize the error by driving back and forth until the instantaneous error is in the tolerance range. The driving speed is made to be slow so that the oscillation gets damped very quickly or does not happen at all. The controller produces the response curve similar to the one shown in Figure 4.5.

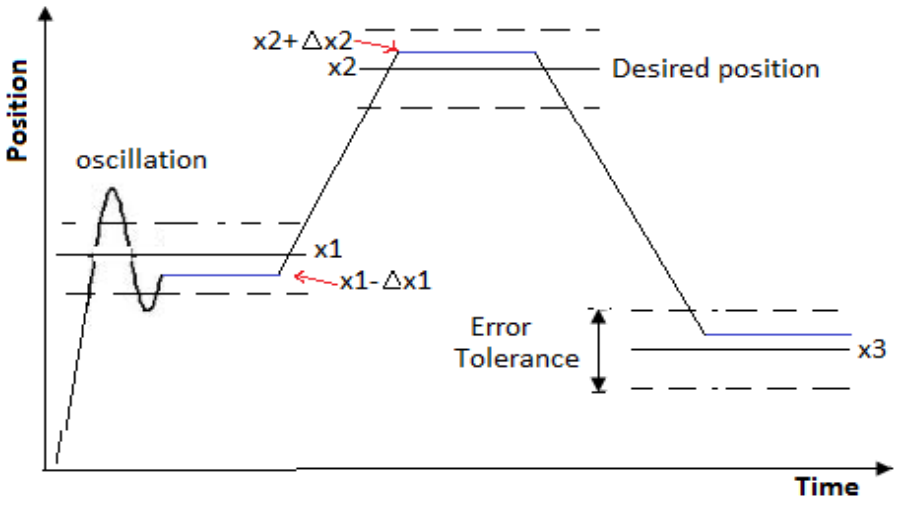


Figure 4.5 Possible response(s) of the proposed position controller (curve is synthesized)

The curve shown in Figure 4.5 illustrates three possible cases of approaching the desired position. The first case is a small oscillation around the desired position and settles a little above or below the desired position in the tolerance range. This case occurs rarely and it happens because the speed of the robot is not slow enough to settle right away. The second and third cases are that the Ceilbot settles without oscillation just above or below the desired position in the tolerance range.

As mentioned previously, the proposed controller does not accumulate errors. Instead it takes into consideration the last residual error into the calculation of the new position error. As an example, suppose the robot is instructed to move to position $x1$ from zero position as shown in figure 4.5. Assume that, the robot has moved to position $x1$, oscillated and settled at $x1 - \Delta x1$ position. In this case, the residual position error is $\Delta x1$. Then the robot is, say, instructed to move to $x2$ position. Now, the controller calculates the new position error as

$$desired\ position\ (x2) - \ current\ position(x - \Delta x1)$$

$$= x2 - (x1 - \Delta x1) \tag{16}$$

Based on this new position error, the controller drives the robot and, say, it ended at position $x2 + \Delta x2$. Here, the new error is not a result of previous error. It is simply a consequence of the error tolerance already set in the beginning. This way the controller avoids accumulating errors.

4.4. Object Recognition and position determination

As discussed in section 4.2.4, in order for the motorized trolley to move from one position to another position on the track, the desired position (set-point) has to be given. The desired position of the Ceilbot can be directly provided by the user operating the Ceilbot - manual operation mode, or the Ceilbot itself can calculate the desired position by detecting the object of interest using the Kinect camera - semi-autonomous operation mode. In this section, object recognition and object position determination methods are presented.

4.4.1. Object recognition

Object recognition is the primary operation that the Ceilbot will do if the mode of operation of the Ceilbot is chosen to be autonomous. Kinect camera is used for taking colored pictures of the scene where the object is to be found. The Ceilbot computer processes the images to recognize if the desired object is in the scene. However, recognizing different types of objects reliably based on only images is a challenging task.

There are many different algorithms that can be used for object recognition [27]. These algorithms always search some distinctive features that describes the objects to be recognized. However, these object features do not appear the same all the time for the

algorithms. Various conditions such as intensity of light in the scene, orientation of the object, distance of object from camera or the size of object affect the performance of the algorithms. This leads to failure of recognizing the object of interest. Therefore, the algorithms that are available today for object recognition have their own pre-conditions to be satisfied to guarantee consistent performance in terms of recognizing the required object. A brief review of commonly used object recognition algorithms are presented in the next section.

4.4.1.1. Template matching

This technique uses template images as the initial information about the object to be recognized and the algorithm searches through the scene image, usually pixel by pixel, to find a match between the template and scene images. This simple matching technique can be slower if the scene image dimension or resolution is high, although there are some methods that can be used to speed up the searching process. This technique, however, may fail or perform poorly if orientation, size and intensity of the scene image changes. To improve the performance of this technique appreciable number of research has been done. The research presented in [28], for example, demonstrates that template matching algorithms can be modified in such a way that it can still perform well regardless of orientation, scale and illumination changes in the scene image.

4.4.1.2. Scale Invariant Feature Transform (SIFT) and Speeded Up Robust Feature(SURF)

These algorithms are popular in computer vision applications such as object recognition. These algorithms are slightly different in the way they compute the feature descriptors from the image of the object. Nevertheless, in both methods, the algorithms must be trained about the object to be recognized before they are used for actual recognition. Information about the objects are then stored in a database for reference. Unlike template matching algorithm, the information about the objects is not stored in the form of image. Instead, the algorithms extract key descriptors (distinctive features) of a given object and store the resulting array of data in the database. When a scene is provided to the algorithm, it extracts the features of the scene and the matching algorithm compares each feature point with the stored features in the database to find the match.

In terms of speed of computation, research results show that SURF is faster than SIFT [29], [30]. However, in terms of robustness in object recognition, these algorithms, in general, perform well. When the performance of each of these algorithms are evaluated in terms of scale variation, orientation change, illumination variation and blur of the scene image, one algorithms is usually better than the other.

4.4.1.3. Color based object recognition

This method uses color of the object as the feature for detection and recognition. If the object has a single color, then recognizing the object is usually simple as long as the illumination over the object is the same. However, if the object is multi-colored, then a simple color detection will fail to work. Improvements on the algorithm for multicolored object recognition is necessary as presented in [31].

One of the attractive feature of this algorithm is that it is invariant with the scaling of the object and orientation of the object as well. In addition, it performs well even when the object in the scene is partially occluded. Implementation of the algorithm is also relatively simple. Despite its merits, its performance degrades if illumination level on the object varies.

Object recognition, in general, is still the challenging problem in computer vision field. As discussed in the previous sections, there is no single object recognition algorithm that can perform consistently well regardless of illumination, orientation, blur and size variations. In this thesis, color based object recognition algorithm is selected for it is relatively simple to implement. In terms of performance, it is also invariant for changes in view angle, size, and orientation of the object.

To recognize the object of interest, just like all other object recognition algorithms, the features of the object has to be taught to the algorithm i.e. learning the features of that particular object to be recognized is the primary operation. This is done by converting the Red, Green and Blue (RGB) image into Hue, Saturation and Value (HSV) format and then adjusting the values of H, S and V to filter out the required color. The feature of the object is saved in terms of range of H, S and V values (H-minimum and H-maximum, S-minimum and S-maximum, V-minimum and V-maximum). These HSV values represent the color of the object at a given illumination level.

When a new image is available, the recognition algorithm first uses the HSV values previously stored to filter out the color, then it extracts the edges of the filtered image and calculates the contours. Based on the contour calculation result, the area covered by the contour is calculated. Then, the calculated area is compared to a threshold value to decide whether the object is recognized or not. The threshold is used to avoid false areas due to noise left after filtering. If the area is larger than the threshold, then the object is considered to be recognized and centroid of the area is calculated for the position estimation of the object. The procedure is illustrated in Figure 4.6 and the pseudocode for the object recognition algorithm is shown in algorithm 4.2. The functions used in the pseudocode are available in the OpenCV library.

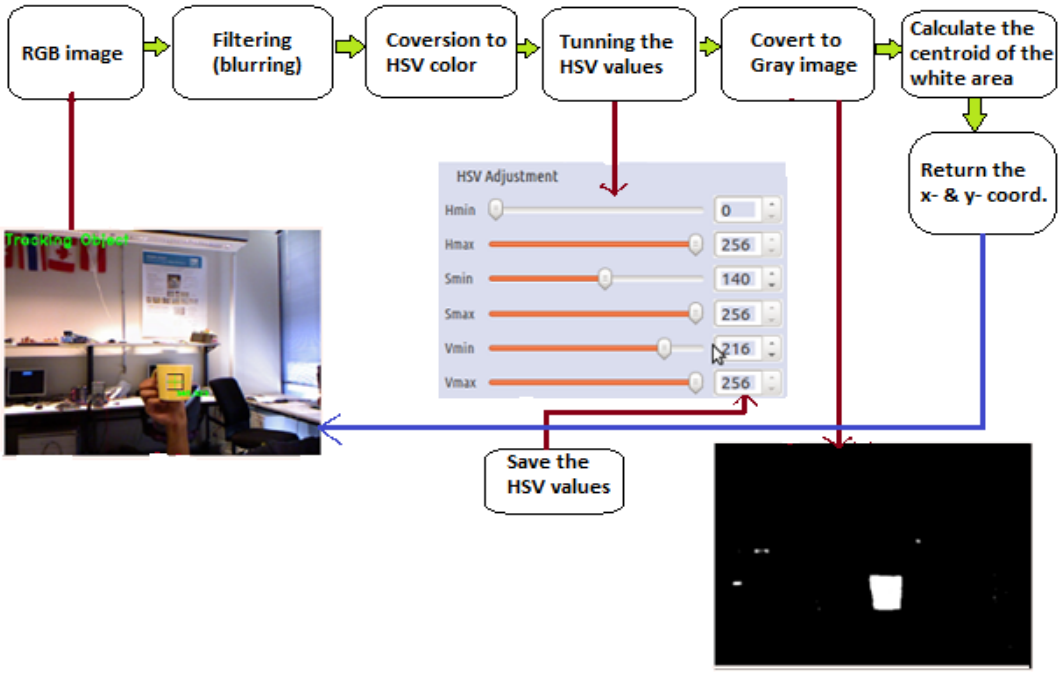


Figure 4.6 Color based object recognition procedures

4.4.2. Object Position determination

In order for the Ceilbot to work semi-autonomously, one of the tasks that it needs to do is finding the 3D location of the object. This is accomplished by using the Kinect camera depth sensor. The depth sensor captures the distance of the scene points with respect to the camera optical frame. Using the point cloud library (PCL), the 3D position of each of the scene points can be determined.

Algorithm 4.2 Object recognition algorithm pseduocode using color as feature

```
get image from the camera
apply gaussian blurring
convert RGB image to HSV image
filter the HSV image by varying HSV values
apply gaussian bluring
apply morphological erosion    //eliminates a portion of noise
apply morphological dilation   //enhances the required pixels
extract the edges of the filtered image using Canny edge detector
find the contours and hierarchy   //finds the contours and stores them In a hierarchy
if(hierarch.size( ) > 0 )
    number of objects found = hierarch.size( )   //some of them are due to noise
    if(number of objects found < maximum number of objects expected)
        for(int i = 0; i ≥ 0, i = hierarchy[i][0])
            moment = moments(contours[i])
            area[i] = moment.m00
            momentX[i] = moment.m10
            momentY[i] = moment.m01
        end for
        for(int j = 0; j < number of objects found; j + +)
            maximum area = max(area[j]) , index = j
                //max area corresponds to object
        end for
        if(maximum area > threshold area)
            object recognized = true
            x coordinate of object = momentX[index] / maximum area
            y coordinate of object = momentY[index] / maximum area
        else
            object recognized = false
        end if
    end if
end if
```

For the Ceilbot, the object recognition algorithm calculates the x- and y-coordinates of the object location in pixels with respect to the image plane reference point. The image plane has

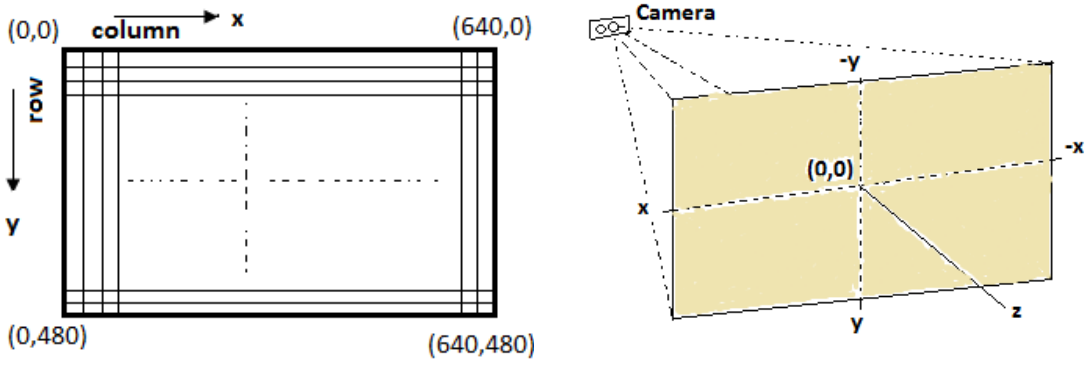
a dimension of 640 x 480 pixels, each pixel representing the intensity value. The reference position of the image plane is the top left corner as shown in Figure 4.7 (a). Therefore, every pixel position can be referenced by the x-coordinate (column) and the y-coordinate (row) with respect to the reference position.

Similarly, the depth image has a dimension of 640 x 480 pixels. In this case, each pixel contains the color information and x-, y- and z-coordinates of the scene in units of millimeters or meters with respect to the reference position located at the center of the depth image plane as shown in Figure 4.7 (b).

Now, given two image of similar dimension (one image from the RGB camera and the other from the depth camera), a one-to-one correspondence can be made between pixels in the respective images. Each pixel in the RGB image can be accessed using the row and column information. Whereas, the pixels in the depth image are accessed by their position number. Each pixel has its own number from 0 to $640x480 - 1 = 307{,}199$ arranged as shown in figure 4.8. Thus, the correspondence equation from the RGB pixel to the point cloud pixel (depth) is given by

$$pixel_{depth} = (row_{rgb} - 1) * 640 + column_{rgb} \tag{17}$$

Therefore, from (17) the pixel containing the position information about the recognized object is obtained. The position of the centroid of the object is now available for the Ceilbot to perform its task.



(a) (b)

Figure 4.7 (a) Image plane (of an RGB camera) and pixel referencing and (b) Depth image plane and its coordinate system. The point cloud is projected on this image plane.

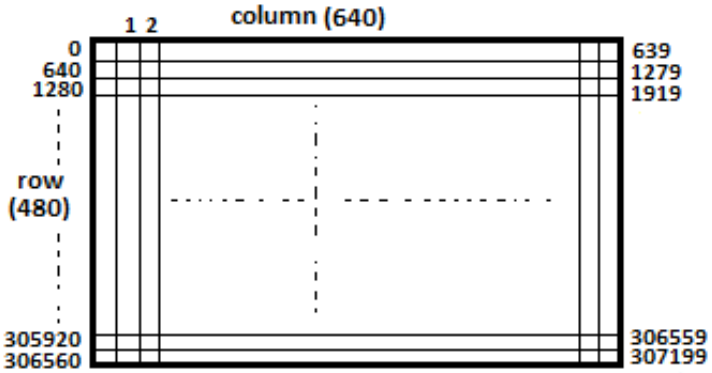


Figure 4.8 Addressing of the pixels of the depth image (point cloud)

4.5. Manipulator

After the Ceilbot determines the 3D position of the object, the next step would be to approach the object and perform the desired manipulation. A simple manipulator is, therefore, designed just for demonstration purposes as a replacement of a non-existing manipulator arm. The manipulator is constructed using two servo motors each having 180 degrees of maximum rotation angle. These two servo motors provide two degrees-of-freedom of motion for the manipulator. The end-effecter is a laser pointer and when the Ceilbot reaches the object, it points to the object using the pointer. The configuration of the manipulator is shown in Figure 4.9.

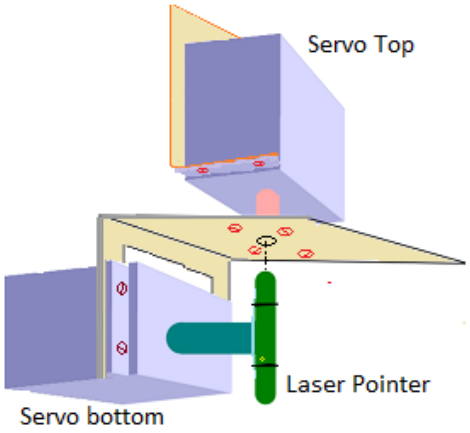


Figure 4.9 Model manipulator using two servomotors and laser pointer as an end effecter

At this point, the position of the object is known. For a successful manipulation of the object the angles of the servo motors need to be calculated from the 3D position information of the object. This is commonly known as inverse kinematics of manipulators. Geometric approach is used to solve the inverse kinematics problem. Shown in Figure 4.10 is the 3D geometry depiction of the manipulator, the camera and the object.

The manipulator is H distances below the camera. The end-effecter of the manipulator and the camera reference point are on the same axis (y-axis). Assume that the object is at point A which corresponds to the second quadrant of the image plane (as shown in Figure 4.7 (b)).

Considering triangle OCB, with right angle at C, one of the servo angles (θ) can be calculated as

$$\theta = \tan^{-1}\left(\frac{-x}{z}\right) \tag{18}$$

Similarly, from triangle OBA, with the right angle at vertex B, the second servo angle (β) can be obtained.

$$\beta = \tan^{-1}\left(\frac{y-H}{\sqrt{x^2+z^2}}\right) \tag{19}$$

The sign of the coordinates simply indicates the quadrant in the image plane at which the object location is projected. Therefore, by first determining the quadrant based on the signs of the x- and y-coordinates, (18) and (19) can be modified as follows

For quadrant II,

$$\theta = \tan^{-1}\left(\frac{|x|}{z}\right)$$
$$\beta = \tan^{-1}\left(\frac{y-H}{\sqrt{x^2+z^2}}\right) \tag{20}$$

Applying the same procedure as above, the servo angles can be calculated for other quadrants. The summary of equations are shown in Table 4.1.

The equations shown above provide the angular position for each servo motor from the reference position. In this case, the reference position is set to be 90 degrees for both servo motors and at this reference position the laser pointer is adjusted in such a way that it is aligned in parallel with the z-axis except that the pointer is translated H distances along the y-axis as illustrated in Figure 4.10.

Table 4.1 Summary of equation used to calculate servomotor angles

Servo Angle	Quadrant I	Quadrant II	Quadrant III	Quadrant IV
$\boldsymbol{\theta}$	$\tan^{-1}\left(\frac{x}{z}\right)$	$\tan^{-1}\left(\frac{\lvert x\rvert}{z}\right)$	$\tan^{-1}\left(\frac{\lvert x\rvert}{z}\right)$	$\tan^{-1}\left(\frac{x}{z}\right)$
$\boldsymbol{\beta}$	$\tan^{-1}\left(\frac{y-H}{\sqrt{x^2+z^2}}\right)$	$\tan^{-1}\left(\frac{y-H}{\sqrt{x^2+z^2}}\right)$	$\tan^{-1}\left(\frac{\lvert y\rvert+H}{\sqrt{x^2+z^2}}\right)$	$\tan^{-1}\left(\frac{\lvert y\rvert+H}{\sqrt{x^2+z^2}}\right)$

Now, the magnitude of the angular position with respect to the reference position is known from θ and β, where θ represents the pan angle corresponding to the top servo motor and the angle β represents the tilt angle corresponding to the bottom servo motor. The direction of rotation is determined from the signs of x- and y-coordinates or the quadrants.

Therefore, the final angular position of the servo motors is determined by adding/subtracting the values of θ and β with/from the reference angle. Thus, the final angular position of the servo motors is determined by the equations summarized in table 4.2. Figure 4.10 shows the angular position ranges of the servo motors and their set reference position (90 degrees).

Table 4.2 Summary of equations for angular position calculation

	Top Servo [Pan Angle]	Bottom Servo [Tilt Angle]
Quadrant I	$90+\theta$	$90+\beta$
Quadrant II	$90-\theta$	$90+\beta$
Quadrant III	$90-\theta$	$90-\beta$
Quadrant IV	$90+\theta$	$90-\beta$

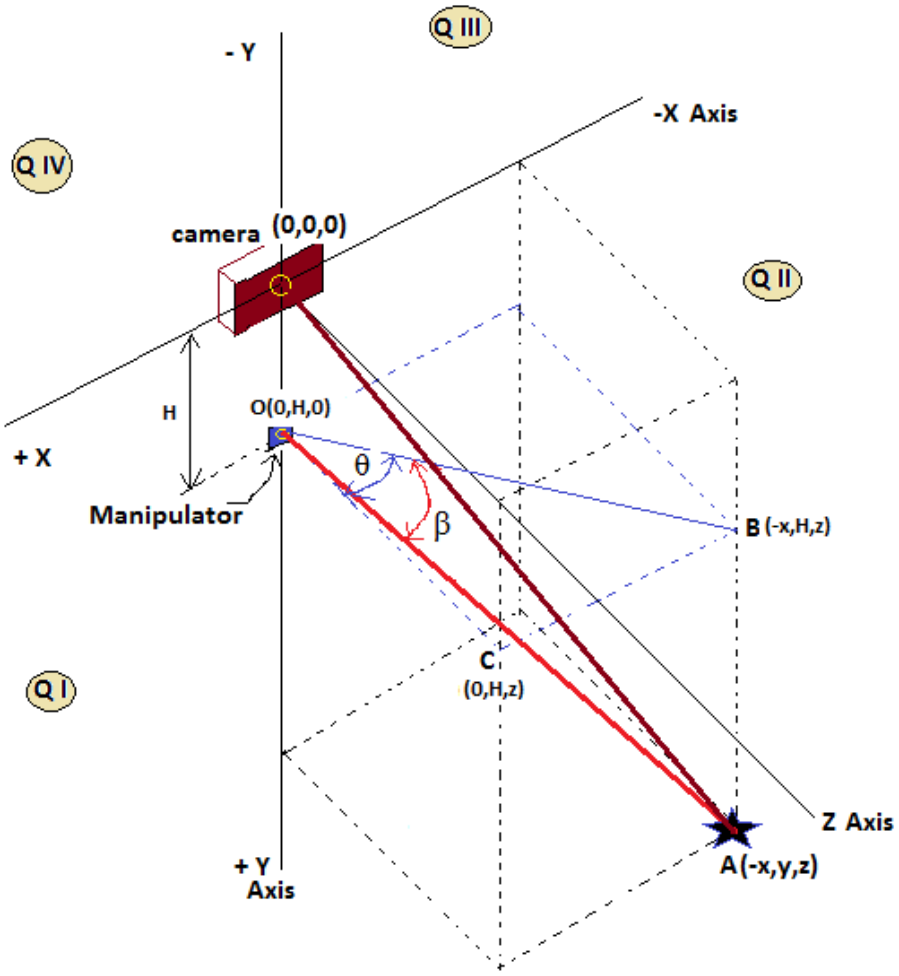


Figure 4.10 Illustration of location of the camera, manipulator and the object in 3D space

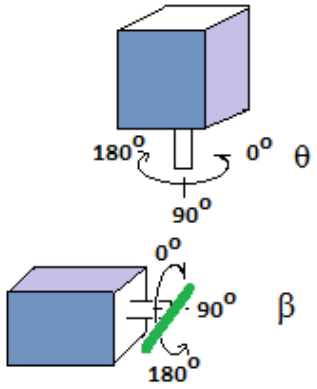


Figure 4.11 Rotation direction and reference angle of the servo motors

4.6. Graphical User Interface

For easy interaction between a human and the Ceilbot, a graphical user interface (GUI) is developed. This GUI enables the user to have full control of the Ceilbot and helps monitor important parameters such as position information of the Ceilbot and servomotor angle values. The GUI, therefore, simplifies the interaction between the user and the Ceilbot. This is formally known as Human Robot Interaction (HRI).

The user interface allows the user to operate the Ceilbot in two modes of operation: fully manual mode and semi-autonomous mode. In the manual mode of operation, the user can easily change the position of the Ceilbot, the camera tilt angle and the manipulator servomotor angles. As a feedback information for the user, live video is displayed along with other status information as shown in Figure 4.12. Similarly, for the autonomous operation mode, the user can select commands from the list and confirms by clicking the start button. The Ceilbot executes the selected commands one by one and informs the user when the execution is completed. The status information is also displayed on the same window as shown in Figure 4.12.

Calibrating the Ceilbot is the other feature included in the Ceilbot application. It can be done manually or automatically. Manual calibration is as simple as clicking the 'Calibrate' button after selecting the 'calibrate' option on the GUI. Automatic calibration is triggered when the total distance traveled exceeds the specified threshold distance in the 'General Settings' tab. Calibration, in the Ceilbot application, resets the position error accumulated due to encoder errors, puts the Ceilbot at the reference position and resets the values of the variables used in the motor controller code.

The GUI also provides options for configuring the values of parameters of the motor controller, object recognition algorithm and automatic calibration. Furthermore, the GUI provides a logging view feature to display the data published by all the ROS nodes described in section 4.8. Snapshot of the settings and logging info tabs can be seen in appendix A.

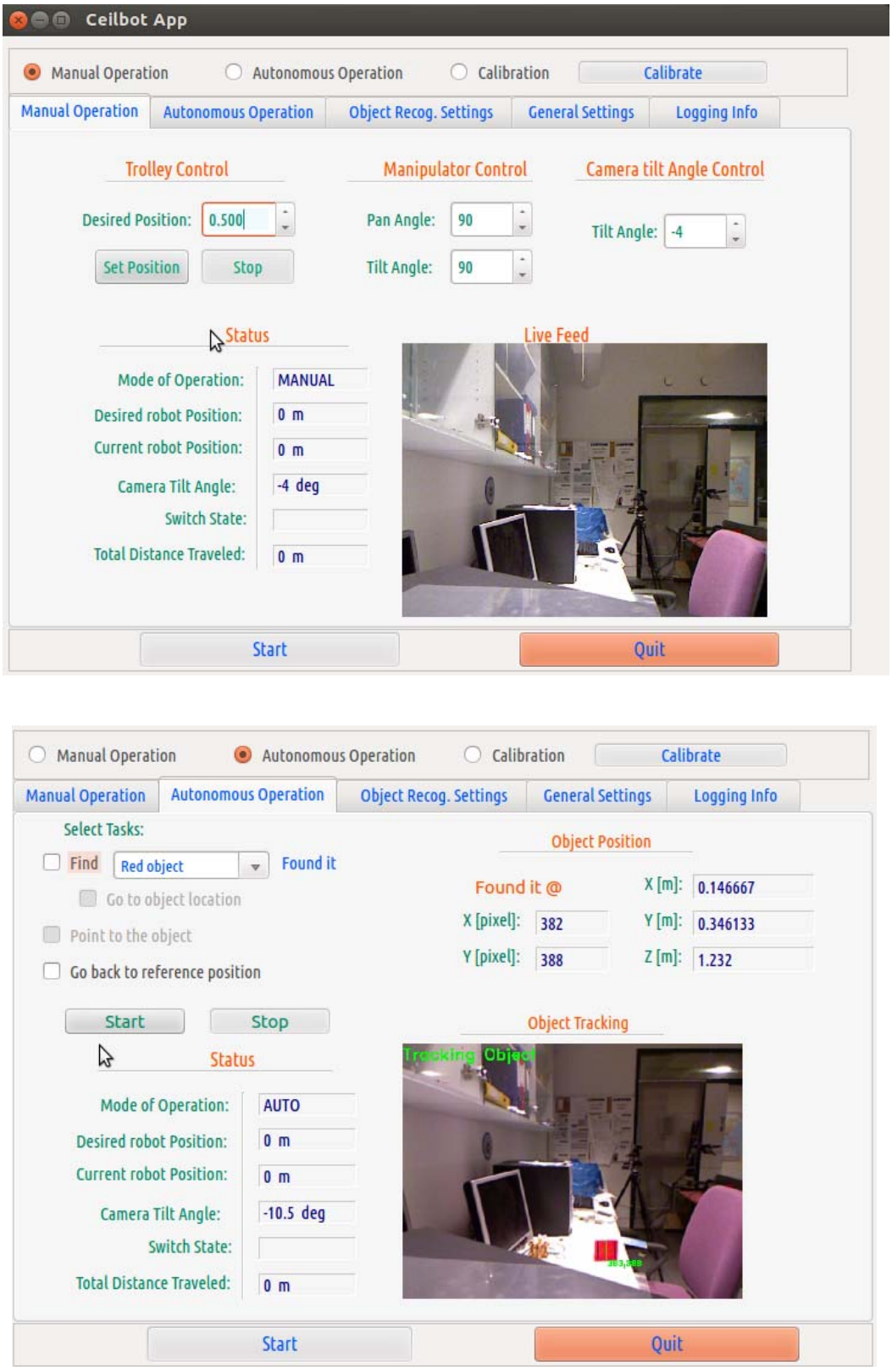


Figure 4.12 The GUI (Top) Manual mode (bottom) Autonomous mode

4.7. Communication with the Ceilbot computer

For convenient operation of the Ceilbot, there should be a wireless link between the user computer (desktop or laptop) and the Ceilbot computer. This wireless link enables the user to send commands to the Ceilbot and get feedback information from the Ceilbot computer.

Commonly used wireless communication mechanisms for other applications such as radio, Bluetooth and Wi-Fi networks can be implemented for the Ceilbot application. In the Ceilbot system, however, the data to be transferred from the Ceilbot computer to the user computer involves large size color images and point cloud data. Transferring large data through the Bluetooth link or radio link may not be the viable option. Therefore, Wi-Fi network is chosen for our application.

The developed code is tested over the Wi-Fi network. Commands from the user computer to the Ceilbot computer has been successfully transmitted and status feedback information from Ceilbot computer to the user desktop computer is also successful. However, there seems to be a problem in transferring the color image data and its corresponding point cloud data from the Ceilbot computer to the user computer. The problem is due to low data transfer rate of the Wi-Fi network. The same code is also tested on a wired network just to test if the system is actually functioning properly. Test result shows that proper communication is possible when wired network is used confirming that the data rate of the Wi-Fi network used for the testing of the system is indeed low.

4.8. Software Implementation

Software implementation structure of the Ceilbot system is shown in Figure 4.13. Each block in the figure represent software modules or ROS nodes and arrows indicate the direction of data transfer. Brief description of each node is presented as follows.

4.8.1. Trolley Controller

This node implements the modified proportional controller presented in section 4.3. It receives instructions and motor control parameters from the QNode node and motor position feedback information from the ArduinoNode node as illustrated in Figure 4.14. It also provides status information such as the total distance the Ceilbot traveled, desired position received to the QNode node.

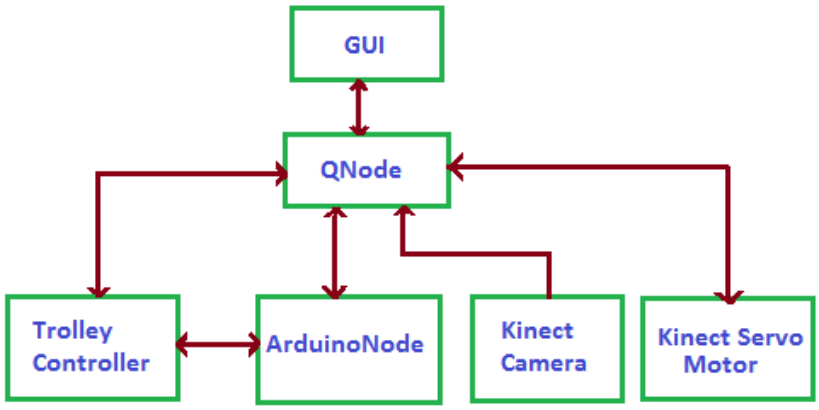


Figure 4.13 Software implementation of nodes and their interaction

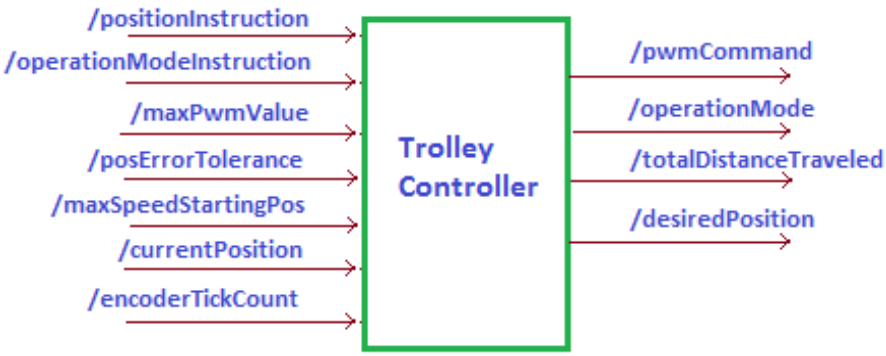


Figure 4.14 Trolley Controller node subscribed and published topics

4.8.2. ArduinoNode

This node is implemented on the Arduino Mega2560 board and communicates to the Trolley Controller and QNode nodes through the USB port. Its two main responsibilities are translating the commands coming from the QNode and the Trolley Controller nodes and sending feedback and status information back to the nodes. Translated commands are often driving the manipulator servo motors and the DC motor at a given speed depending on the mode of operation (calibration mode or normal operation mode). The topics subscribed and published by this node is depicted in Figure 4.15.

4.8.3. QNode

This node implements the object recognition and position determination algorithm, manipulator joint angles (servomotor angles) calculation function and provides an interface

between the GUI and other nodes. In addition, analyzing the user input coming through the GUI, generating commands to the ArduinoNode and Trolley Controller nodes and updating GUI display parameters are the other roles of this node. Figure 4.16 shows the topics subscribed and published by this node.

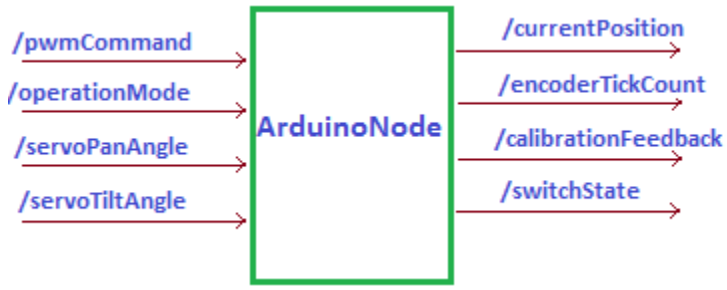


Figure 4.15 ArduinoNode node subscribed and published topics

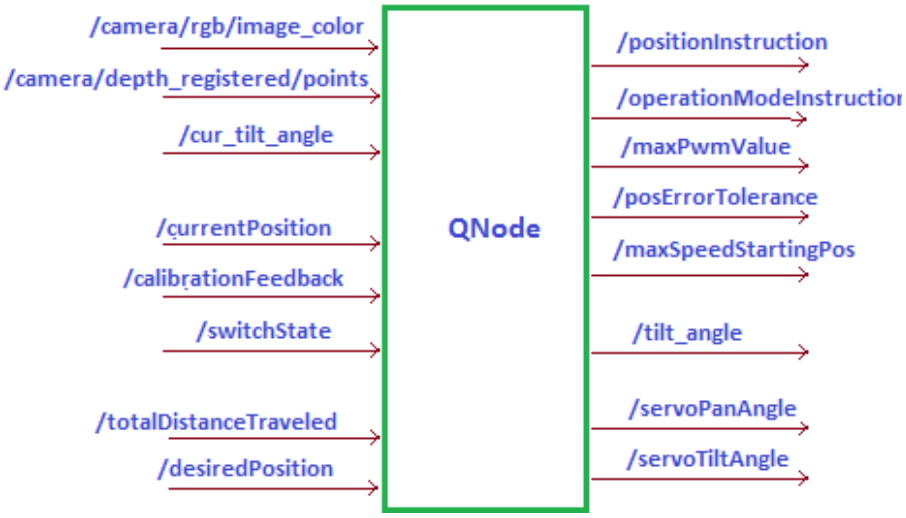


Figure 4.16 QNode node subscribed and published topics

4.8.4. Kinect camera node and Servomotor

The camera node is readily available node in the ROS package which publishes large number of topics including raw image data, point cloud data, camera information, calibrated/processed image and point cloud data. For the Ceilbot application, only two topics are subscribed from this node as shown in Figure 4.16.

The Kinect servomotor node is an independent node that publishes the current tilt angle of the camera and other topics. It also allows tilt angle adjustment through the subscribed topic 'tilt_angle'. Although the camera and the servomotor are on the same physical device, the nodes are independent.

5. Results of Ceilbot Implementation

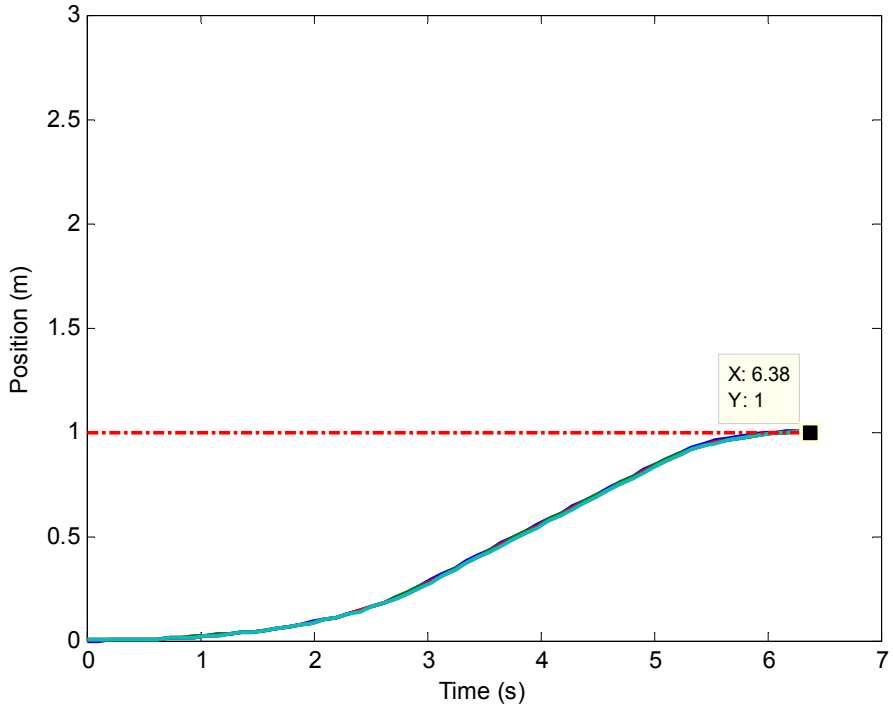
In this section, the results of the motor controller and object recognition algorithms that are discussed in the previous sections are presented.

5.1. Position Control

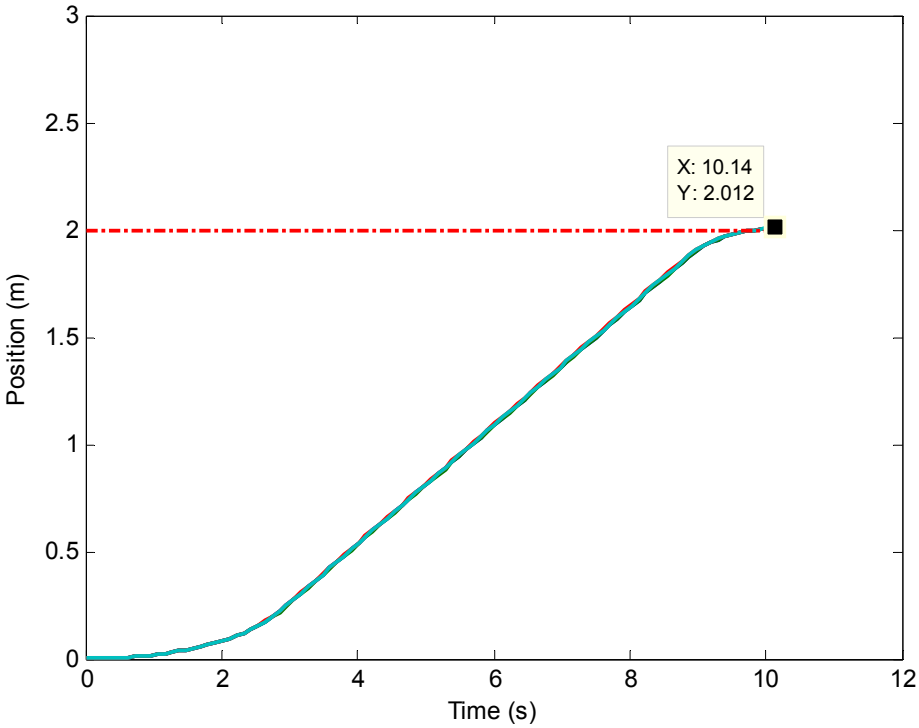
To evaluate the performance of the position control mechanism presented in section 4 above, three different desired positions are considered. The desired positions are measured (calculated) with respect to the reference position. The Ceilbot track length used for the testing is 3 meters long and the desired positions (set-points) used for the experiment are 1 m, 2 m, and 2.75 m. The Ceilbot has been tested more than 4 times for each of the test positions to verify the repeatability of the implemented controller and the resulting data is tabulated in Table 5.1. The results show that the position inaccuracy is always less than $0.5cm$ for all of the test positions. The plots in Figure 5.1 show the position controller response curve for the three test positions (only 4 measurements are considered for each distance). The controller results in a response curve without overshoots (oscillations) most often and very rarely with a small overshoot.

Table 5.1 The motor controller response for the three test positions

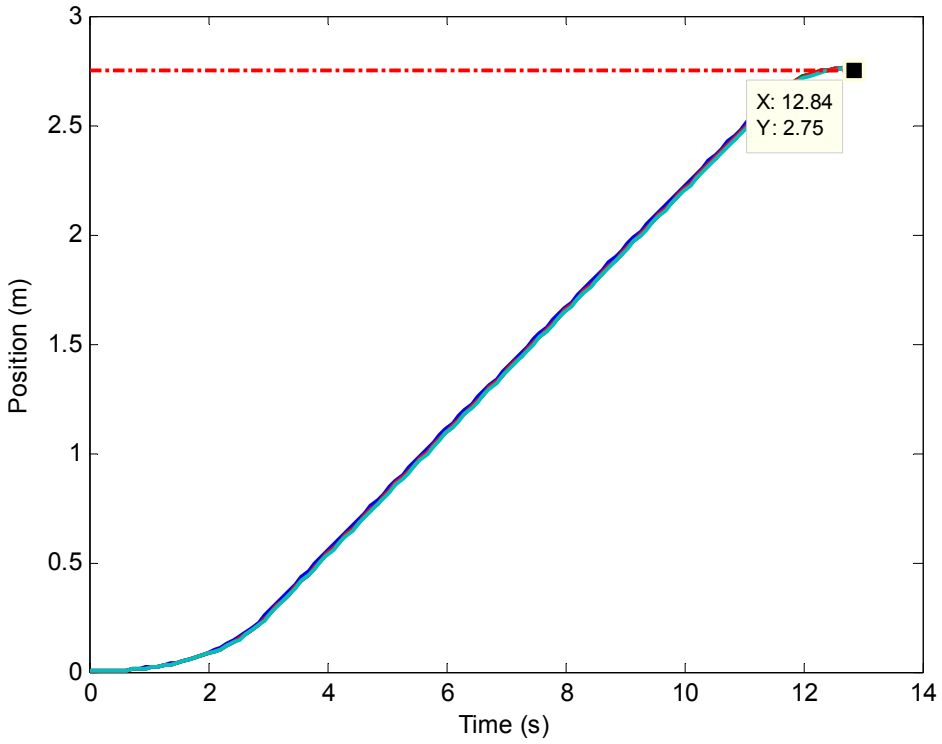
1 m		2 m		2.75 m	
Controller output	|Error|	Controller output	|Error|	Controller output	|Error|
1.00264	0.00047	1.99821	0.00179	2.74856	0.00144
0.99967	0.00033	1.99973	0.00027	2.74804	0.00196
0.99900	0.00100	1.99889	0.00111	2.74792	0.00208
0.99867	0.00133	2.00099	0.00099	2.74744	0.00256
0.99796	0.00204	2.00027	0.00027	2.74728	0.00272
0.99703	0.00297	2.00116	0.00116	2.74815	0.00185



(a) Desired position 1 m



(b) Desired position 2 m



(c) Desired position 2.75 m

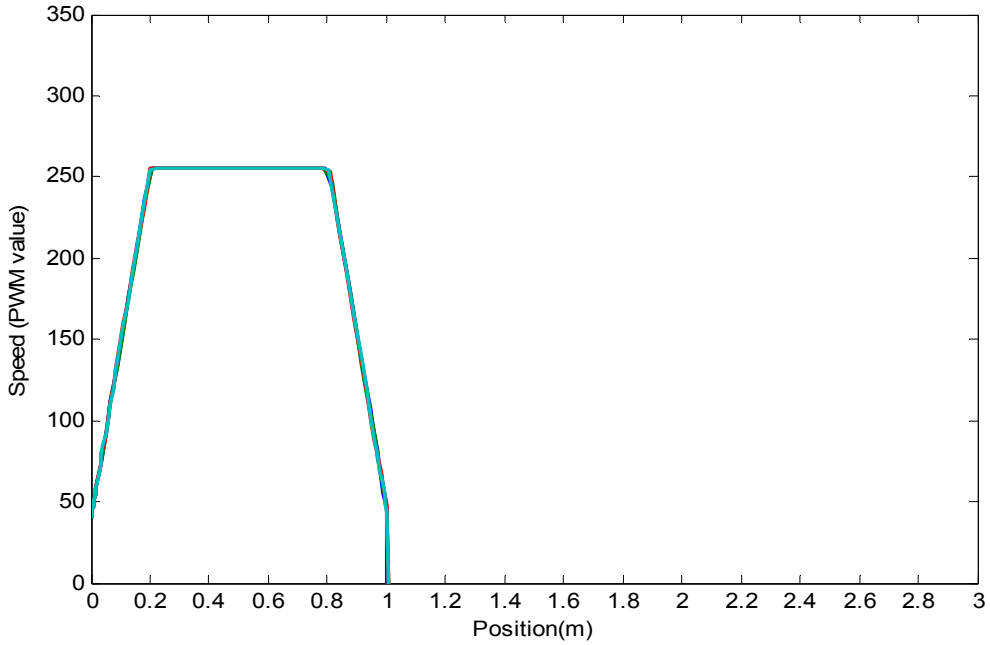
Figure 5.1 The motor controller position response curve for three desired positions: 1 m, 2 m and 2.75 m from the reference position.

5.2. Speed control

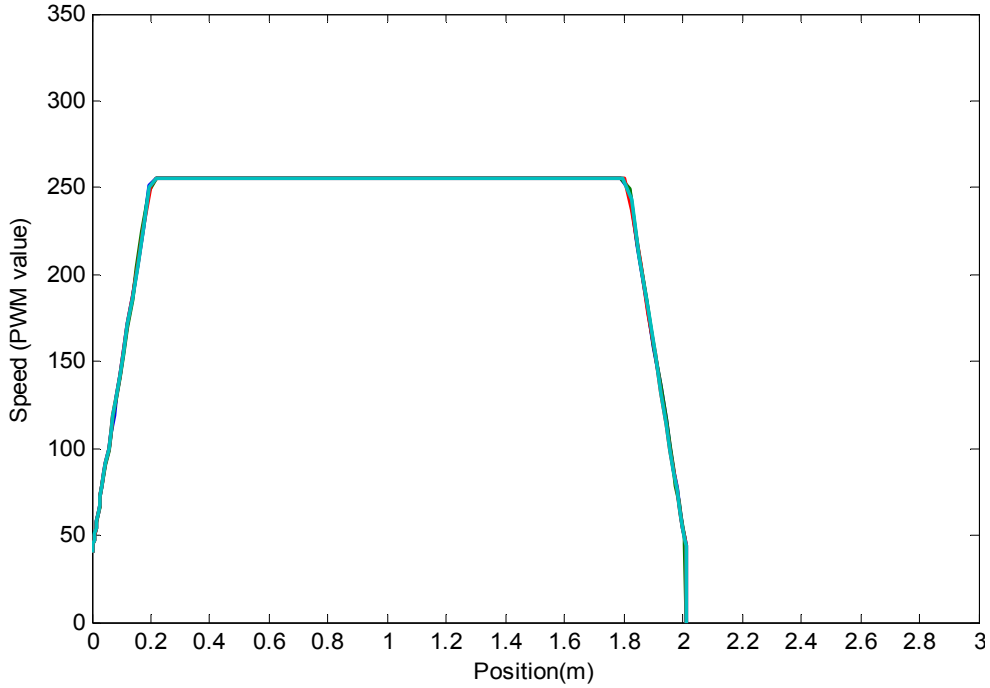
In the same settings presented in the previous section, the speed response of the proposed controller is also evaluated. As discussed in section 4 the speed of the Ceilbot is configured to follow trapezium shaped curve to make sure that the robot starts and stops smoothly to ease on the mechanical systems and power supply. In addition, the same operation produces better position accuracy. The corresponding speed plots for the three test positions described in the previous section are shown in Figure 5.2. The plots show the speed of the motor in terms of Pulse Width Modulation (PWM) values. This PWM value simply indicates the fraction of power the motor is being provided. The maximum PWM (255) corresponds to the maximum or full power the motor gets and minimum PWM (0) corresponds to zero power supply. Therefore, higher PWM means that the speed of the motor is higher.

There are three segments in the speed plots as discussed in section 4. The Ceilbot accelerates for the first $d = 20\,cm$ and decelerates the last $d = 20\,cm$ at the same rate. For both the

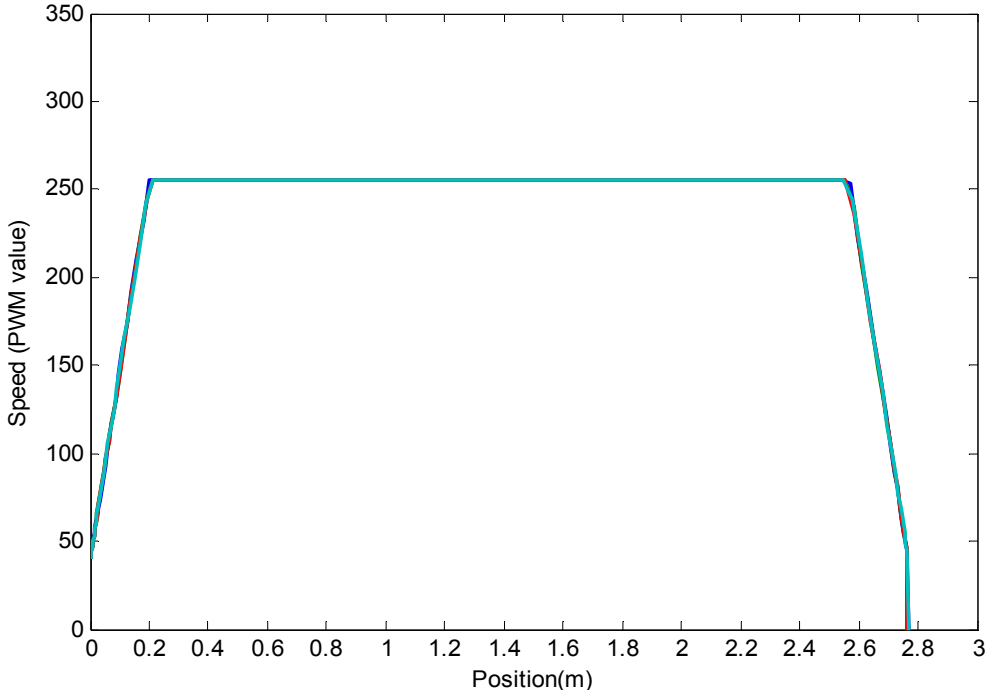
acceleration and deceleration segments the Ceilbot takes about 2.7 seconds each at 24 V power supply without carrying any load. For the constant speed segment, however, the speed is about 27.5 cm/s.



(a) Speed curve when desired position is 1 m



(b) Speed curve when desired position is 2 m



(c) Speed curve when desired position is 2.75 m

Figure 5.2 The speed controller response curve for speed of the Ceilbot for the three test positions

5.3. Object recognition and position determination

The object recognition algorithm described in section 4.3 performs very well for object colors that are different from the background. For this algorithm to work properly the illumination level over the object should not change much.

The color based object recognition algorithm is tested using three objects. The objects are yellow cup, orange multimeter, and green file cabinet. HSV values are adjusted to filter out the background to get required objects. The effect can be seen on the tracking window. The HSV values can be saved for each object for later use. The algorithm performs very well for all objects regardless of their orientation, size of the object and view angle as shown in figure 5.3.

The position determination, using the point cloud library (PCL), also works very well specially for relatively larger objects. The only problem with this part of the algorithm is that when the size of the object shrinks, due to large distance between camera and object, the algorithm sometimes misses the center point and returns wrong position information. Figure

5.4 shows the results of the object recognition and position determination algorithms for the objects mentioned above.

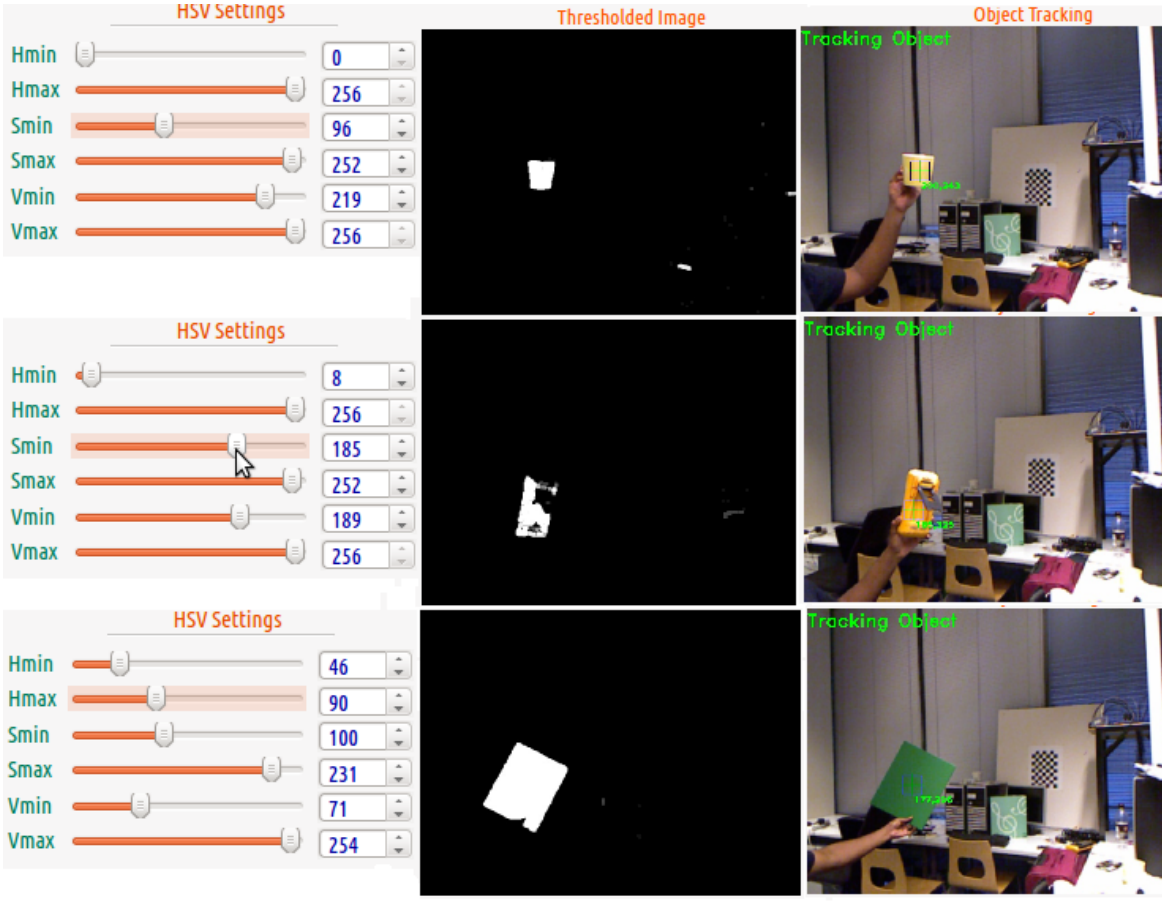


Figure 5.3 Object recognition using color based recognition algorithm described in section 4

5.4. Integration of object position determination and motor controller

Unlike the manual operation mode which gets the set-point (position) from the user, the semi-autonomous mode of operation of the Ceilbot gets the set-point or desired position of the object from the object recognition and position determination algorithm. Object recognition algorithm determines the x- and y-coordinates of the object in pixels and from the point cloud data, the 3D location of the object corresponding to the x and y pixel coordinates is extracted. The manipulator servo angles are also determined from the 3D location

information. These values are used by the motor position controller to drive the trolley (motor) to the desired position. This operation has been tasted and performance is very good.

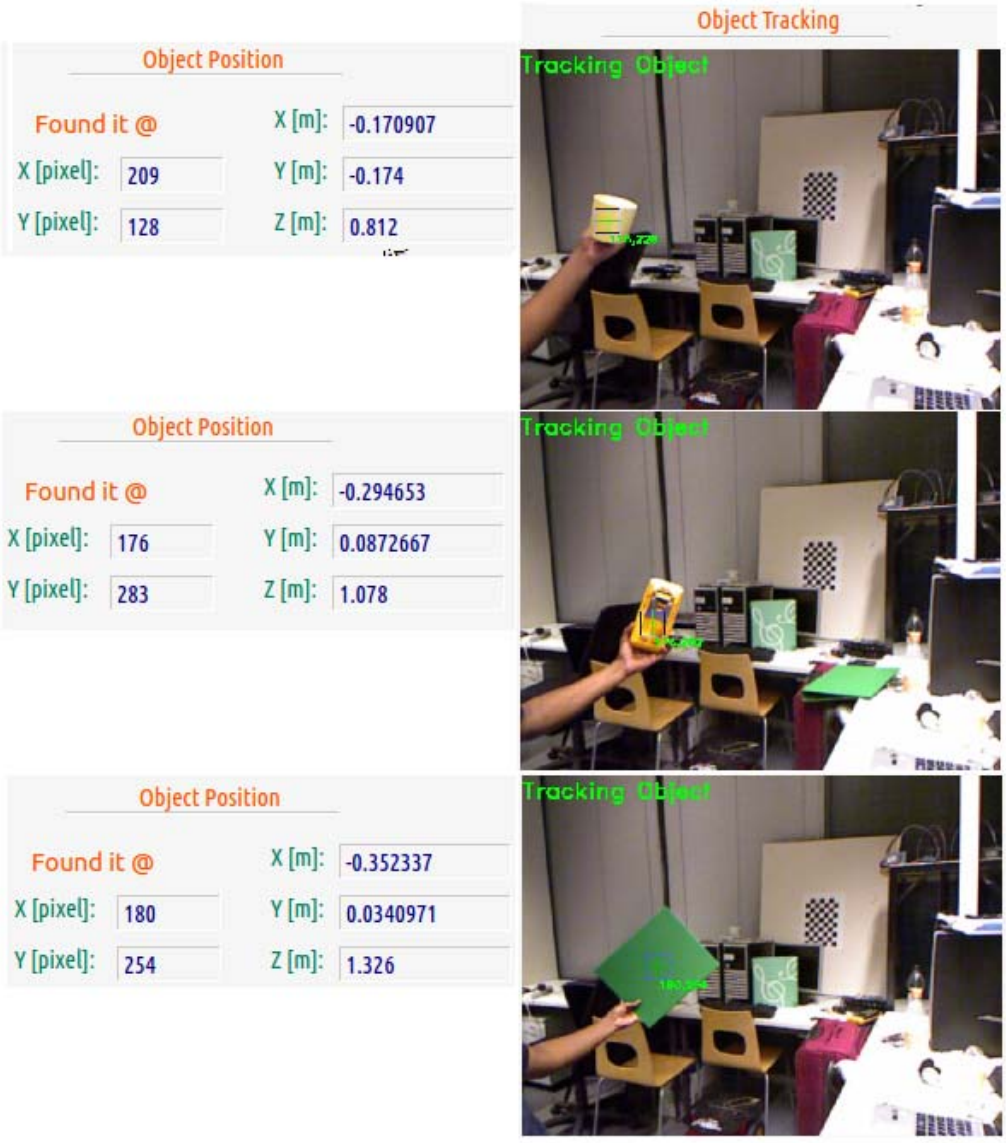


Figure 5.4 Object recognition and position determination using color based image recognition and point cloud library (PCL)

5.5. Manipulator performance

The manipulator described in section 4 consists of two servomotors with a laser pointer connected to the lower servo used as an end-effector. The manipulation is simply pointing to

the object using the laser pointer. This manipulator is intended to be used for demonstration purposes only. Hence, enough attention has **not** been given for its construction and integration with Ceilbot. Nevertheless, in the manual mode of operation, controlling the manipulator is straight forward as the servomotor angles are manually provided by the user through the GUI. In the semi-autonomous mode, however, the manipulator servomotor angles are determined based on 3D location information of the object. This integration works reasonably well if the camera is fixed at one particular tilt angle (-31 degree). If the camera tilt angle changes, then the angle calculation will be erroneous since the calculation always assumes that the camera and the end-effector are on the same plane.

6. Conclusions

In this thesis, the development of a ceiling mounted robot called Ceilbot has been presented. The robot uses a track attached to the ceiling for its operation. The operating environment of the robot by itself allows it to access a continuous power supply and minimizes or eliminates obstacles on its way which is unlike any other mobile robot operating on the ground. A list of possible applications have also been covered. As part of the development, a modified proportional controller is implemented for the robot position and speed control. Measured data shows that the controller provides very good positioning of the Ceilbot with high level of repeatability and inaccuracies of less than 0.5 cm. In addition, the controller produces trapezium shaped speed curve that involves acceleration, constant (maximum) speed and deceleration with flexible control of how long the robot needs to accelerate to reach the maximum speed and decelerate to arrive at the required position. Furthermore, the controller keeps track of the current position of the Ceilbot at all times with respect to the reference position.

The developed Ceilbot application offers manual mode of operation as well as semi-autonomous operations. In the manual mode of operation, the user operating the Ceilbot provides the set-point position, the camera tilt angle value, the servomotor angle values and other parameters through the GUI. In the semi-autonomous operation mode, the Kinect camera is used to capture color image and point cloud data of the scene. The object recognition algorithm processes the image and point cloud data to recognize objects of interest based on color information and determines the 3D location of the object with respect to the camera, then the set-point (desired position) of the Ceilbot with respect to the reference position is calculated by taking into account the current position of the Ceilbot. Thus, given the desired position, the Ceilbot then approaches the object for manipulation. The color based object recognition algorithm is chosen over other algorithms for its simplicity to implement and its good performance regardless of size, orientation (rotation), partial occlusion and view angle variations, although it is sensitive to lighting (illumination) variations.

A graphical user interface (GUI) is implemented to make interaction between the user and the Ceilbot easier. Sending commands to the Ceilbot, modifying controller and image processing parameter settings and displaying status information about the Ceilbot for monitoring are the major functionalities provided by the GUI. A simple manipulator is also built using two

servomotors to demonstrate a completely integrated robot system-Ceilbot. The manipulation is simply pointing to the object using the laser pointer. The angles of the servomotors are determined from the 3D location information of the object.

Recommendations for future work

Despite the good performance of the color based object recognition algorithm against the variation of some parameters, the sensitivity to illumination makes it almost unusable. Therefore, a better recognition algorithm needs to be developed to make the Ceilbot more effective in the (semi-) autonomous mode operation. From the comparative studies of object recognition algorithms, it is known that no single algorithm performs well when there are changes in object size, orientation and other variations. Hence, application should be picked first and then algorithms should be chosen based on the specific application of the Ceilbot.

Object location determination only using the Kinect depth sensor may not be good enough for practical applications as the Kinect depth sensor has a limited range (~ 4 m) and its accuracy degrades with distance. Therefore, other depth or distance sensors like ultrasonic or laser should be experimented if integration with the image processing is possible.

References

[1]. Oliver Prenzel, "Process model for the development of Semi-autonomous service robots", PhD dissertation, University of Bremen, Germany, 2009.

[2]. Graefe, V.; Bischoff, R., "Past, present and future of intelligent robots," *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium on* , vol.2, no., pp.801,810 vol.2, 16-20 July 2003

[3]. Jarvis, Ray. "Intelligent Robotics: Past, Present and Future." *IJCSA* 5.3 (2008): 23-35.

[4]. "Vacuum Cleaning robot", [Online]. Available: http://www.witt-ltd.com/iRobot/Vacuum_Cleaning_Robots/Roomba_780 [Accessed: Mar 02, 2014].

[5]. "Executive summary of Industrial robots and service robots", [Online]. Available: www.worldrobotics.org/uploads/tx_zeifr/Executive_Summary_WR_2013.pdf. Accessed: Mar 02, 2014].

[6]. Andersson, Magnus, et al. "ISR: An intelligent service robot." *Sensor Based Intelligent Robots*. Springer Berlin Heidelberg, 1999. 287-310.

[7]. Frank Tobe. (Nov 12, 2012). "Where are the Elder Care Robots?" [Online]. Available: http://spectrum.ieee.org/automaton/robotics/home-robots/where-are-the-eldercare-robots. [Mar 02, 2014].

[8]. "Ceilbot, background", [Online]. Available: http://autsys.aalto.fi/en/Ceilbot. [Accessed: Mar 02, 2014].

[9]. "SmartTrack, Mobile robotic surveillance system", [Online]. Available: http://sentrytechnology.com/downloads/SmartTrack%20Cut%20Sheet3.pdf. [Accessed: Mar 02, 2014].

[10]. "Boots on the right track with SmartTrack", [Online]. Available: http://www.cctv-information.co.uk/i/Boots_on_the_Right_Track_with_SmartTrack. [Accessed: Mar 02, 2014].

[11]. "AIPCT: Advanced IP Camera Track", [Online]. Available: http://www.revolutionary-robotics.com/products/aipct.html. [Accessed: Mar 02, 2014].

[12]. "Camera Track System", [Online]. Available: http://www.telemetricsinc.com/products/camera-robotics/cts-camera-track-system/cts-camera-track-system/camera-track-system. [Accessed: Mar 02, 2014].

[13]. "Vector elite, Clinician's guide", [Online]. Available: http://www.bioness.com/Documents/Clinicians%20EliteMp612-00624-001_RevB[e-file].pdf. [Accessed: Mar 02, 2014].

[14]. "ZeroG Gait and Balance training system", [Online]. Available: http://www.aretechllc.com/ZeroG_Brochure.pdf. [Accessed: Mar 02, 2014].

[15]. Stepan, Gabor, et al. "ACROBOTER: a ceiling based crawling, hoisting and swinging service robot platform." *Beyond Gray Droids: Domestic Robot Design for the 21st Century Workshop at HCI*. Vol. 2009. No. 3. 2009.

[16]. "Companies making the necessary transition from industrial to service robots", [Online]. Available: http://singularityhub.com/2012/06/06/companies-making-the-necessary-transition-from-industrial-to-service-robots/. [Accessed: Mar 02, 2014].

[17]. Thrun, Sebastian, et al. "MINERVA: A second-generation museum tour-guide robot." *Robotics and automation, 1999. Proceedings. 1999 IEEE international conference on*. Vol. 3. IEEE, 1999.

[18]. Sato, Tomomasa, et al. "Construction of ceiling adsorbed mobile robots platform utilizing permanent magnet inductive traction method." *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 1. IEEE, 2004.

[19]. Fukui, Rui, et al. "HangBot: A ceiling mobile robot with robust locomotion under a large payload (Key mechanisms integration and performance experiments)."*Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011.

[20]. "Autonomous Ceiling Robot- Contactless Energy Transfer", [Online]. Available: http://www.tue.nl/universiteit/faculteiten/faculteit-electrical-engineering/onderzoek/onderzoeksprogrammas/electromechanics-and-power-electronics-epe/research/projects/precision/autonomous-ceiling-robots/. [Accessed: Mar 02, 2014].

[21]. "Ceilbot previous work at Aalto", [Online]. Available: http://autsys.aalto.fi/en/Ceilbot/PreviousWork. [Accessed: Mar 02, 2014].

[22]. "Software Overview", [Online]. Available: http://www.willowgarage.com/pages/software/overview [Accessed: July 2014]

[23]. Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." *ICRA workshop on open source software*. Vol. 3. No. 3.2. 2009.

[24]. "ROS tutorial: Creating a ROS msg and srv", [Online]. Available: http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv [Accessed: July 2014].

[25]. "Qt cross-platform application and UI framework…", [Online]. Available: https://qt-project.org/ [Accessed: July 2014].

[26]. "PID controller", [Online]. Available: http://en.wikipedia.org/wiki/PID_controller. [Accessed: July 2014].

[27]. "Outline of object recognition", [Online]. Available: http://en.wikipedia.org/wiki/Outline_of_object_recognition. [Accessed: July 2014].

[28]. Kim, Hae Yong, and Sidnei Alves De Araújo. "Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast." *Advances in Image and Video Technology*. Springer Berlin Heidelberg, 2007. 100-113.

[29]. Wu, Jian, et al. "A Comparative Study of SIFT and its Variants." *Measurement Science Review* 13.3 (2013): 122-131.

[30]. Juan, Luo, and Oubong Gwun. "A comparison of sift, pca-sift and surf." *International Journal of Image Processing (IJIP)* 3.4 (2009): 143-152.

[31]. Gevers, Theo, and Arnold WM Smeulders. "Color-based object recognition." *Pattern recognition* 32.3 (1999): 453-464.

Appendix A

Graphical User Interface screen shots

1) Manual operation mode

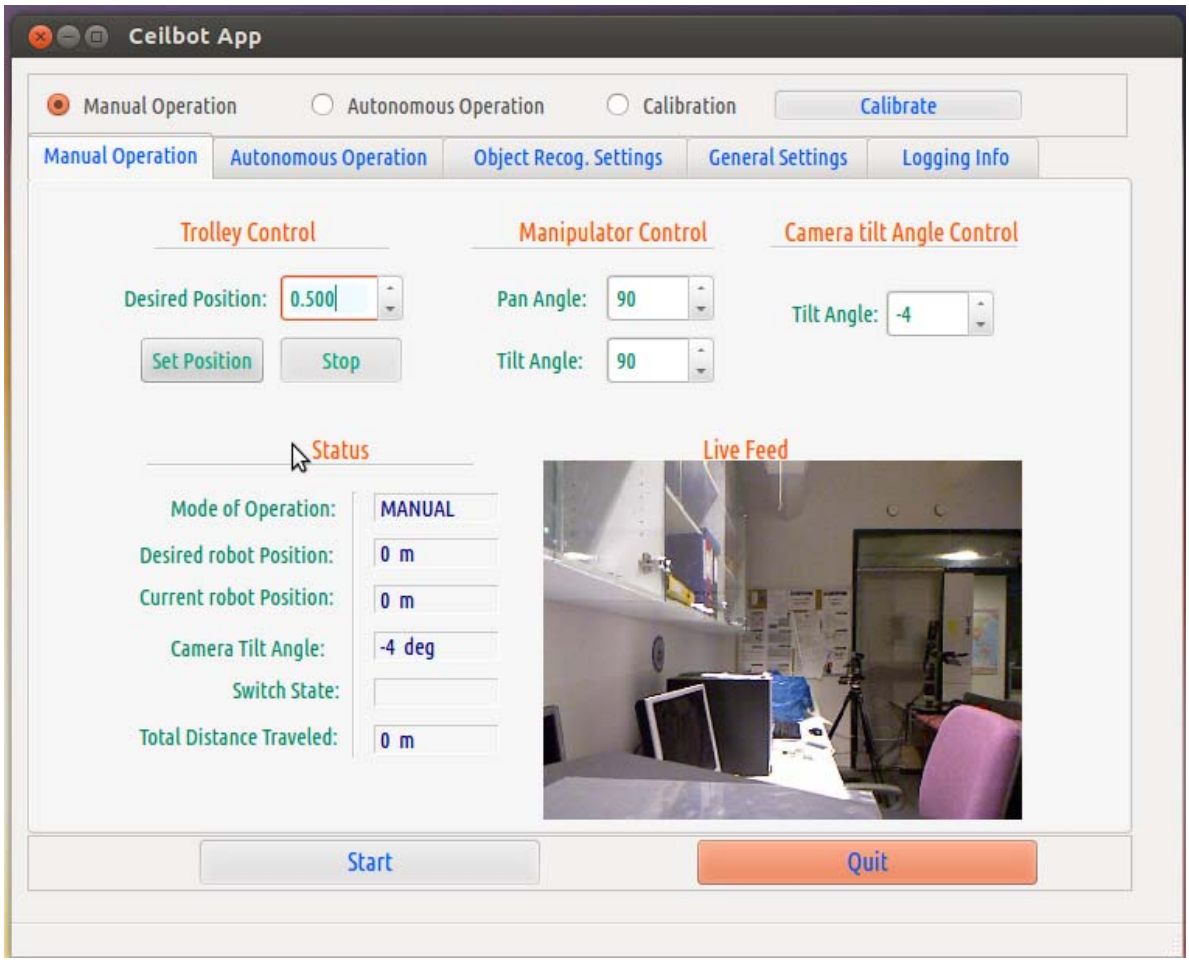


Figure A1. Manual Operation mode tab.

2) Autonomous operation mode

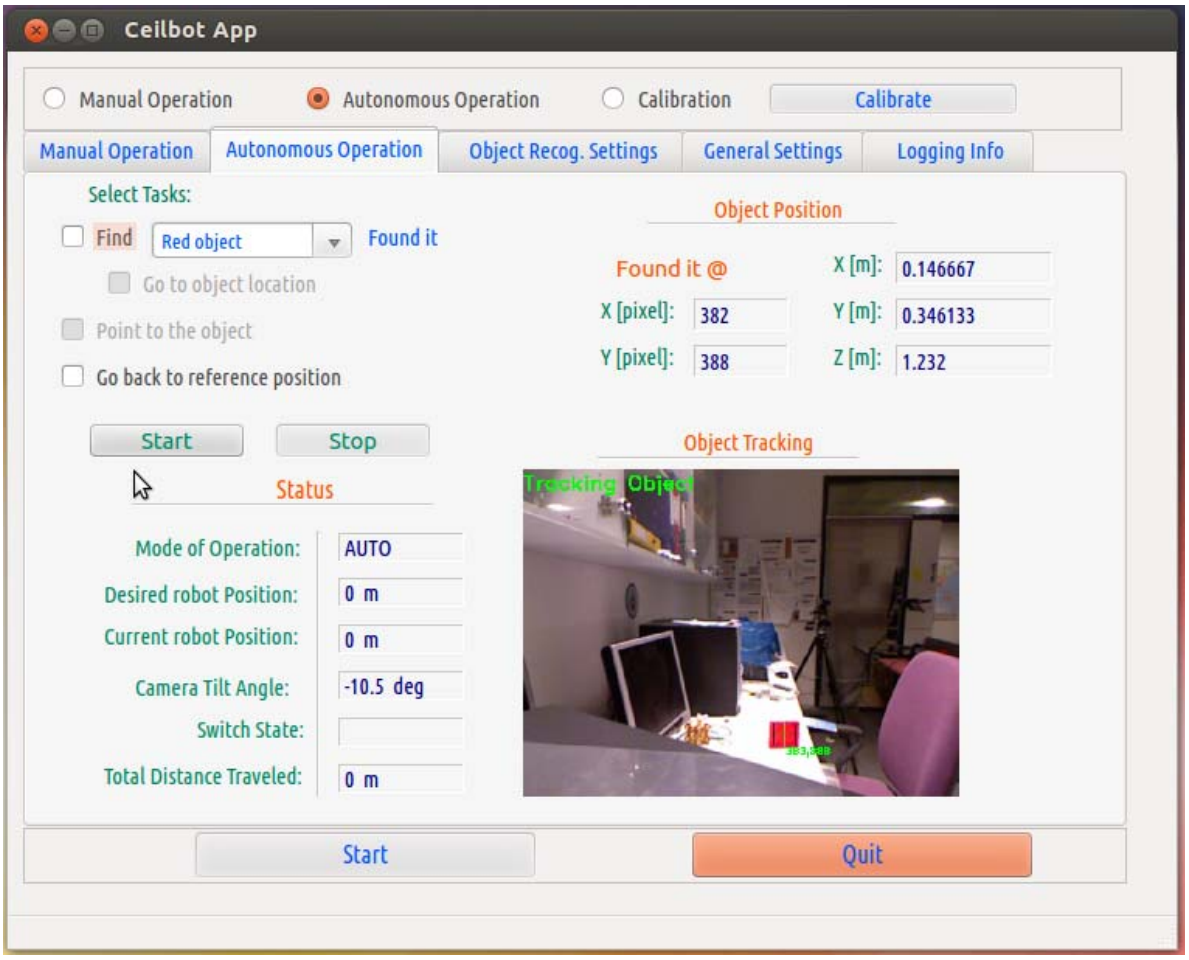


Figure A2. Autonomous operation mode tab

3) Object recognition settings

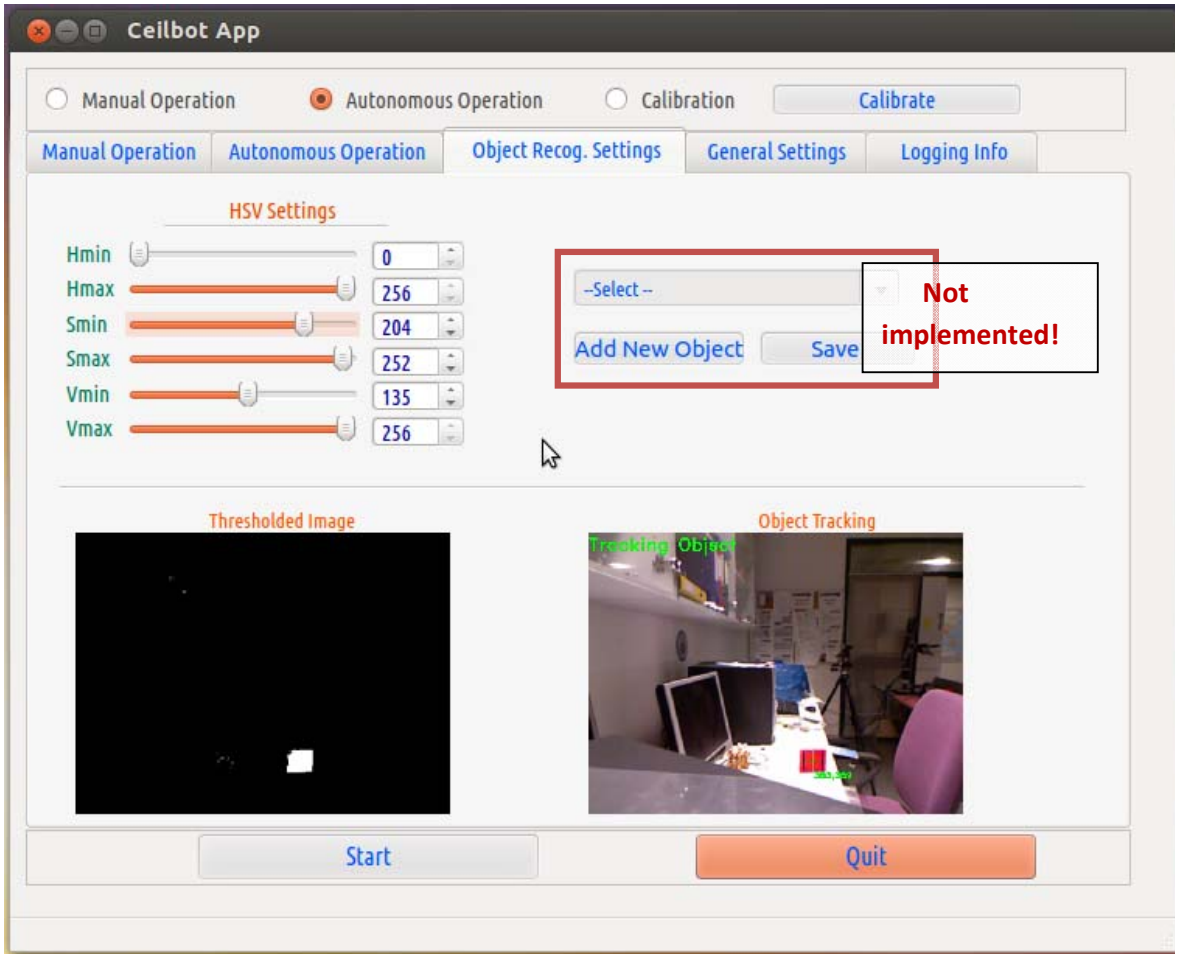


Figure A3. Object recognition HSV settings tab

4) General settings

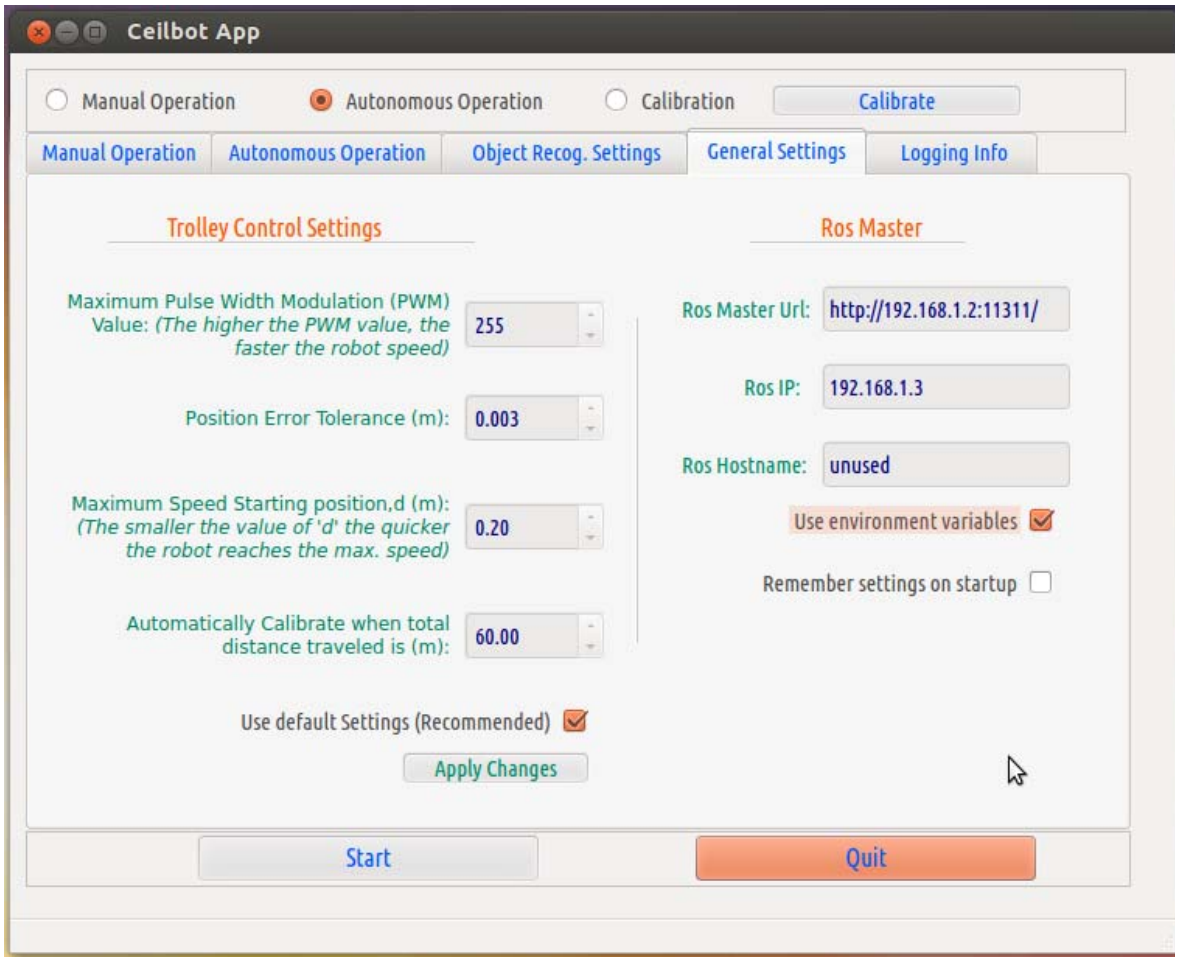


Figure A4. General settings tab

5) Logging Info

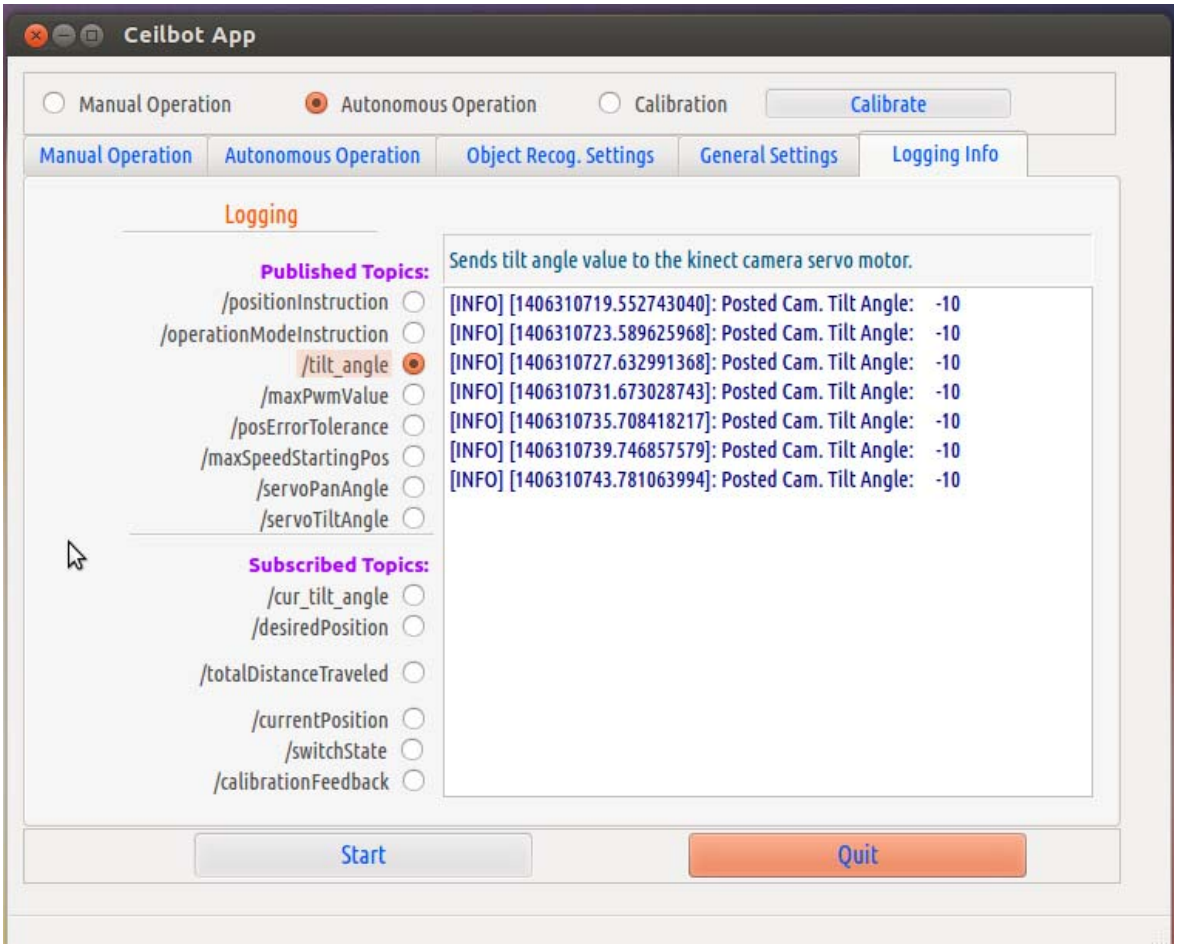


Figure A5. Logging Info tab