Master Thesis
Computer Science
Thesis no: MCS-2010-22
May 2010

A Vision and Differential Steering System for a Mobile Robot Platform

Abujawad Rafid Siddiqui

School of Computing
Blekinge Institute of Technology
Box 520
SE – 372 25 Ronneby
Sweden

Contact Information:
Author(s):
Abujawad, Rafid, Siddiqui
Address: Folkparksvagen 22:14
E-mail: rafid_2k3@hotmail.com

University advisor(s):
Prof.Craig Lindley
School of Computing

Table of Contents

ABSTRACT

Context: Effective vision processing is an important study area for mobile robots which use vision to detect objects. The problem of detecting small sized coloured objects (e.g. Lego bricks) with no texture information can be solved using either colour or contours of the objects. The shape of such objects doesn't help much in detecting the objects due to the poor quality of the picture and small size of the object in the image. In such cases it is seen that the use of hybrid techniques can benefit the overall detection of objects, especially, combining keypoint based methods with the colour based techniques.

Robotic motion also plays a vital role in the completion of autonomous tasks. Mobile robots have different configurations for locomotion. The most important system is differential steering because of its application in sensitive areas like military tanks and security robot platforms. The kinematic design of a robotic platform is usually based on the number of wheels and their movement. There can be several configurations of wheels designs, for example differential drives, car-like designs, omni-directional, and synchro drives. Differential drive systems use speed on individual channels to determine the combined speed and trajectory of the robot. Accurate movement of the robot is very important for correct completion of its activities.

Objectives: A vision solution is developed that is capable of detecting small sized colour objects in the environment. This has also been compared with other shape detection techniques for performance evaluation. The effect of distance on detection is also investigated for the participating techniques.

The precise motion of a four-wheel differential drive system is investigated. The target robot platform uses a differential drive steering system and the main focus of this study is accurate position and orientation control based upon sensor data.

Methods: For object detection, a novel hybrid method 'HistSURF' is proposed and is compared with other vision processing techniques. This method combines the results of colour histogram comparison and detection by the SURF algorithm. A solution for differential steering using a Gyro for the rotational speed measurement is compared with a solution using a speed model and control outputs without feedback (i.e. dead reckoning).

Results: The results from the vision experiment rank the new proposed method highest among the other participating techniques. The distance experiment indicates that there is a direct and inverse relation between the distance and detected SURF features. It is also indicated by the results that distance affects the detection rate of the new proposed technique. In case of robot control, the differential drive solution using a speed model has less error rate than the one that uses a Gyro for angle measurement. It is also clear from the results that the greater the difference of speeds among the channels the less smooth is the angular movement.

Conclusions: The results indicate that by combining a key-point based technique with colour segmentation, the false positive rate can be reduced and hence object recognition performance improves . It has also become clear that the improved accuracy of the proposed technique is limited to small distances and its performance decreases rapidly with increase in the distance to target objects.

For robot control, the results indicate that a Gyro alone cannot improve the movement accuracy of the robotic system due to a variable drift exhibited by the Gyro while in rotation. However, a Gyro can be effective if used in combination with a magnetometer and some form of estimation mechanism like a Kalman filter. A Kalman filter can be used to correct the error in the Gyro by using the output from the magnetometer, resulting in a good estimate.

Keywords: Vision system, Mobile robot, Differential drive, Gyro measurement.

INTRODUCTION

The rising cost of healthcare and developments in the field of Robotics have increased the potential role of assistive agents in the home. In addition to providing needful assistance to handicapped people, such agents can be quite handy for the automation of small tasks. Some of these tasks can be static (in a fixed location, e.g. washing the dishes) while others can be dynamic (involving movement, e.g. automated cleaning, finding lost objects, mobile surveillance, etc.). In order to successfully achieve such tasks, in many situations a robotic agent could be greatly aided by a good visual recognition system that further requires efficient techniques for object recognition.

Apart from significant applications for mobile robots in security, health care and industrial automation, the most attractive and challenging research area is development of humanoid robots. It might seem by the human-like appearance of humanoid robots developed to date that attaining intelligence and robust locomotion is not very far away. But the challenges associated with the development of such autonomous robotic systems can only be appreciated when one engages in their development process. There are many autonomous systems that must be present in such robots in order to complete autonomous tasks. The coordination of activities among systems is also important for successful completion of desired tasks. One such system is vision processing of scenes for identification of useful segments, objects, etc. that are needed for decision-making.

In order to investigate robotic autonomous object recognition, the target objects chosen for the work reported in this thesis are small sized coloured objects (e.g. Lego bricks). In general such objects, with no texture information, can be detected using either colour or contours of the objects. The shape of such objects does not help much in detecting the objects for the following reasons:

- Poor image quality of small robot vision systems such as the one used in the experiments.
- Small size of the objects placed in the scene pose a challenging problem for efficiently acquiring the contour or edges of the objects.
- Colour segmentation results in loss of shape detail due to either noise in the image or due to variation of brightness. This in turn limits the shape extraction to outer contours of the objects that mostly consist of basic shapes for small sized objects. Since such small contours with basic structure can be found in many non-target objects in the scene, it increases the false positive rate resulting in decrease of performance. .

A good navigation system is also an important factor for successful completion of mobile robotic operations. The robotic system used in this study makes use of Arduino boards for the management of navigation functions. An embedded program inside the microcontroller on the Arduino board will be used to detect obstacles using the input from infrared (IR) and ultrasonic sensors. The controller will also perform navigation commands received from a remote computer connected through a wireless interface and processing visual images from the live video camera on the robot. These commands will be generated autonomously based on the results of the vision processing system.

The target robot has a differential drive steering system that is therefore the main focus of motion control in this study. The differential drive steering system determines the direction of movement based on the speed of its individual channels. In a four-wheel differential drive system there are usually two independent channels (right and left channels), one on each side of the robot. A forward movement can be attained by keeping equal speed on both channels, and changes in direction can be achieved by the difference in speed of the channels. The vehicle moves toward the opposite side of the channel that has higher speed as compared with the other channel.

There are therefore two contributions of this study; one is a novel hybrid object detection algorithm 'HistSURF' and its comparison with the other vision processing techniques. The second contribution is the comparison of two differential drive solutions, one using a Gyro and the other using speed modelling. The results indicate that the novel HistSURF algorithm, combining the key-point based method SURF with colour segmentation, helps in reducing the false positive rate that is the major problem with colour segmentation alone. Robotic motion experiments indicate that a Gyro alone is not helpful in reducing the overall error rate due to friction and other forces. However, careful design of a robotic system with combined measurement angle from a Gyro and magnetometer or Compass bearings can lead to better results.

CHAPTER 1: BACKGROUND

There has been a great deal of development in object recognition in past decade. The main aim of these efforts has been to detect salient structures from images and to develop discriminating feature descriptors for object matching. Object recognition requires that local features must be invariant to translations, rotation, scaling, affine transformations, occlusions and partial appearance. In the recent past, key-point based feature detection techniques [1, 2] have attained a considerable amount of success in the detection of objects in images. SIFT (Scale Invariant Feature Transform) [1] is one example having outstanding performance among this type of algorithms. It can extract distinctive invariant features from any image which can then be used to differentiate an image from other images or an object within the images from other objects. SIFT has been used with success in many areas including object detection, robot navigation, and image retrieval. Various extensions or variants to SIFT have also been proposed for performance enhancement in specific fields of study, like PCA-SIFT (speeding up using dimensionality reduction) [3], CSIFT [4] (for colour images), and a fast approach using SIFT [5]. Besides outstanding performance and some speeded up variants, applications of SIFT in real-time detection has been avoided due to slow detection rates. This problem is solved by the later technique of SURF (Speeded Up Robust Features) [2]. SURF improves SIFT by the use of integral images as proposed by *Viola and Jones* [6]. The robustness of SURF features has generated many applications for the method and it is being widely used in different domains [7-11]. Yet it is difficult to detect objects with plain structures having no texture detail, like Lego bricks, with only SURF features. In such situations, joining some region based techniques with key point based methods seems to be a preferable approach.

This study proposes a new technique, HistSURF , which is the combination of Histogram matching and SURF feature matching. With HistSURF, segmentation of the image is performed using HSL (Hue Saturation Luminanace) colour space [17]. A voting mechanism for the hybrid technique is also defined. This technique inherits the merits of both techniques and still is fast enough due to SURF being applied on a limited area of the image. The proposed scheme is evaluated by comparison with histogram matching, SURF features and shape identification.

The vision system of a mobile robot plays an important role in the effectiveness of robotic operations. There have been many efforts to find suitable methods for object detection for mobile robots [12-16]. Despite a fair amount of success in the recent past, there is need for improvement. We intend to apply HistSURF on robotic platform for autonomous object search and try to find out if it works in real-time.

Vision Related Work:

1.1 Colour Based Segmentation

The use of colour information for the detection of objects is an old method but is still a very important technique due to its use in robotic games (e.g. for the detection of balls or goal positions). Image segmentation based upon colour is performed in such robotic game playing systems for fast and inexpensive processing to detect objects [17- 19]. . A fast and inexpensive scheme for the detection of objects in interactive robots has been proposed in the study [20] which uses colour segmentation. The HSV/HSL colour space has been preferred over RGB colour space due to the fact that there is strong correlation between the three colour components (Hue, Saturation and Luminance) and also it is difficult to take distance between two points in RGB space. Moreover, changes in illumination conditions affect the RGB space more than HSV colour space. HSV colour space has been used successfully along with learning algorithms to recognize the objects in images [17, 18].

In this study, colour segmentation is performed using HSL colour space in order to partition the image into meaningful parts. These segments are then used as the input to the other selected recognition techniques (Contour matching, SURF matching, Histogram matching, and HistSURF).

1.2 Some Hybrid objects detection methods.

There have been some efforts to combine colour information with key point based techniques. Some of them use SIFT combined with colour information. Mostly a direct combination using concatenation or by some sophisticated technique is used. In [21] a normalized RGB model is combined with SIFT descriptor to get partial illumination invariance. In [22] a combination of colour and shape information was proposed to achieve performance improvement for image indexing. An extension to local features has also been proposed using colour features [23]. There has also been a merger of the SIFT features with colour invariant space [4].

There have also been attempts using an indirect or multistage combination approach. A relevant study is [24] in which colour moments are combined with SIFT descriptor. Another indirect combination is the study [25] in which a multistage approach has been used to combine colour information with SIFT features. It has been noticed that colour information and geometrical invariance often complement each other and can give better results if combined [26].

Robotic Motion Control

Mobile robots have different configurations for locomotion. Mostly common are wheeled robots that are used in research and education. There are also different types of steering control systems incorporated into them. The most relevant system to this study is a differential steering control system.

In order to solve the problem of robotic locomotion, a specific robotic system is modelled to attain desired movement. This modelling depends on the kinematic design of the mobile robot which in turn depends on the application area in which it is deployed [27]. The kinematic design of a robotic platform is usually based on the number of wheels and their movement. There are several typical configurations of wheels designs, for example differential drives [28], car-like designs [29], omni-directional [30] and synchro drives [31]. The relevant configuration is differential drive and steering system because it has been used in the *Spinosaurus* target robot that is used for the experiments in this study.

Differential drive systems use speed on individual channels to determine the combined speed and trajectory of the robot. The difference in speeds on the channels determines the rotation direction of the robot and same speed in forward or backward direction results in forward and backward motion, respectively. The rotation of wheels of robotic system is done by the use of DC motors. The speed of DC motors is dependent on the input voltage given to it. These voltages are controlled to achieve differential speed on the channels of the robot. In order to control the speed of the channels, different voltage control mechanisms are used [32]. The most relevant technique is PWM (Pulse Width Modulation). It has been used in this study to attain differential speed for faster and reliable voltage switching. For the reliable generation of PWM signals a controller system (Arduino) is used and in order to achieve voltage variations to the motors derived from the PWM signals, a motor controller is used. More detail about differential drive is explained in the methodology section.

CHAPTER 2: PROBLEM DEFINITION

Effective vision processing is an important study area for mobile robots which use vision to detect objects. It has also been an area of research due to its application in humanoid robots[19, 34, 36-38] because being like a human means a robot should have a vision processing system that is comparable to vision processing of a human. Humans learn an object by its features and can distinguish between different poses of a target object fairly easily as compared with vision system on robots. Object recognition by robots uses features like colour, texture and contour to segment an image into parts in order to learn[17-19].

The problem of detecting small sized coloured objects (e.g. Lego bricks) with no texture information can be solved using either the colour or the contour of the objects. The shape of target objects doesn't help much in detecting the objects for the following reasons:

- Poor image quality of small robotic systems such as the one used in the experimentation reported here.
- Small size of the objects placed in the scene pose a challenging problem for efficiently acquiring the contour or edges of the objects, since they are represented by few image pixels.

Colour segmentation results in loss of shape detail due to either noise in the image or due to variation of brightness. This in turn limits the shape extraction to outer contour of the objects that mostly consist of basic shapes for small sized objects. Since such small contours with basic structure can be found in many non target objects in the scene so it increases the false positive rate resulting in decrease of performance.

Another obvious solution that can be used in such cases is colour segmentation. Colour segmentation can be used to isolate small sized objects by using blob detection (thresholding the size of the blob). This method is also not very robust because the colour segmentation problem is not yet solved in the general case. Thresholding the size of colour blobs can limit the blobs incorrectly generated due to noise but will also limit small size objects being detected by the robot. In other words, thresholding poses a maximum detection distance limit on the robotic system.

There have been improvements in key-points based detection methods, as discussed in the previous section, over the past few years. A thorough elaboration of vision methods is given in the theoretical section of the report. This has lead to the couple of promising techniques like SIFT and SURF. Since such methods process whole images, they consume a lot of computational resources. Although SURF is promised to be very robust and efficient, processing small sized object poses a challenge with this technique. It is also important to note that due to lack of texture information, detection of key-points is not a robust solution.

The use of hybrid methods is quite beneficial in such cases and can increase the overall accuracy of the system. This study focuses on the performance of a hybrid method using colour segmentation as a pre-processing step. The main driving force behind this is the success of such hybrid systems in the past as described in the previous section. A novel hybrid system, 'HistSURF', is proposed and compared with other techniques in experimentation for vision processing.

The focus of the investigation is to study the performance of robotic vision and the use of vision as a basis for robot control in automated search tasks. It could also be interesting to explore the effect on visual object recognition of changes in the distance between the mobile robot and the target object.

The motion mechanism of a mobile robot includes many parameters, such as: number of wheels, speed mechanism and steering system. The configuration of a robotic system is application specific with the number of wheels ranging from two to six, and a speed mechanism can be uni-speed, auto-acceleration or differential drive. A steering system can also take many configurations, such as: omni-directional, syncho drive, car-like and differential drive steering. The four-wheel differential drive system is used in the platform being used for the development and experimentation in this project.

The differential drive steering system determines the orientation of movement based on speed of its individual channels. In a four-wheel differential drive system there are usually two independent channels (right and left channel) on both sides of the robot. A forward movement can be attained by keeping equal speed on both channels and directional movement depend on the difference in speed on the channels. The vehicle moves toward the opposite side of the channel that has higher speed as compared with the other channel.

The differential drive is usually controlled by a speed model designed specifically for the system. This model can be a simple relation between angular and linear velocity or can be a complex system that takes all frictional forces into account. In both cases, such model is specific to the application area and is not flexible or able to be generalized without careful reanalysis. Another solution used in such cases is the measurement of orientation of the robot using some hardware measurement like a Gyro(scope), Compass, magnetometer, etc.. These measurement components provide ease of operation but their use is not without problems. As with any other electronics measurement equipments these suffer from line noise due to external electromagnetic interference and jitters caused mostly by the internal circuitry. In this study a comparison for turning of robot is made between a Gyro solution and a modelling solution.

1.1 Aim:

The main aim is to develop a prototype navigation system for robotic platform as well as a vision system for detecting target objects. The platform will be used to investigate performance of the visual system for the mobile robot and an experiment on robotic motion will be performed to determine the suitability of a Gyro for motion sensing in the specific robotic architecture. These topics will comprise a prestudy for the use of motion sensing for enhancing automated visual object detection.

1.2 Objectives:

- Literature Search.
- Development of vision system for robotic platform.
- Development of navigation system.
- Experimentation to evaluate developed systems.
- Result writing.

3.1 Research Questions

RQ1: What will be the performance of the visual object detection system?
RQ2: Does the distance from the camera affect the performance of participating vision algorithms?

RQ3: Can a Gyro be used for reducing the error in robotic movement due to friction, skidding, noise in the transduction of control signals to wheel motion, geometrical uncertainties (wheel alignment, shape, axial displacement), etc.?

The research questions will be answered using experiments on vision and the navigation system being developed. An experimental investigation on vision will determine the better method among the selected techniques. Also investigated by the vision experiment will be the presence of any impact on performance due to object distance from the camera for a key-point based method like SURF. The experiment on robotic motion will answer the third research question that addresses the problem of the mismatch between ideal robot motion and position based upon intended results of control signals (i.e. motion and position determination by *dead reckoning*) and actual motion and position for any mobile robotic system.

CHAPTER 3: METHODOLOGY

3.1 Vision Methodology

In this section, process of object detection that is used in the experiment is explained. The vision process that includes a new hybrid algorithm '*HistSURF'* along with the established methods *ColorHistogram* and *TrainedShapes* will be described. The other two techniques used, PCA and SURF, are described below in Chapter 4 Theoretical Work.

3.1.1 Colour Histogram

This section describes the process used to detect objects in an image using *colour histogram comparison [35]*. Keeping in view the nature of the target objects (small coloured rectangular blocks), segmentation seems an appropriate approach in order to identify objects within colour images. As first step, the image is processed for smoothing and noise removal. A mean filter is applied to remove noisy pixels. Mean filtering is a simple, intuitive and easy to implement method of smoothing images, *i.e.* reducing the amount of intensity variation between one pixel and the next. The colours of this smoothened image are analysed in HSL colour space. Each desired colour of the object (Red, Blue, Yellow, and Green) is extracted by selecting an appropriate range for Hue, Saturation and Luminance. The details of HSL colour space are described in the subsection of Chapter 4 on Image Processing. After HSL filtration, rectangular regions of candidate objects containing colour blobs are obtained. Only blobs with $\tau < B$ are selected, where B is the number of pixels in the extracted colour blob and τ is the threshold that is set manually.

After the extraction of candidate objects, a method for determining the true objects is required. A model image for each coloured object is used for training the histogram. Three colour histograms are extracted for the training image and normalized to achieve invariance using the following computation:

$$RHist = (h_r - \overline{h}_r)/max(h_r)$$

$$GHist = (h_g - \overline{h}_g)/max(h_g)$$

$$BHist = (h_b - \overline{h}_b)/max(h_b)$$

where h_r ,h_g *and* h_b are three (red, green and blue) colour histograms for the training image while $\overline{h_r}, \overline{h_g}$ *and* $\overline{h_b}$ are their respective means.

The normalized training histograms are used to determine whether the candidate object is actually a target object or not. A normalized mean Euclidean distance is used to obtain the similarity measure between each training object and a target object. Three distances d_r ,d_g *and* d_b are computed for respective colour histograms as follows:

$$d_r = \sum_i \sqrt{(Th_r(i) - h_r(i))^2}$$

$$d_g = \sum_i \sqrt{(Th_g(i) - h_g(i))^2}$$

$$d_b = \sum_i \sqrt{(Th_b(i) - h_b(i))^2}$$

Finally a similarity measure S is defined using the normalized mean of histogram distances.

$$S = 1 - \sum_c \left[\frac{d_c - \min\{d_r, d_g, d_b\}}{\max\{d_r, d_g, d_b\} - \min\{d_r, d_g, d_b\}}\right] / 3$$

where $d_c \in \{d_r, d_{g,} d_b\}$ is the normalized distance between the respective colour histograms. The value of S >0.3 is used for detection of the objects.

3.1.2 TrainedShapes

Another approach that is used for object detection is the comparison of object shapes, *TrainedShapes*. A shape is represented in the form of a feature vector. A number of feature vectors are stored and are compared against the feature vector of the unknown object.

The boundary around the pixels of a colour blob, obtained by the application of HSL filtration, is used for object representation. The procedure for extraction of a colour blob is the same as explained in the previous section. The extracted boundary points are then resampled into an equal number of N points using cubic spline interpolation. A centroidal distance function r_i for i = 1,2,…,N is given as follows[39]:

$$r_i = \sqrt{(x_i - \overline{x})^2 + (y_i - \overline{y})^2}$$

where (x_i,y_i) represent the coordinates of the ith boundary point and $\overline{x}$ represents the centroid.

The distance vector $r_i = \{r_1, r_2, \dots, r_N\}$ is transformed into the frequency domain by using Fast Fourier Transform (FFT). The feature vector *f* is then defined as follows [39]:

$$f = \left(\frac{|F_1|}{|F_0|}, \frac{|F_2|}{|F_0|}, \dots, \frac{|F_N|}{|F_0|}\right)$$

where $|F_i|$ is the ith Fourier coefficient. The division by $|F_0|$ results in scale invariance and taking the magnitude gives the rotational invariance. A training set $F = \{f_1, f_2, \dots, f_k\}$ is formed, where 'k' is the number of training images.

A Euclidean distance 'd_m' is taken between the target feature vector 'l' and each of the feature vectors in the training set F. A degree of difference (DOD) is computed by calculating the mean of the differences, where an object is considered as detected if DOD >0.4:

$$DOD = \sum_m \frac{d_m}{k}$$

3.1.3 HistSURF

This section describes a novel hybrid method that is used for object detection, referred to here as *HistSURF*. A direct combination approach is used to combine Histogram comparison and the SURF features. The results of both techniques are used for object detection. A combined similarity measure is defined as follows:

$$S = \beta R_c + (1 - \beta) R_s$$

where R_c is the result of the color histogram comparison algorithm while R_s is determined by the following function:

$$R_s = \begin{cases} 1 & matchedFeatures > \tau \\ 0 & otherwise \end{cases}$$

where τ is a threshold for matched features retuned by the SURF algorithm and a value between 3-10 is used in this study. β is the combining factor whose value should be determined by a learning procedure, but that is not the target of this study. β = 0.5 is used in this study.

3.2 Robotic Navigation

In this section, robotic drive structure, locomotion and trajectory control are explained. There are many kinematic designs for mobile robots, with variations in the number of wheels and their configuration. A very common configuration is the differential drive system.. The robotic system used for this study is a four wheel differential drive robot with sensors and actuators as shown in the figure 3.1.

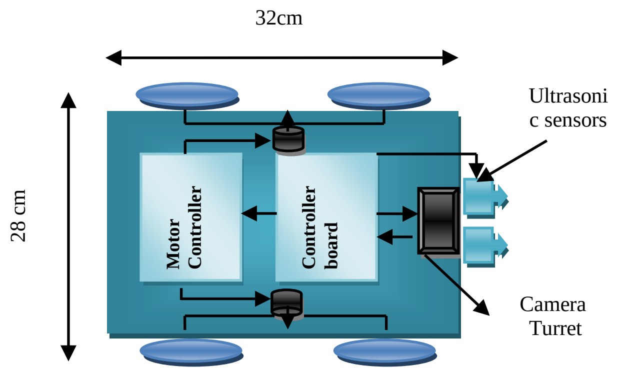


Figure 3.1. Robot with sensors and actuators.

The overall navigation process includes an instinctive behavior layer [40] as well as a command layer which can override the instinctive process. A flow of the navigation process is depicted in the diagram 3.2.

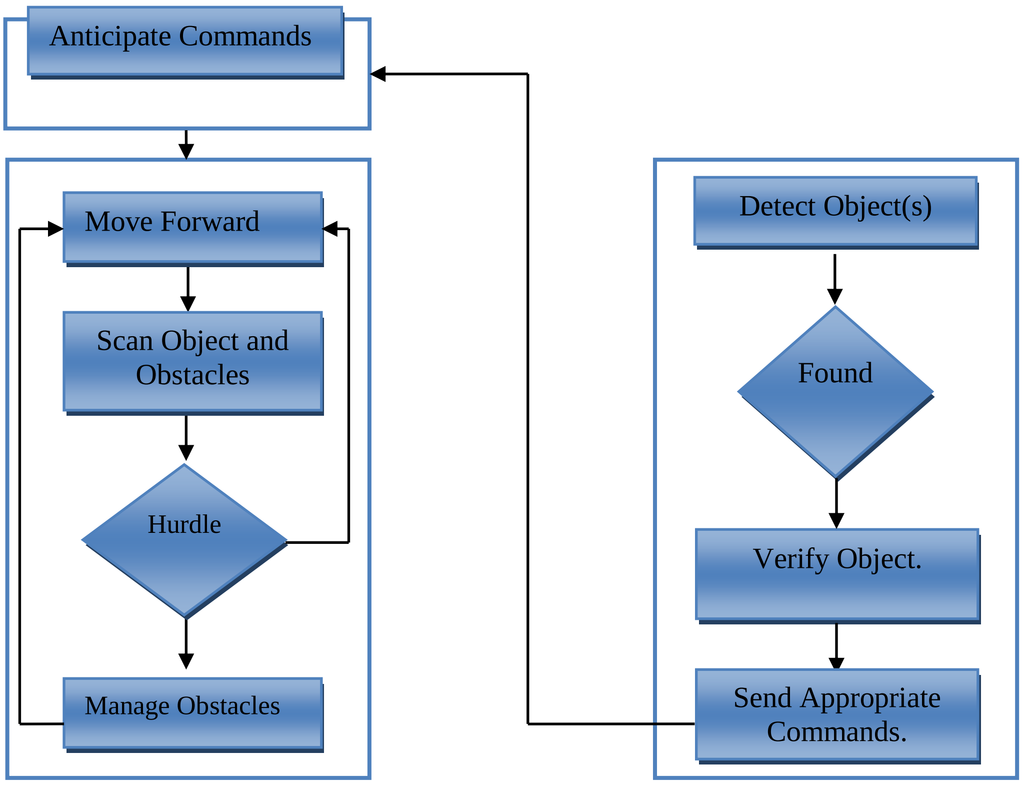


Figure 3.2: (a) Command & Instinctive layer (Robot) (b) Vision Processing (PC).

3.2.1 Differential Drive

A differential drive is a type of mobile vehicle that has separate motors driving the left and right sides of the vehicle, which can therefore have differential speeds. A common configuration of differential drive uses two independent primary drive motors, one for each side wheel, and one or two caster wheels at the rear of the platform to stabilize the vehicle. The combined speed and direction of the motor connected with each wheel determines the overall direction and speed of the vehicle. If the speed on one wheel (channel) is same as the speed on other wheel then vehicle either moves in forward or in reverse direction without turning. The difference in speeds on the channels results in rotation of the vehicle. The vehicle will turn toward the side which has less speed than the speed on the other channel. If one channel moves forward and the other channel either remains stationary or moves backward then vehicle will spin around the stationary wheel or the center of the axle of the two wheels, respectively. Hence differential drives provide a mechanically simple mechanism with high maneuverability that is easy to control [41].
The mobile robot being used has a differential control which is depicted in figure 3.1. In this particular configuration four motors are used, with each pair of motors on the same side connected in parallel to the electrical output of a single channel of the motor controller. Hence the motors on one side are driven by a single channel of the controller board. The motor controller which receives PWM signals from the microcontroller board. PWM control is described below.

In order to achieve different speed steps, a *virtual gearing* technique is used. In this technique, the overall speed of the vehicle is divided into eight different speed steps controlled by different PWM values and realised by different motor drive voltage levels, referred to using the metaphor of different mechanical gears. The forward and backward speed on each channel is determined as follows:

$$S_f = stationarypoint + stepsize * gear$$
$$S_b = stationarypoint - stepsize * gear$$

where *stationarypoint* is determined for each channel individually and *stepsize* is a constant number used to vary gear speed. The value of *gear* varies from 0 to 9 where 0 stands for no movement and 9 for the highest possible speed.

3.2.2 Motor Controller

The motor controller on the robot is a 'Pololu TReX Dual Motor Controller' as shown in figure 3.3. The controller board can receive serial commands as well as PWM signals from five channels to drive two motors. Each channel can be moved independently, and it is also possible for a combined movement to be generated by a PWM signal (as configured by jumper, but combined control is not used in this study). The speed controller board receives input signals from the main robot microcontroller board and increases/decreases voltage on each channel with appropriate polarity, which determines the direction and speed of the DC motors.

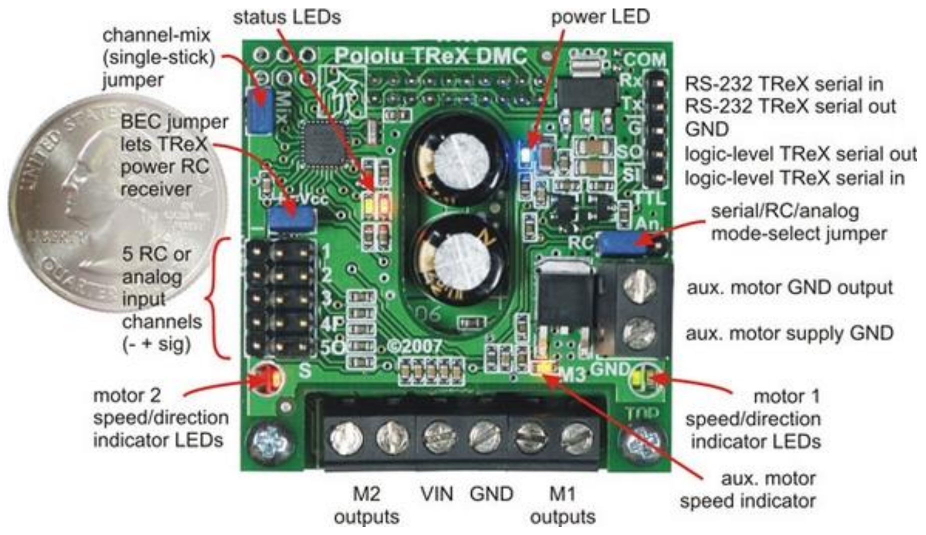

status LEDs
power LED
channel-mix (single-stick) jumper
Pololu TReX DMC
RS-232 TReX serial in
RS-232 TReX serial out
GND
logic-level TReX serial out
logic-level TReX serial in
BEC jumper lets TReX power RC receiver
serial/RC/analog mode-select jumper
5 RC or analog input channels (- + sig)
aux. motor GND output
aux. motor supply GND
motor 2 speed/direction indicator LEDs
motor 1 speed/direction indicator LEDs
aux. motor speed indicator
M2 outputs
VIN
GND
M1 outputs

Figure 3.3: Pololu TReX Dual Motor Controller [33]

3.2.3 Microcontroller Board

The control of the motor movement by sending PWM signals to the motor controller, reading sensors values and serial communication over the wireless network are performed by a microcontroller on an Arduino embedded controller board. An Arduino board contains multiple digital and analogue I/O lines for reading or writing digital or analogue output. This board is responsible for the generation of the PWM signals that are fed into the RC/analogue channels of the motor controller. The software library used to program PWM outputs from the microcontroller board for the generation of PWM signals represents servo rotation angles of 0 to 180. When these values represent motor voltages, 180 is the voltage for full speed forward, 0 is the voltage for full speed reverse, while at 90 voltage is 0 V and the motor is stationary.

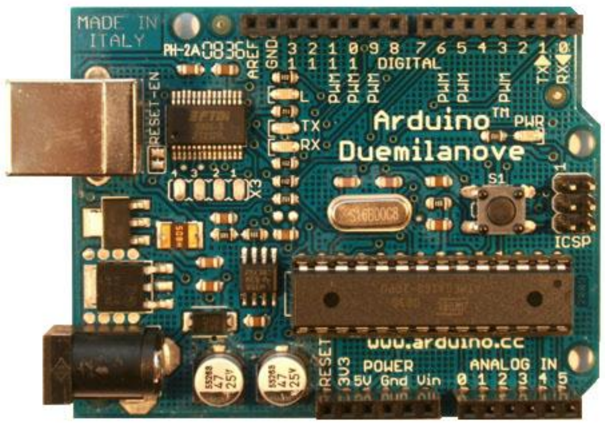
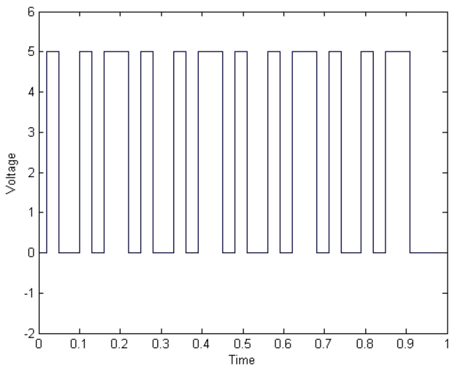
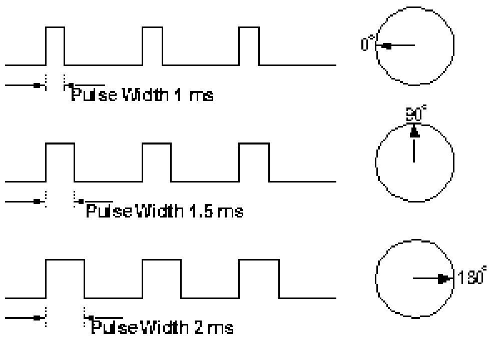


Figure 3.4: Arduino Duemilanove Board.

3.2.4 PWM (Pulse Width Modulation)

PWM [42] is a technique that is used to provide continuous intermediate values of power between fully off and fully on, but using only two discrete voltage levels. Different voltages can be obtained by varying the portion of the time that is used for on verses the portion of the time is used for off. The duration of the on time is called pulse width. The proportion of the on time to the off time is called the *duty cycle* of the pulse. A 100% duty cycle is when the pulse is on throughout a cycle. The microcontroller board therefore generates voltages of either 0 or 5V, with the 5V level being held for a period constituting the width of the pulse being generated. A sample PWM signal with the Frequency of 50Hz is shown in the figure below.





(a) (b)

Figure 3.5:(a)A PWM signal with frequency of 50Hz. (b)Servo PWM and angle[42].

The pulse used to drive a servo is a specialized form of PWM signal but has some differences. In PWM, voltage is varied using the duty cycle of the pulse in order to vary the voltage. This is useful to drive D.C. motors, but the servo signal used to drive the motor controller board doesn't rely directly on transferring the duty cycle to the motors. A servo moves its shaft according to the pulse width of 1 to 2ms supplied to it after each 20ms. It can be seen in the figure 3.5(b) that a pulse width of 1.5ms makes the shaft stationary at 90°. A pulse width of 0-1.5ms rotates the servo from 0-90° while a pulse width of 1.5-2ms rotates the shaft from 90° to 180°.

PWM servo pulses are effective in controlling the D.C motors of the robot since the robot's motor controller uses these pulses to control the speed and direction of motors on its two independent channels. In addition to using two servos PWM channels to drive the motor controllers, direct servo PWM pulses are used to control two servos driving the turret of the robot that has a camera mounted within it.

3.2.5 Gyro

In order to determine the correct orientation of the robot, a Gyro [43] is used. A Gyro is a device that can be used to determine the rotation speed of an object when the object rotates around one or more of its axes. A 3-axis gyro can return three dimensional rotational velocity parameters. For example, a rotational acceleration around the x-axis represents an angular turn made by robot around the x-axis, and similarly for the y and z axes.
In this study, the Gyro is used to determine the angle attained by the differential drive system when turning, and then the output is compared with the output of the dead reckoning model described in the previous section.

3.2.6 Movement Model

The problem of movement control is solved using two methods, one with the Gyro as explained earlier and second with the modelling of the system motion and position in relation to its starting point using dead reckoning [44]. Motion control is based on the speed of the motors on each channel. While in rotation, the robot moves with an angular velocity. The angular velocity and the time taken to complete the motion at linear speed 'v' is defined as follows:

$$\omega = \frac{d\theta}{dt} \qquad \text{(i)}$$

$$t = \frac{S}{v} \qquad \text{(ii)}$$

where θ is the angular displacement over time t and 'S' is the arc length of the robotic spin around its axis in case of angular movement. Using definition of arc length, $S = l\theta$ and (ii) the total turning time can be calculated as follows:

$$t = l\theta/v \qquad \text{(iii)}$$

where in this case, l=38cm is the length of the axle of the robot. At linear velocity v=60cm/sec (first step of differential drive), and the desired angle $\theta=\pi/2$ rad, the total time needed is one second. So at step one, the angular velocity is $\pi/2$ rad/sec obtained by moving the both channels at same speed with different direction.

In case of turning with a larger diameter, the equation should take the difference of speed on each channel into account. The smooth turn with a radial movement is attained by moving the two channels at constant but different speed in the same direction. The combined speed v_c at which robot will turn is the difference of speed on right and left channels i.e. $v_c = v_R - v_L$ so, equation (iii) can be redefined as follows:

$$t = l\theta / |v_R - v_L| \qquad \text{(iv)}$$

Using the above equation, the time required for the robot to complete a smooth turn can be attained. The degree of smoothness depends on the speeds of the independent channels.

3.2.7 Obstacle Avoidance

The problem of obstacle avoidance is a basic but crucial part of any autonomous navigation system. In order to detect obstacles various detection configurations may be designed. In this study, IR sensors along with ultrasonic range finders are used. The ultrasonic sensors tackle the stationary as well as moving obstacles that are encountered in front of the robot while IR range finders help in determining the correct direction for the robot to turn after it has detected a frontal obstacle.

i. IR Range Finder

There are two Sharp GP2D12 [45] infrared (IR) range finders attached on both sides of the robot's turret. Each IR sensor is attached to an analog port on the Arduino board. These are used to detect obstacles around the robot. The connections of the sensor are shown in figure 3.8. The GP2D12 uses an IR emitter and a linear CCD array detector that is 3/4" away from the IR emitter in order to detect the distance from the object. It can detect distances from 10cm to 76cm.

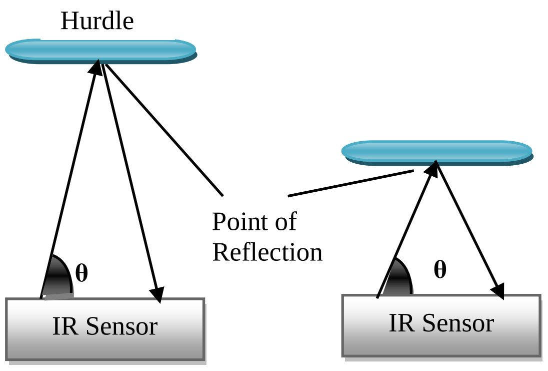


Figure 3.7: Functioning of an IR sensor.

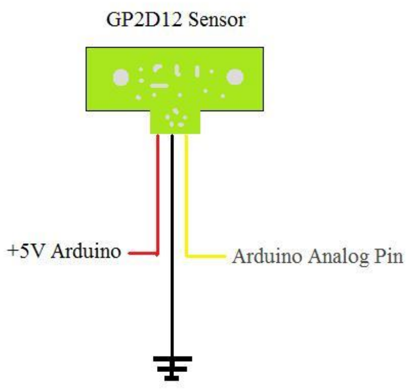


Figure 3.8: Pin out of the GP2D12 IR range finder.

A light is transmitted from the IR emitter which is reflected back and is detected by the IR detector. The distance from the sensor is calculated from the triangle formed by light emission and reflection.

ii. Ultra Sonic Range Finder

Two ultrasonic range finders (Maxbotix EZ1) [46] are mounted on the robot, one in front of the robot on the main body to detect stationary hurdles and one on the turret to avoid the moving obstacles. The circuit board along with the wiring diagram is shown in figures 3.9 and 3.10, respectively. An ultrasonic distance finder works in a way similar to an IR range finder except it uses time taken by a sound wave for the calculation of distance. A sonar wave is emitted by a transmitter which bounces back after striking an object and is then detected by the receiver.

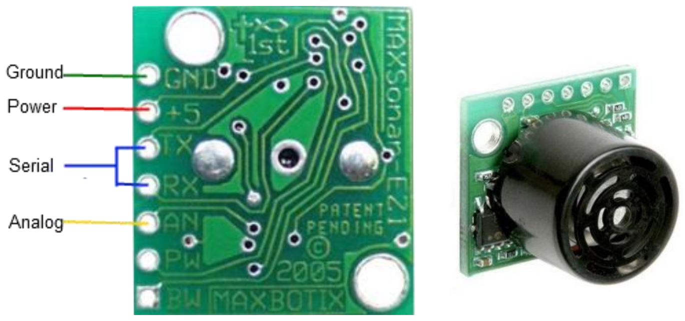


Figure 3.9: MaxBoltix Ultrasonic range finder.

The analogue output of the ultrasonic sensor is used, as seen in the connection diagram (figure 3.10). The output from the sensor is a voltage with 9.8mV increase per inch of object distance from the sensor. The analogue input of arduino is converted to digital using a 10-bit Analogue to Digital Converter (ADC) by a o the board and this value can be read by the microcontroller chip (ATMega328). This results in a value in the integer range from 0 to 1023 (i.e. 10 binary digits). Since arduino and the sensors operate on a 5V supply, every increase of 4.88mV on analogue line results in a unit increase for the value of the ADC. In terms of distance, it means an increase of two ADC values is attained for each inch.

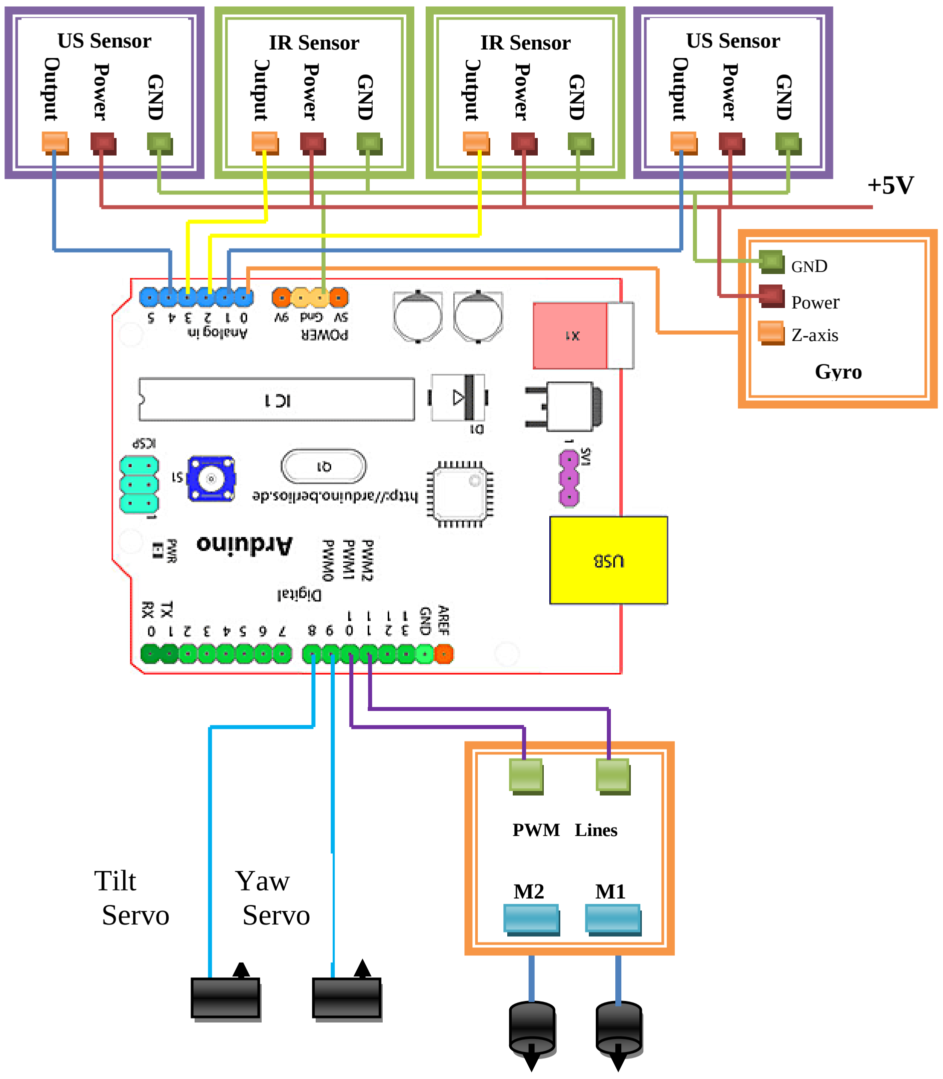


Figure 3.10: Connection diagram used for connecting sensors to Arduino.

The obstacle detection system works as an instinctive layer for the robotic decision making. It makes use of ultrasonic and IR sensors connected by Arduino as shown in figure 3.10. Based on the input from the sensors, the microcontroller takes autonomous decision for controlling the motors and servos. The servos are used to achieve camera yaw and tilt while DC motors are connected with the output channels of the motor controller to provide the differential drive mechanism.

iv Instinctive behavior

The instinctive behavior [40] of the robot is achieved by reaction rules driven by the output of the sensors. The coordination of the sensors on the moving turret and the sensor fixed in front of the robot are used to implement an obstacle avoidance mechanism. There are two IR and one Ultrasonic sensor attached to the turret of the robot. As the robot moves forward, its turret is continuously yawed left and right approximately 40 degrees from the center. This makes the camera as well as the Ultrasonic sensor scan the environment. A second ultrasonic sensor that is statically attached to the body of the robot repeatedly scans the in front of the robot. The stopping criterion for the detection of frontal obstacles is defined as follows:

$$D = \begin{cases} 1 & U_f < 30cm \text{ or } U_t < 25cm \\ 0 & otherwise \end{cases}$$

where U_f , U_t are the average distances taken for five repeated ultrasonic scans of two ultrasonic sensors respectively. The value of 'D' determines whether the robot is stopped or not. A non zero 'D' initiates the process that determines the turn direction.
In order to determine the side which the robot should turn, the turret of the robot is first moved 20 degrees to left and brought to the center slowly. Meanwhile, readings from the left IR sensor are taken repeatedly and the robot is rotated left if all IR readings remain below a threshold. The process is repeated for the right IR sensor in case of any IR reading falling below the minimum threshold distance. The pseudo code for the operation can be given as follows:

```
If ( D > 0)
{
        Apply Brakes().
MoveTurret (90-scanrange)
While(angle < 90)
  {
     MoveTurret(angle)
    angle=angle +1;
    if (LeftIR < MinDist )
     {
        Blocked = true
       Break
     }
  }
if ( Blocked = True )
{
     Blocked = false
     MoveTurret(90 + scanrange)
While(angle > 90)
  {
     MoveTurret(angle)
     angle=angle -1;
     if (RightIR < MinDist )
      {
         Blocked = true
        Break
      }
   }
   Turn(Right)
}
Else
Turn(Left)
}
if ( Blocked = True)
 ChangeDirection(Reverse)
```

3.2.8 Communication Protocol

There is bidirectional wireless communication between the robot and the remote computer, each having an xBee transceiver[47]. In order to perform such messaging communication, a serial protocol has been used for sending character sequences over the serial interface. Arduino SerialSoft library [42] is used for the transmission and reception of the data across the serial interface. In Table 3.1 there is a listing of each operation with a character sequence that is needed for the completion of the task. The character sequence consists of an alphabet followed by one or series of digits depending on the nature of the operation. For example, sending a character sequence 'L4' turns the camera 40° to the left. Similarly, a character sequence 'S045' turn the robot 45° to the right.

Table 3.1: Serial Communication Protocol.

No.	Function	Character Sequence
1	Camera Yaw left / Camera Yaw right	L[0-9] / R[0-9]
2	Camera Tilt Up / Camera Tilt Down	U[0-9] / D[0-9]
3	Accelerate/Decelerate	W[0-9] / X[0-9]
4	Turn Right / Turn Left	S[000-360] / A[000-360]

CHAPTER 4: THEORETICAL WORK

In this section, some of the related techniques used in the study are been explained along with an overall architecture of the system and a vision framework. Some object detection methods are discussed as a reference that are directly used in the study or are related to the study.

4.1 Robotic Architecture

The robotic architecture consists of multiple devices that are interconnected to give the final solution. A central controller is responsible for the management of motor movement and processing of sensor input. It also is connected to a remote computer by wireless serial data link. There is a second, but unidirectional, wireless link that connects the camera output with an A/V receiver attached to the remote computer.

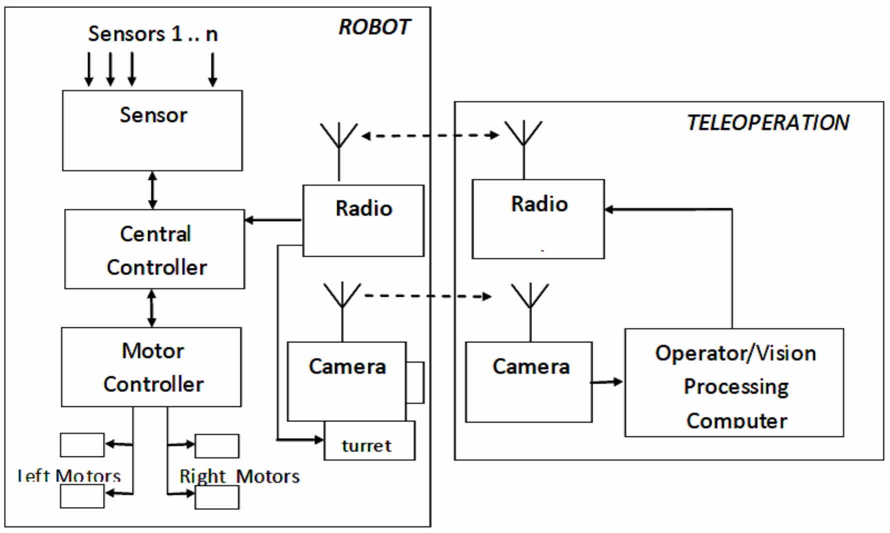


Figure 4.1: Robotic architecture diagram.

There is a turret on the robotic platform that helps in achieving head-like yaw and tilt for the vision system. The motor controller is responsible for the movement of individual channels (and hence pairs of wheels) of the robot. A serial wireless data link is used to transfer control information between a remote computer and the central controller system. This wireless communication is achieved by using an Xbee wireless device that can create a wireless PAN network. The basic electromechanical platform was provided at the beginning of the project. However, the platform was continuously developed, changed and extended as part of this project, resulting in the final configuration and specific components described in this thesis. All robot software was created within this project, together with the communications link with the remote computer. This software incorporates various code examples from diverse open sources on the internet (e.g. at www.arduino.cc), integrated into a coherent system within this project.

4.2 Vision Framework

The vision framework consists of offline and online processing. In online processing,

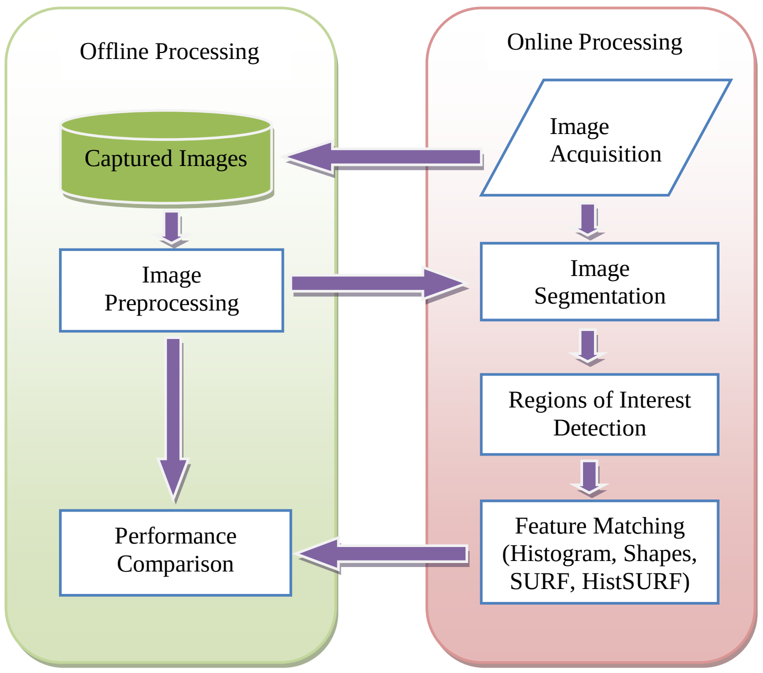


Figure 4.2 Framework for the vision system

images captured by the wireless camera are processed at the remote computer and object detection is conducted in real-time. In offline processing, performance evaluation of the various image processing techniques used is performed on a database of already acquired images.

4.3 Object Detection

There has been an enormous amount of research in the area of object detection due to its importance in the field computer vision. Humanoid robots which identify objects for the understanding of the environment pose high requirement for object detection and recognition [19, 34, 36-38]. For such robotic systems, a vision algorithm is needed that is not only accurate in detecting objects but also does so in a timely fashion. Many object recognition approaches have been proposed but still the problem of object detection has not been solved with great generalizability[34]. Each method has its own specific application area where it can provide good results. It is impossible here to list all the object detection and recognition methods that are available since this is very huge research area. Some of the methods which are relevant to the current study and mobile robots are discussed.

The methods available for object detection can be classified into two major classes, one of methods based on appearance and the other of methods based on models [34]. Appearance based approaches use different views of the object that is to be detected, for comparison. Alternatively, in model based approaches, a general model is built from the geometry extracted from images of the objects. A combination of these approaches could also be used which could result in improved performance in certain situations. The methods based on appearance are most relevant to this study so they will be explored in greater detail. Algorithms based on appearance can further be classified based on their way of storing the views of an object. Storage can be either global or local, where global stores complete views of an object, while local stores them in the form of local features.

4.3.1 Global Approaches

The range of algorithms under this category use complete views of an object for the detection of the object. The views are stored using some representation technique and are matched against the test image of an object using some matching criteria. The matching criteria could be as simple as Euclidean distance or could be complex relations defined to achieve better accuracy. Global approaches need image segmentation as a pre-processing step to extract the object out of the image. This can be a potential drawback of using global appearance-based approaches since the problem of image segmentation is not solved in the general case [34]. Image segmentation is usually performed with assumptions like one colour or static background, static camera and controlled environment.

4.3.1.1 Color Cooccurrence Histograms

Colour information is used to segment an image when the vibrant feature of the object is its colour. An approach toward detecting objects using its colour is *histogram comparison*. The only problem with this approach is that it doesn't consider the spatial relationship of pixels. Since geometrical information is not embedded in this approach, objects with the same colour and different appearance cannot be differentiated. An extension to this approach, Color Cooccurrence Histogram (CCH) was proposed in [35] that embeds the spatial relationship between the pixels. The objects from the images are extracted and then represented in the form of CCH that includes the occurrences of two pixels $p_1 = (R_1, G_1, B_1)$ and $P_2 = (R_2, G_2, B_2)$. The distance between the pixels is calculated using Euclidean distance $s = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ where (x_1, y_1) and (x_2, y_2) are the position of pixels in respective image planes of the objects. The number of distinct colours is reduced to m distinct colours using quantization. The distances are also quantized into a set of n distances. Quantization is performed by applying a k-means clustering algorithm [35]. A CCH is formed using a quantized version of the histogram and distances. These CCH are stored as views of the object for later comparison.
Whenever, a new image arrives, it is searched for a rectangular area that has a CCH similar to one of the stored CCHs. Both rectangles can be of different sizes. The comparison of the image CCH, $\mathrm{CCH}_i\,(i, j, k)$ with the model CCH, $\mathrm{CCH}_m\,(i, j, k)$ is made by calculating the intersection between them:

$$P = \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{n} \min\{\mathrm{CCH}_i(i,j,k), \mathrm{CCH}_m(i,j,k)\}$$

This intersection is used as a similarity measure for deciding the presence of the object in a test image. The decision for object detection is made if the intersection exceeds a certain threshold T. As proposed in [35] the size of the search rectangle and the number of different colours m to be used is determined by analysing the false alarm rate. The efficiency of the algorithm can be improved if rectangles with 50% overlap of the area are used [35]. This algorithm can perform with a reasonable degree of accuracy even in the clutter and occlusion as shown in the test image in figure 4.3.

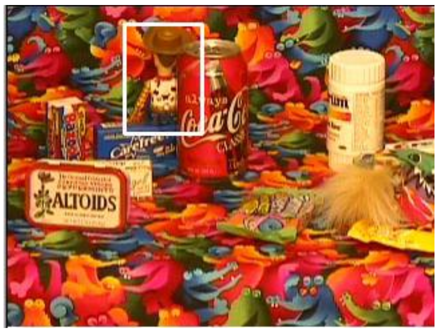


Figure 4.3 Result of experiment in clutter and occlusion [35].

As with any other colour based detection algorithms the CCH method is also vulnerable to changes in lightning conditions.

4.3.1.2 Object Detection using Integral Images

Another approach that uses rectangular regions as a search strategy is object detector proposed by Viola and Jones [6]. This algorithm doesn't require segmentation as a pre-processing step since segmentation could be a potential drawback due to it having not been solved in general. The detector proposed by Viola and Jones can be used to detect a target object in test images but it cannot be used to solve the classification problem. It was intended to solve the problem of face detection for which it is an effective and efficient detection solution. Object detection using the proposed procedure involves division of an image into *rectangular features*. There are three different rectangular features that have been proposed as shown in figure 3. If two rectangle features are used then the resultant value will be the sum of intensities within both rectangles. In the case of three rectangular features, the sum of pixel values within two outside rectangles is subtracted from the sum from the middle rectangle. The diagonal sums are subtracted in the case of four rectangular features.

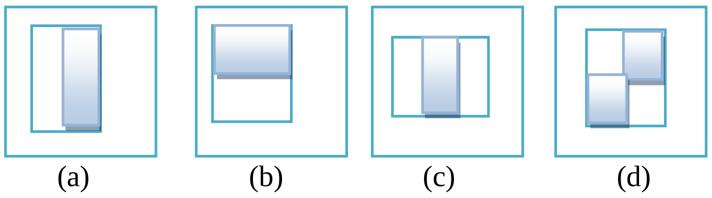


Figure 4.4: Two rectangle features are shown in (a) and (b). Three and four rectangular features are shown in (c) and (d) respectively [6].

The proposed algorithm is based on a new image representation that is called "Integral Images" that results in fast detection. The integral image at location *a*, *b* is the sum of all the features above and left of the (a, b), as shown in figure 4:

$$\mathrm{II}(a,b) = \sum_{\mathrm{m=0}}^{\mathrm{a}} \sum_{\mathrm{n=0}}^{\mathrm{b}} \mathrm{I}(a,b)$$

where II(a, b) is the integral of the image and I(a, b) is the original image.

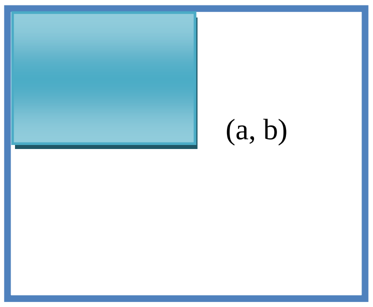


Figure 4.5: The value at point (a, b) is the sum of all features to left and above of (a, b) [6].

A recursive definition that can be used to obtain integral images in a single image processing pass is:

$$s(a,b) = s(a,b-1) + \mathrm{I}(a,b)$$
$$II(a,b) = II(a-1,b) + s(a,b)$$

This recursive model is used to calculate any rectangular sum with four array references. Due to the adjacency of rectangles, two rectangular features need six array references, three rectangular features need eight and four rectangular features need nine [6].
Another important contribution by Viola and Jones is boosted learning through use of the AdaBoost algorithm [6]. This algorithm can improve the process of detection by boosting ordinary learning algorithm like a decision tree or rule learner. It is also called a weak learner because even a best classification function cannot classify training data well [6]. This weak classifier $c_i(z)$ with feature f_i , threshold t_i and a parity p_i is given as:

$$c_i(z) = \begin{cases} 1 & : \quad p_i f_i(z) < p_i t_i \\ 0 & : \quad otherwise \end{cases}$$

where z is a 24x24 rectangular window of an image. It has been seen that single feature learning (i.e. using only one rectangular feature) is not good for obtaining greater accuracy; usually the range of the error rate for good features is between 0.1 and 0.3 while the range for bad features is between 0.4 and 0.5 [6]. The processing time of the algorithm for a greyscale image of size 384x288 is 0.7 seconds. This time seems to pose a difficulty for its use in real-time applications. In order to improve the efficiency of the classifier, a cascade of boosted classifiers is proposed that is claimed to improve the efficiency as well as the detection rate of the algorithm [6]. This cascade of classifiers uses positive result from one classifier as a trigger for the start of the next classifier. A negative result at any stage results in rejection of that rectangular window, as shown in figure 5.

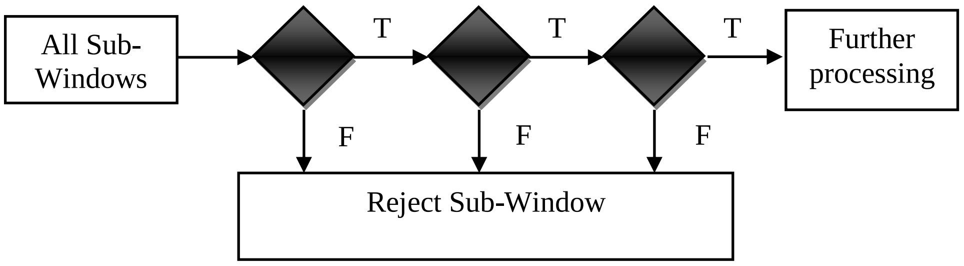


Figure 4.6: A control flow of cascade of classifiers [6].

4.3.2 Local Approaches

Methods using the local approach toward representation store local features obtained from views of an object. These features are usually interest points or feature points, which are used to create feature descriptors for later matching. There is a difference between a feature point and a feature descriptor: a feature point is any 2D point in the image plane selected due to its uniqueness with respect to its neighbourhood, while a feature descriptor is obtained from a collection of feature points when a certain degree of invariance to different views of an object is attained.
The feature descriptors formed by interest points or key-points are matched to determine any similarity among them. In order to calculate the similarity between two feature points, many methods have been proposed. There are two promising feature description

methods that have yielded good results in experiments. These are SIFT (Scale Invariant Feature Transform) [1] and SURF (Speeded Up Robust Features) [2].

4.3.2.1 Scale Invariant Feature Transform (SIFT)

The Scale Invariant Feature Transform is a key-point based algorithm that provides both a method for feature point calculation and a feature descriptor that is invariant to scale and transformation. The feature descriptor is formed using gradient histograms and its dimension is 128. Key-points from the image are detected that are invariant to image transformations. An image convolution is performed using Gaussian G(a, b, σ) in order to form a scale space of the image that is defined as a function S(a, b, σ) and is given as follows [1]:

$$\mathrm{S}(\mathrm{a},\mathrm{b},\sigma) = G(a,b,\sigma) * I(a,b)$$

where * is the convolution operator and *I(a,b)* is the image pixel at position (*a*, *b)*. The Guassian G(a, b, σ) with σ being the Gaussian smoothness operator is given as

$$G(a,b,\sigma) = \frac{1}{2\pi\sigma^2}\exp\left\{-\frac{a^2+b^2}{2\sigma^2}\right\}.$$

A *Difference of Gaussian* function (DOG), D(a, b, σ) is calculated in order to find the stable key-points in the scale space of the image. The DOG is obtained by taking the difference of two neighbouring scales that are separated by a multiplicative factor *c [1].*

$$D(a,b,\sigma) = \big(G(a,b,c\sigma) - G(a,b,\sigma)\big) * \mathrm{I}(\mathrm{a},\mathrm{b})$$
$$D(a,b,\sigma) = S(a,b,c\sigma) - \mathrm{S}(\mathrm{a},\mathrm{b},\sigma)$$

There is a process for construction of DOG in [2]. A set of scale spaces is obtained by repeatedly applying convolution on the input image so that multiple images separated by multiplicative factor *c* are obtained. The set of scale spaces is divided into sub sets named *octaves*. Each *octave* has *n* number of images and the multiplicative factor $c=2^{1/n}$. A set of DOG images is obtained by subtracting the images from adjacent scale spaces in each octave. The local extrema for each DOG is obtained by searching for local minima and local maxima. A point is selected as a stable key-point if it is largest or smallest among all the 28 neighbours (i.e. eight neighbours in the same scale and nine neighbours across scales). The process of creating DOG images for one octave is shown in figure 6.

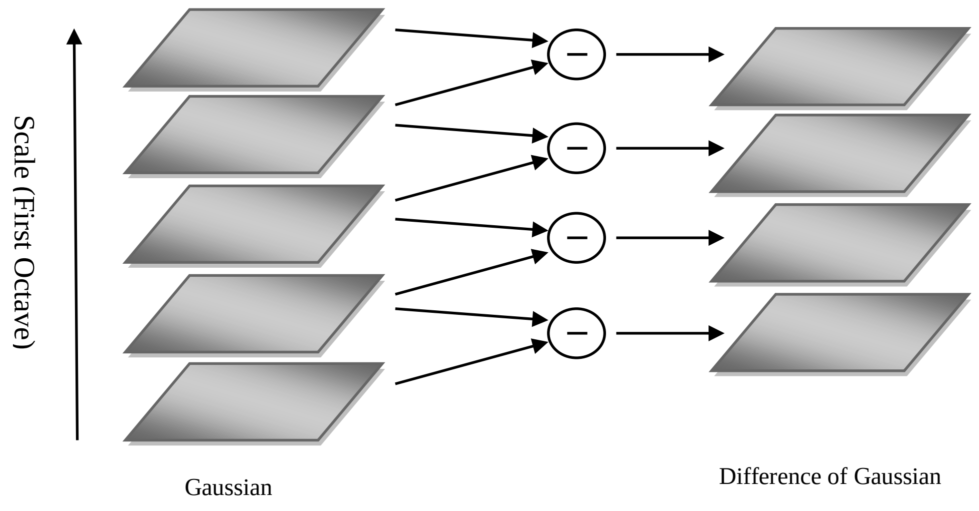


Figure 4.7: Conversion from Gaussian images to DOG images in one octave [2].

The Gaussian image that has σ two times as large as the initial value is resampled to half of its original size in each octave. The resampling is performed by only considering every second pixel in each row and column. The repeatability of the stable key-points is dependent on the Gaussian smoothness factor σ of each octave. A suitable value has been proposed in [2] that is $\sigma = 1.6$. To preserve the highest frequencies which could be discarded by high smoothness, the image is doubled to get the effect of desired smoothness with $\sigma = 0.5$.
The other contribution of the SIFT algorithm is a method for generating the invariant feature descriptor for each image. This is done by calculating an orientation θ(a, b) and gradient magnitude m(a, b) in a region around the selected key-point. The images are selected which are in the closest neighbourhood in scale space of the key-point and the following calculations are performed [34]:

$$r_a = S(a+1,b) - \mathrm{S(a-1,b)}$$
$$r_b = S(a,b+1) - \mathrm{S(a,b-1)}$$
$$m(a,b) = \sqrt{r_a^2 + r_b^2}$$
$$\theta(a,b) = \arctan\frac{r_b}{r_a}$$

where r_a, r_b are the guassian of the neighbouring scale space of the key point, respectively.
These gradient orientations are arranged in the form of a histogram that has 36 bins and covers the 360 degree range. Orientations inside the histograms are weighted by the magnitude m(a, b). A feature descriptor is formed from the key-points having the highest peaks and those that have a magnitude within 80% of the highest peak. In each direction, a feature descriptor with a different orientation is obtained. A sample histogram and key-point descriptor is shown in figure 4.8.

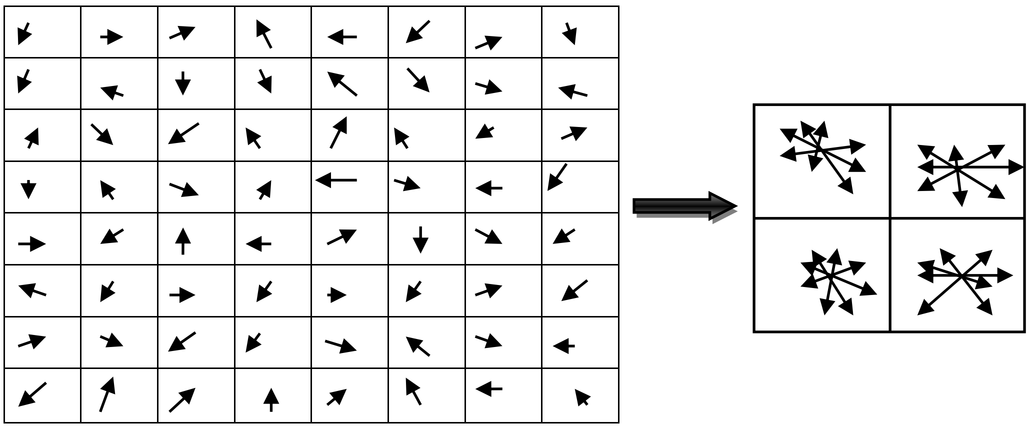


Figure 4.8: A sample view of orientation histogram and key-point descriptor [1].

4.3.2.2 Speeded Up Robust Features (SURF)

A fast variant of SIFT algorithm is Speeded Up Robust Features (SURF) [2], that uses a skimmed version of SIFT and a Hessian matrix based measure for detection. It also uses the concept of ‘Integral Images’ in order to increase the efficiency of the algorithm. In [2] a method for detection and a method for feature description are proposed.
The approach toward finding key-points is based on Hessian matrix approximation. As the first step, an integral image is computed for the original input image. After obtaining

the ‘integral image’, a Hessian matrix is computed. A Hessian matrix H(a, b, σ) for a point (a, b) in the Image I is given as follows [2]:

$$H(a,b,\sigma) = \begin{bmatrix} s_{aa}(a,b,\sigma) & s_{ab}(a,b,\sigma) \\ s_{ab}(a,b,\sigma) & s_{bb}(a,b,\sigma) \end{bmatrix}$$

where $s_{aa}(a,b,\sigma)$ is the convolution of the Gaussian second order derivative for a point (a, b) in image I and similarly for $s_{ab}(a,b,\sigma)$ and $s_{bb}(a,b,\sigma)$.

The cropping and discretization of Gaussian images result in loss of repeatability under rotations. To overcome this problem, box filters are used, which are approximate second order Gaussian derivatives and which can be computed with low computational cost using integral images. The approximation for box filters D_{aa} , D_{bb}, and D_{ab} is calculated using the following [2]:

$$\det(H_{approx} = D_{aa} D_{bb} (wD_{ab})^2$$

where ‘w’ is the relative weight used to balance energy conservation between the Gaussian and the approximated Gaussian Kernel [2].

$$w = \frac{|s_{ab}(a,b,1.2)|_F \, |s_{bb}(a,b,9)|_F}{|s_{bb}(a,b,1.2)|_F \, |s_{ab}(a,b,9)|_F} = 0.912$$

where $|k|_F$ is the frobenius norm. As it can be seen, the change in weights depends on scale but these are kept constant since they don’t result in significant impact on detection. The later process of creating octaves and an orientation histogram is quite similar to that of a SIFT descriptor.

4.4 Image processing

The area of image processing is also quite large and impossible to summarise in general with a brief chapter. In this section only those methods that are related to this study are discussed.

4.4.1 Segmentation

The problem of extracting various regions from the image is solved using segmentation. These regions can be colour blobs, geometric shapes (i.e. lines, rectangles, circles, etc.), or certain textures. The process of segmentation is used as a pre-processing step to most object detection algorithms. In comparison to human ability to segment an image obtained by eye, computer based solutions still struggle to attain efficiency and generality. Usually the problem of segmentation is solved by a specific feature and the assumption of a controlled environment. Object detection in robotic vision is usually performed on a segmented image rather than a complete image due to real-time requirements posed by such systems [17-19]. In this section, some of the relevant 2D segmentation methods are discussed.

4.4.2 Thresholding

A very simple form of 2D image segmentation is *thresholding* in which background is discarded and relevant area is extracted. This method is applied on grayscale images that can be considered as 2D matrices of intensity values ranging from 0 to 255. The decision for either keeping the pixel or discarding it is made based on the value of grayscale threshold. The function used to generate a segmented image is[34]:

$$I'(a,b) = \begin{cases} v & \text{if}(a,b) \geq t \\ 0 & \text{otherwise} \end{cases}$$

where t can range from zero to the value of the pixel and *v* is the maximum pixel value. The pixels with zero intensity value are considered as background and the ones with non-zero intensity are considered as the segmented region. The value of the threshold is determined either manually or based on histogram analysis. As with any other segmentation methods, it is suitable to use certain controlled images, e.g. images with a black background. Thresholding is very effective for some biomedical or industrial images while it is not a good solution for vision of humanoid robots.

4.4.3 Background Subtraction

Another simple segmentation method used in the context of a static camera and object detection is *background subtraction* [34]. In this method, an initial image containing only the background from a static camera is taken and is subtracted from subsequent images to get the segmented regions. A general mapping function could be follows:

$$I'(a,b) = \begin{cases} v & \text{if } |I_t(a,b) - I_0(a,b)| \geq m \\ 0 & \text{otherwise} \end{cases}$$

where m is the predefined threshold and I_0 is the initial image. This approach is usually feasible in case of object tracking but is not applicable for moving camera situations. If motion is needed to be segmented only instead of static objects, the previous approach can be used, with a difference between two consecutive frames being taken by:

$$I'(a,b) = \begin{cases} v & \text{if } |I_t(a,b) - I_{t-1}(a,b)| \geq m \\ 0 & \text{otherwise} \end{cases}$$

4.4.4 Colour Based Segmentation

The segmentation of an image into meaningful parts can be performed using colour information. Some important colour spaces for the achievement of the purpose include RGB, HSV and HSL. A brief description of these spaces is presented below and HSL is selected as a segmentation technique.

a) RGB Colour Space

RGB (Red Green Blue) colour space is based on the RGB colour model in which basic colours (Red, Green, Blue) of light are added together to form all possible colours. Any image in RGB colour space can be represented with MxNx3 dimensional space, where M is the number of rows and N is number of columns in each colour dimension.

b) HSV/HSL Spaces

Colour segmentation can also be performed using HSV (Hue, Saturation and Value) or HSL (Hue, Saturation, and Light). In contrast to the RGB colour model, HSV/HSL spaces are more commonly used in image processing tasks [17, 18]. These spaces represent colour information in a way that is closer to human perception. *Hue* represents the type of colour in the space, *Saturation* represents the richness of the colour and *Value/Light* represents the intensity of the light. The use of an HSV/HSL model is

suitable for image segmentation by the vision system; any colour can be isolated by selecting the appropriate Hue range and Saturation value.
An HSV/HSL model can be shown as a cone as shown in figure 4.9.

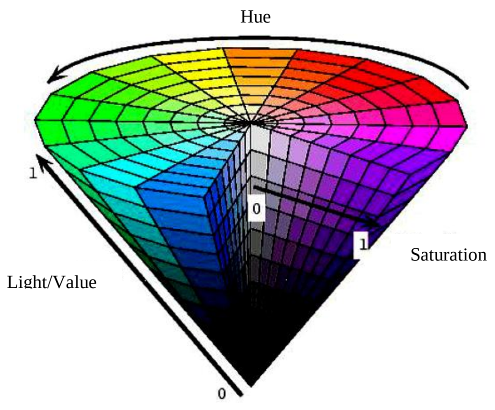


Figure 4.9: HSL colour space.

The conversion of RGB to HSV space can be made using the following formula [17]:

$$H = \begin{cases} \dfrac{(G-B)}{Max(R,G,B) - Min(R,G,B)} \, if \; R = Max(R,G,B) \\ 2 + \dfrac{(B-R)}{Max(R,G,B) - Min(R,G,B)} \, if \; G = Max(R,G,B) \\ 4 + \dfrac{(R-G)}{Max(R,G,B) - Min(R,G,B)} \, if \; B = Max(R,G,B) \end{cases}$$

$$H = H \; x \; 60 \; if \; H < 0 \; then \; H = H + 360$$

$$S = \frac{Max(R,G,B) - Min(R,G,B)}{Max(R,G,B)}$$
$$V = Max(R,G,B)$$

Hue is represented by an angle ranging from 0 to 360 degrees. Saturation is the radius of the cone ranging from 0 to 1 where 0 is a gray image and 1 is the pure colour. The Lightness/Value uses the vertical axis of the cone for representation and ranges from 0 to 1. The minimum value is black and maximum value represents white.

4.4.5 Correlation Methods

The problem of measuring similarity is quite common in image processing. This matching could be between two images or between small segments of the images. Correlation methods can help in determining the degree of similarity between two objects. In this section some normalized and non-normalized correlation methods are discussed.

4.4.5.1 Non-Normalized Correlation Methods

Some common non-normalized correlation methods are described in this section. A simple correlation method is *cross-correlation* [34] function g(x, y) = x . y, defined as:

$$CC(I_1, I_2) = \sum_a \sum_b I_1(a,b) \; . \; I_2(a,b).$$

A large value of this function determines high similarity between images I_1 and I_2. A normalized version of this function is usually preferred, as explained in the next section.

Another commonly used correlation method is *Sum of Squared Differences (SSD)* [34] that is defined as follows:

$$SSD(I_1, I_2) = \sum_a \sum_b (I_1(a,b) - I_2(a,b))^2.$$

This measure gives an error estimate of similarity, so the closer the value is to zero the greater similarity exists between the images. A variant of this error estimate is *Sum of Absolute Differences (SAD)* [34] which is not very sensitive to different lightening conditions in contrast to SSD. This can be defined as:

$$SAD(I_1, I_2) = \sum_a \sum_b |I_1(a,b) - I_2(a,b)|.$$

4.4.5.2 Normalized Correlation Functions

A simple way of achieving better invariance for similarity measures is by using normalized correlation functions. It is always safe to use normalized similarity measures to account for light variations. In order to normalize an image, the arithmetic mean is calculated for the image:

$$A = \frac{1}{n^2} \sum_a \sum_b I(a,b)$$

Normalization is then performed by subtracting the mean from the images to be correlated to get the normalized images N_1 and N_2. The computation of '*normalized zero mean cross correlation'* [34] is done as follows:

$$NCC(I_1, I_2) = \sum_a \sum_b (I_1(a,b) - N_1) \cdot (I_2(a,b) - N_2)$$

Similarly, the normalized version of '*Sum of Squared Differences'* and the '*Sum of Absolute Differences*' can also be computed [34]:

$$NSSD(I_1, I_2) = \sum_a \sum_b ((I_1(a,b) - N_1) - (I_2(a,b) - N_2))^2.$$

$$NSAD(I_1, I_2) = \sum_a \sum_b |(I_1(a,b) - N_1) - (I_2(a,b) - N_2)|.$$

4.4.6 Homography

In order to describe spatial changes in images or objects in images, 2D *homography* [34] is used. 2D homography can be used to estimate the image after distortion, rotation or scaling. The most important and relevant application of 2D homography is the projection of object coordinates in a model image to the test image. A mapping between the original coordinates (*a*, *b*) of an image to new coordinates ($a^{'}$, $b^{'}$) using homography can be done as follows [34]:

$$a' = \frac{v_1 a + v_2 b + v_3}{v_7 a + v_8 b + v_9}$$

$$b' = \frac{v_4 a + v_5 b + v_6}{v_7 a + v_8 b + v_9}$$

where $v_1, v_2, \dots v_9 \in \Re$ are the homographic parameters. In the form of homogeneous coordinates this can be written as [34]:

$$\begin{pmatrix} a's \\ b's \\ s \end{pmatrix} = \begin{pmatrix} v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 \\ v_7 & v_8 & v_9 \end{pmatrix} \begin{pmatrix} a \\ b \\ 1 \end{pmatrix}$$

and also in abbreviated form as [34]:

$$\boldsymbol{x}' = A\boldsymbol{x}.$$

Applying inverse transformation to get inverted matrix [34]:

$$\boldsymbol{x} = A^{-1}\boldsymbol{x}'$$

This equation for the homogeneous coordinates is used to get new image coordinates a', b' by dividing the third entry.

4.4.7 Principal Component Analysis

In image processing, Principal Component Analysis (PCA) [34] is used to transform the coordinate space into lower dimensions. The first coordinate after the projection is called the *principal component* which has the greatest variance compared to other coordinates. In order to describe the process of reducing dimensionality using PCA, let us consider N feature vectors $\boldsymbol{f_k}$ for k=1,2, ... ,N and $\boldsymbol{f_k} \in \Re$. The transformation function g:$\Re^n \rightarrow \Re^m$ with target dimension m < n , is used to reduce the dimensionality with minimal loss of information. The final result of the algorithm is a matrix $M \in \Re^{m \times n}$ in lower dimensional space.

The feature vectors $\boldsymbol{f_k}$ are normalized by subtracting the mean from each coordinate of the feature vector. The mean adjusted data is computed by:

$$\boldsymbol{f'_k} = \boldsymbol{f_k} - \bar{\mathbf{f}}$$

where

$$\bar{\boldsymbol{f}} = \frac{\mathbf{1}}{\boldsymbol{N}} \sum_{\boldsymbol{k}=\mathbf{1}}^{\boldsymbol{N}} \boldsymbol{f_k}.$$

A covariance matrix C is computed using adjusted features in a matrix $F = (\boldsymbol{f}'_1, \boldsymbol{f}'_2, \dots, \boldsymbol{f}'_k)$ as follows:

$$C = FF^T$$

As a next step, feature vectors are transformed to eigen space by computing eigenvectors and eigenvalues. Singular Value Decomposition (SVD) is used to calculate the eigenvalues and eigenvectors of the covariance matrix C since it is a positive singular matrix. The eigenvalue decomposition can be represented as follows:

$$C = VUV^T$$

where U is a square diagonal matrix containing the entries of eigenvalues λ_k of *C* and V is a square matrix whose columns are eigenvectors $\mathbf{v_k}$ of *C*:

$$V = (\boldsymbol{v_1}, \boldsymbol{v_{2,\dots,}} \boldsymbol{v_k})$$

$$U = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{pmatrix}$$

The final transformation matrix in lower dimension *m* is formed by the first *m* eigenvectors corresponding to the greatest eigenvalues. The eigenvalues are sorted in descending order $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ and the transformation matrix M in lower dimension is computed as:

$$M = (\boldsymbol{u_1}, \boldsymbol{u_2}, \dots, \boldsymbol{u_m})^T$$

CHAPTER 5: EMPIRICAL STUDY AND PROTOTYPES

This section describes the experiments performed to answer the research questions of the thesis.

5.1 Vision Related Study

The investigation of a robotic vision system for object recognition has been conducted by an organized empirical study. Each experiment is described in a respective section along with the data collection mechanism. The input for vision experimentation is still frames grabbed from the recording of the scenes obtained from the camera mounted on the robot. The aim is to determine which image processing algorithms may be effective, without initially taking into account how those algorithms could be optimised or implemented for real-time execution. The output is a performance measure that is discussed more specifically in the detailed description of the experiment. The image samples are selected randomly while keeping the size of each dataset the same. The data is filtered and any corrupted or noisy image due to jitter in the wireless communication is carefully omitted from the selection.

5.1.1 Experiment 1

In order to measure the performance of the vision system, an experiment is performed. The purpose of the experiment is to determine how precisely and accurately the selected vision algorithms behave. This will help in determining whether the proposed combined algorithm works effectively in the controlled lab environment. It can also be beneficial for knowing the limitation of the algorithms used for the vision system of the robot.

The experiment is performed on the floor of the GSIL lab in an offline manner using Lego bricks as the target objects. The robot used in the experiment is a four wheel differential drive robot with a camera fixed on the turret. The images taken by the robot are transmitted to a remote PC by a wireless CCD camera. The robot is placed close to target objects with the camera tilted down approximately at 130 degrees from the vertical and images are taken that are later used for performance measurement. The performance evaluation of object detection techniques is usually performed using precision and recall rate of the specific technique along with some combining measure such as harmonic mean or F-score [47-50]. The recall is used to determine how often an algorithm detects an object and precision tells whether the detection is correct. A combined measure such as F-score is used to measure the overall performance of the object detection algorithm.

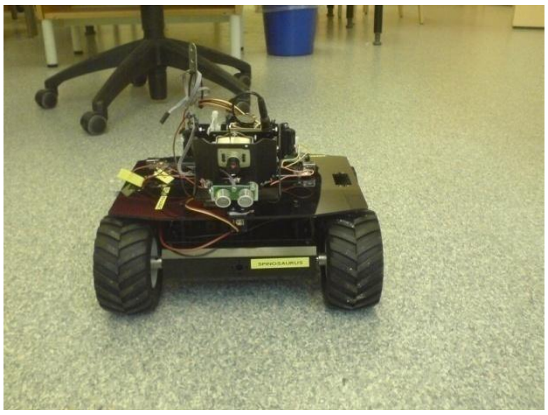


Figure 5.1 Front view of the robot.

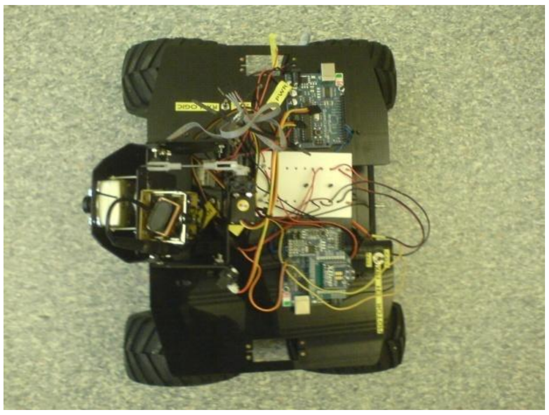


Figure 5.2 top View of the robot.

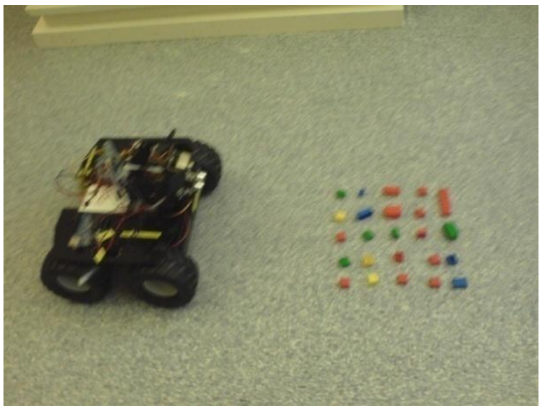


Figure 5.3 Robot in the experiment.

A total of 750 images of size 640x480 are taken by camera at 15fps and assembled into five datasets D1 to D5. Each dataset consisted of 150 images extracted from a video duration of 10 seconds. The average values of precision, recall and F-measure are computed from averages of each dataset.

There are 62 Lego bricks of four colours (Red, Blue, Green, and Yellow) and five different sizes which are used in the experiment. Twenty five target objects for each dataset are placed in front of the robot spaced at approximately 2-4cm apart from each other. Objects in each dataset are arranged in different positions. Variations in orientation and lighting are made for each dataset.

Each participating detection method (Histogram, TrainedShapes, PCA, SURF, HistSurf) is applied on each dataset in order to determine the performance of object detection. The average values are used to compare the individual performance of the participating methods. In order to determine the significance of the results, a pairwise t-test with 95% confidence interval is used. The null hypothesis H_0 assumes that the performance of 'HistSURF' is the same as that of the other techniques, while H_1 assumes that there is an improvement in the performance due to the use of the proposed hybrid algorithm. The results from the experiment are shown in the following chapter.

5.1.2 Experiment 2

The performance of HistSURF depends on the number of features detected by the SURF algorithm. In SURF feature detection, the features extracted from a model image are matched against features extracted from a target image. A certain threshold of these matched features is used as the criterion for detection output for the SURF part of the HistSURF algorithm as explained in the methodology section.

An experiment is conducted to determine the effect of distance on the number of matched SURF features. A total of 250 images at four different distances (25cm, 40cm, 50cm, 60cm), with 50 images for each distance, are used in the experiment. The images are taken from the video clips recorded by placing the target object at the specified distances from the robot. The experiment could help to determine the effectiveness of HistSURF as well as determining the limitations of the technique.

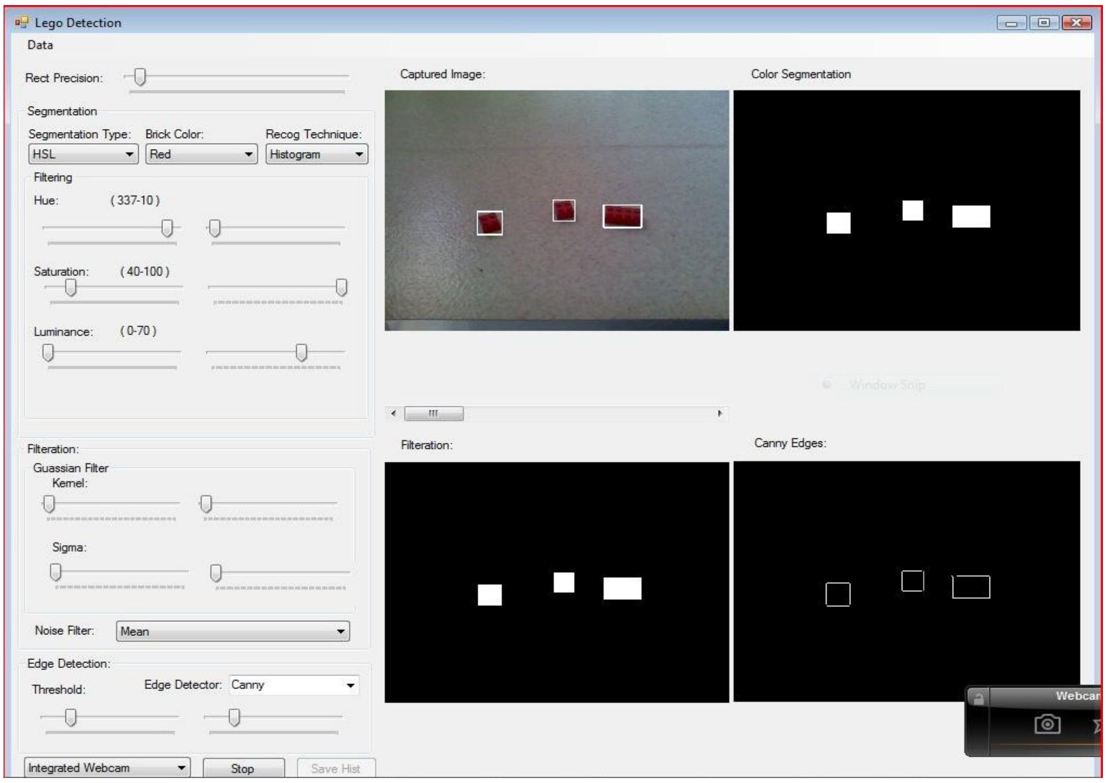
5.2 Vision Prototype

In order to perform the experiment and evaluate the different algorithms, a prototype for vision processing software system has been made. In this prototype, participating vision techniques are implemented for the detection of target objects. The approach used for the development of the prototype is the ‘evolutionary prototyping approach’ since it is a scientific experiment project in which requirements are not clearly available at the start and are subject to change with increasing understanding through prototype iterations. The software is updated in an incremental fashion depending on the new requirements. A screenshot of the software is given in figure 5.4.

The vision prototype is responsible for real-time object detection using the techniques discussed earlier. The process of object detection starts by initiation from the user, after that it is a completely autonomous process. However, some of the segmentation parameters can be tweaked during or before the initiation of detection process. Possible actions that a user can take are visible in the use cases given in figure 5.5. Some of the functions of the prototype need user input, like evaluation procedure. A flow of information between the various systems and coordination among them can be visualized in figure 5.6.

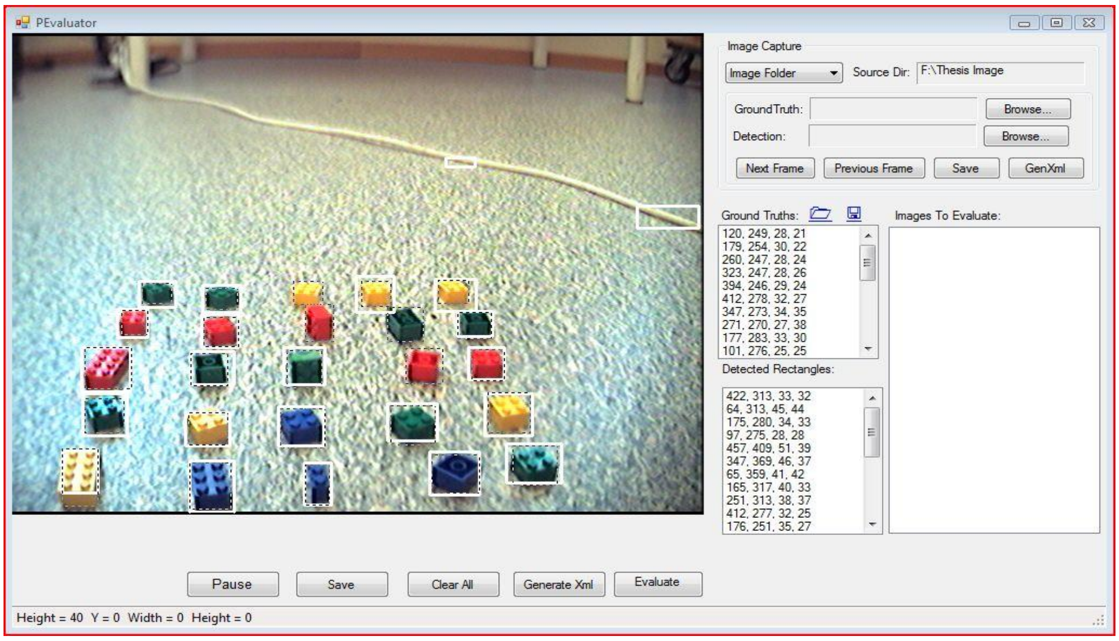
The vision framework of the prototype is mentioned in section 4.2 of the report. This vision prototype is implemented in C#.NET. The image processing part of the software makes use of two open source computer vision libraries, Aforge.NET (www.aforge.net) and EmguCV (www.emgu.com). EmguCV is a .NET wrapper for Intel’s OpenCV library. A high level function diagram of the vision processing is depicted in figure 5.7.

An evaluation interface shown in figure 5.4 is used to evaluate the techniques. The evaluation system takes image source and ground truth rectangles as input and generates an output text file that contains detection counts for the selected vision technique. These are then used to generate precision and recall graphs.



Segmentation Type: HSL
Brick Color: Red
Recog Technique: Histogram
Hue: (337-10)
Saturation: (40-100)
Luminance: (0-70)
Noise Filter: Mean
Edge Detector: Canny
Integrated Webcam

Figure 5.4(a) : Detection of Lego Bricks in progress.



Source Dir: F:\Thesis Image
GroundTruth:
Detection:

Ground Truths:

```
120, 249, 28, 21
179, 254, 30, 22
260, 247, 28, 24
323, 247, 28, 26
394, 246, 29, 24
412, 278, 32, 27
347, 273, 34, 35
271, 270, 27, 38
177, 283, 33, 30
101, 276, 25, 25
```

Detected Rectangles:

```
422, 313, 33, 32
64, 313, 45, 44
175, 280, 34, 33
97, 275, 28, 28
457, 409, 51, 39
347, 369, 46, 37
65, 359, 41, 42
165, 317, 40, 33
251, 313, 38, 37
412, 277, 32, 25
176, 251, 35, 27
```

Images To Evaluate:

Height = 40 Y = 0 Width = 0 Height = 0

Figure 5.4(b) : Evaluation of Lego Bricks in progress.

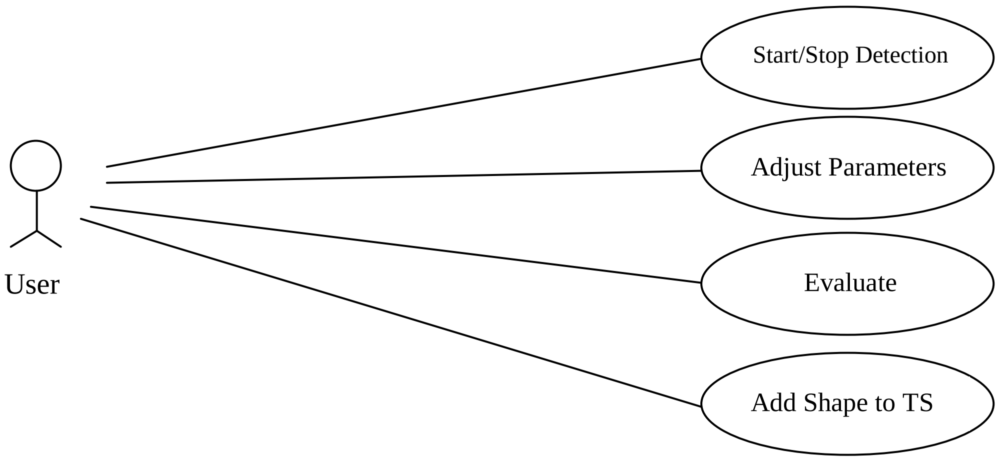


Figure 5.5: Use case diagram of LegoDetect.

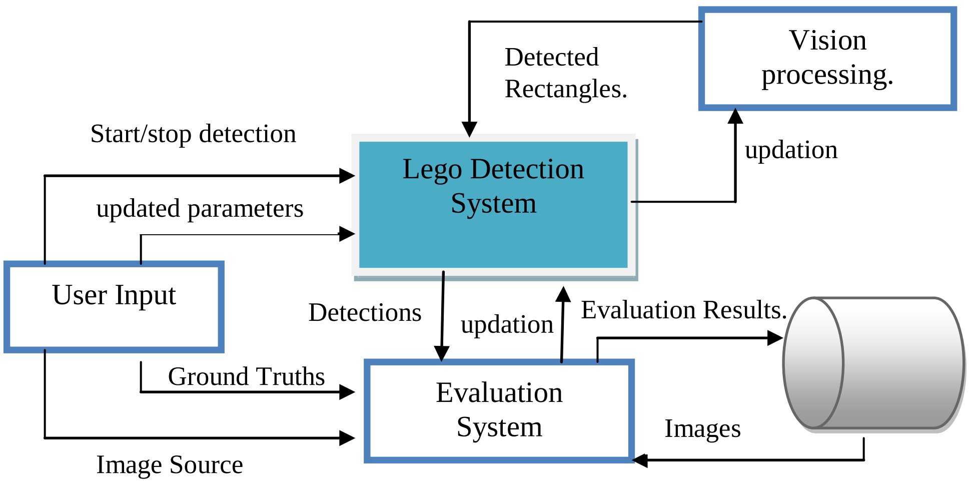


Figure 5.6: Data flow of the vision prototype.

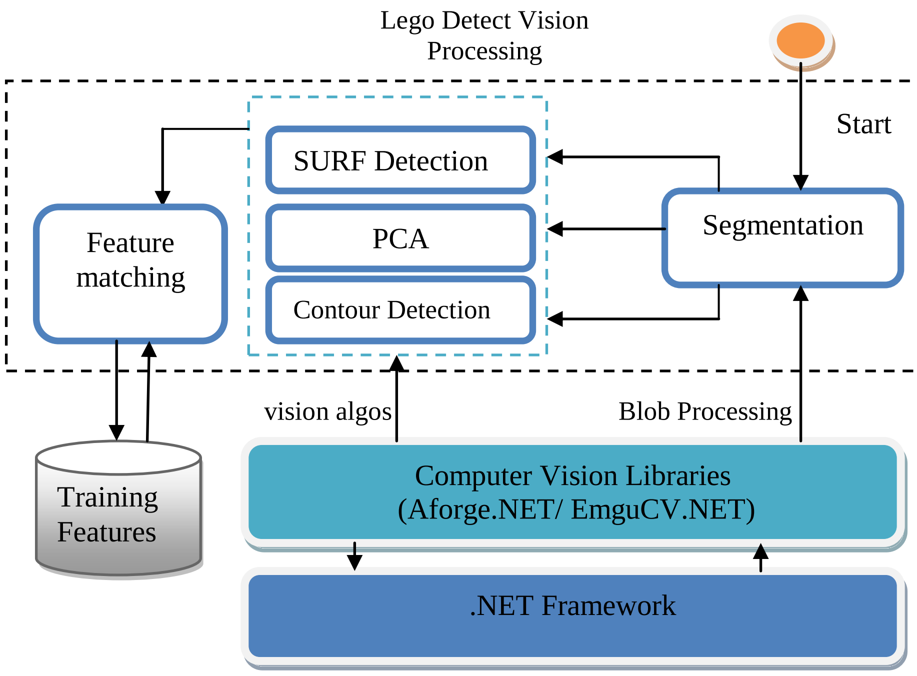


Figure 5.7: Vision processing system of Prototype.

5.3 Navigation Experiment

In this section, the design of the robot navigation experiment is explained. The purpose of the experiment is to determine whether the use of a Gyro helps in attaining the correct desired movement. The experiment is performed on detection of the turn angle of the robot and the performance is compared with the output results obtained from the velocity and time model as explained in the methodology section.

The experiment is performed in the GSIL lab with a controlled environment. The input of the experiment is the desired angle and output are the time and angle attained by robot. A fault is calculated from the readings of desired and attained angles. The experiment is repeated two times, once with the Gyro that is responsible for deciding the final angle and the second time a speed model is used for deciding the attainment of the desired angle.

The robot is asked to turn at various angles and the total time and attained angle is measured. Each time, the robot is initially placed at a fixed location and readings are taken. In the case of spinning around the vertical axis of the robot, the speed of the both channels is kept the same but in opposite directions. In the case of turning with a larger diameter (i.e. during translational movement), different speeds on each channel are used. The angle acquired by the robot and the time taken to complete the movement are measured. Three readings for each angle are taken and the average is used to calculate the fault coefficient.

In order to use the Gyro it is also necessary to observe its behaviour so that line noise can be taken into account. For that purpose, Gyro readings are taken by keeping the robot stationary as well as while it is in motion. All the results of the experiment are explained in the next section.

5.4 Navigation Prototype

A navigation prototype has also been developed that consist of embedded code for the microcontroller on the arduino board. The language used is ‘arduino’ that is an effort to provide a high level abstraction layer for embedded programming (it is a variant of the C programming language). The source code used is included in the Appendix section of this document. In addition to obstacle avoidance and search moves which are performed by this prototype, it is also responsible for handling remote commands.

The software makes use of two libraries (Servo and SerialSoft) that are part of the arduino environment. Apart from using code examples that are included with the arduino environment, all software logic is developed in this study. The libraries are used to communicate with the hardware while a high level decision making structure is implemented as part of this study which can be seen in figure 5.8.

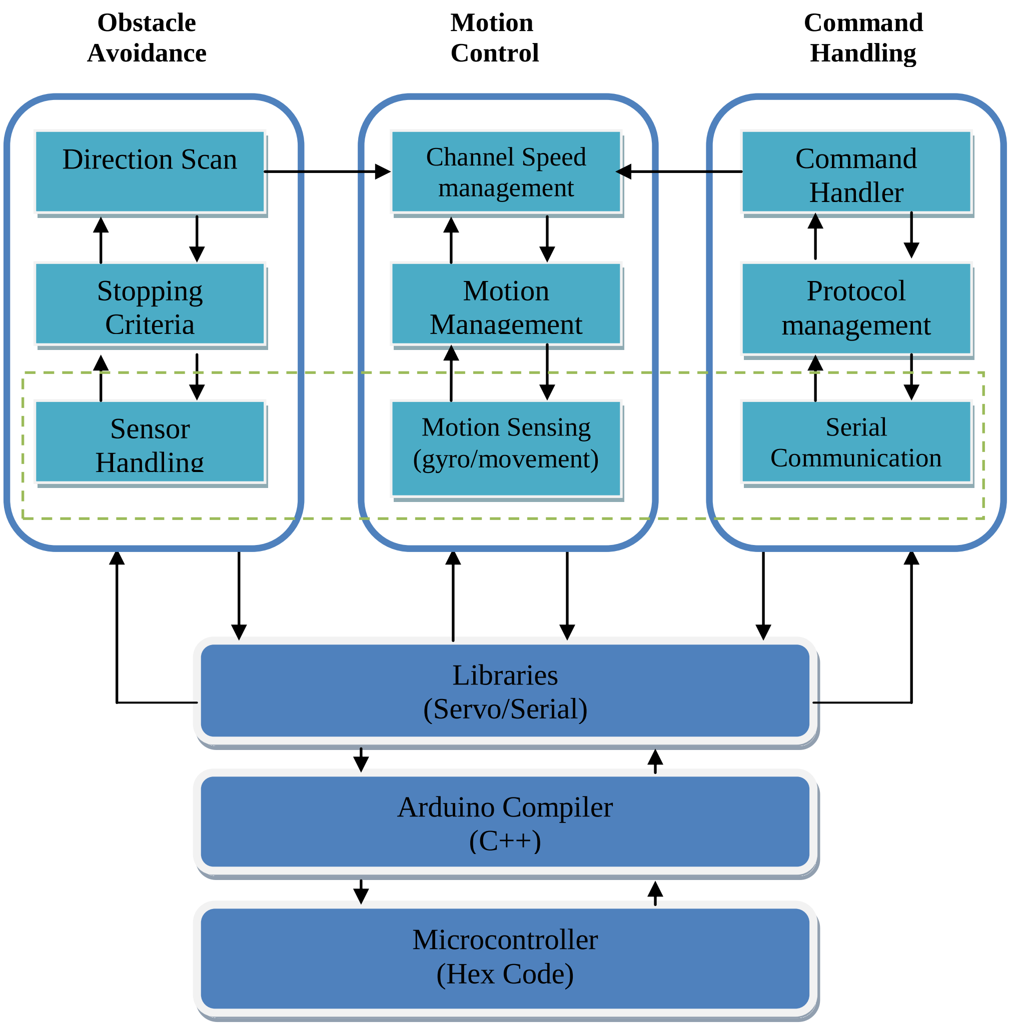


Figure 5.8: Functional diagram of navigation prototype.

The obstacle avoidance system depends completely on the input from the sensors while the motion control system partly depends on when motion sensing is performed by Gyro measurement. The low level functionality like input readings of the sensors is done with the help of arduino libraries and a high level decision framework is implemented based on those

readings. The sensor readings are converted to meaningful distances using simple calculations and then those are averaged to get stable measurements for better decision making.
The flow control of the navigation prototype is based on three main functions which are depicted in figure 5.8. There is coordination among the different functions of the prototype in a non blocking fashion. The functionality is divided into steps and each step is executed for one cycle of the microcontroller. For example, as part of this functionality, the turret is moved in a scanning fashion all the time in order to detect front obstacles as well as objects in the environment. This can block the other sensing systems if executed in a blocking way but it doesn't block in particular implementation because only part of the turret is moved in each execution cycle of the microcontroller. An overall flow control and coordination between the functions is given in the figure 5.9.

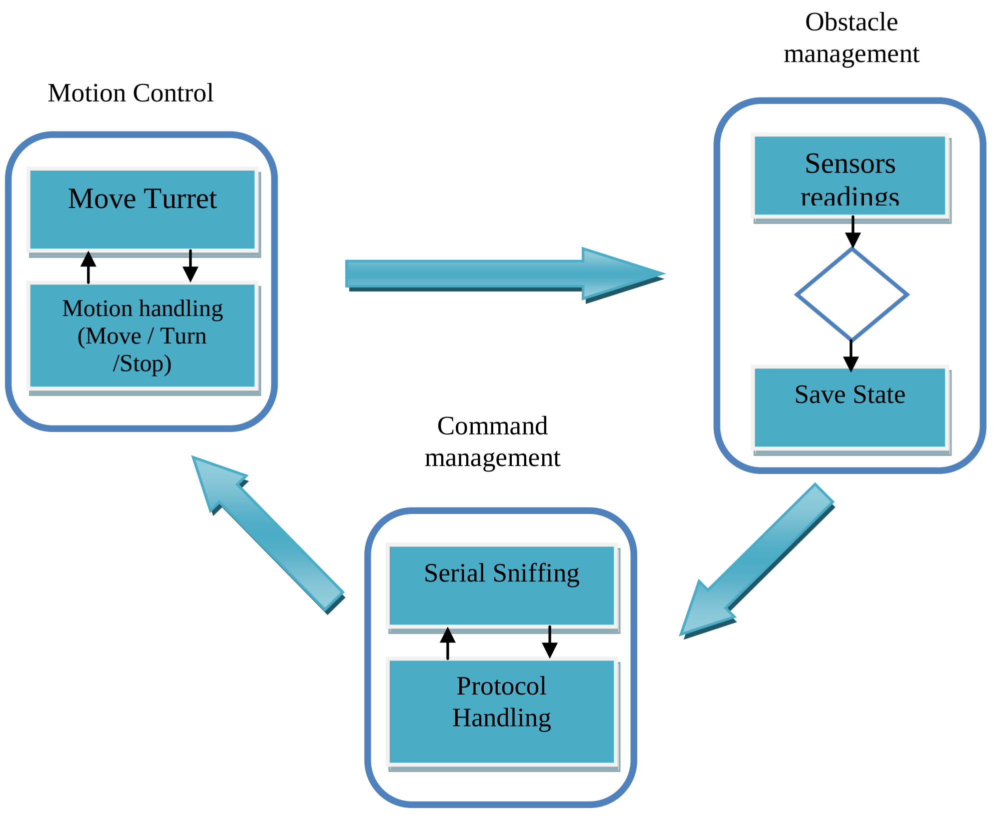


Figure 5.9: Functional flow of navigation prototype.

Chapter 6: Results

In this section, the results obtained from the experiments described in the previous section are presented. In the following, first the results from the vision experiment(s) will be described and then results of the navigation experiments will be presented.

6.1 Vision Results

An experiment is performed in an offline manner to determine the performance of each vision algorithm. Ground truth rectangles are obtained by selecting object regions manually in the pictures. A careful selection for the object regions is made and saved for future comparisons. A cropped portion of the image showing the ground truth rectangles and detections is depicted in figure 6.1.

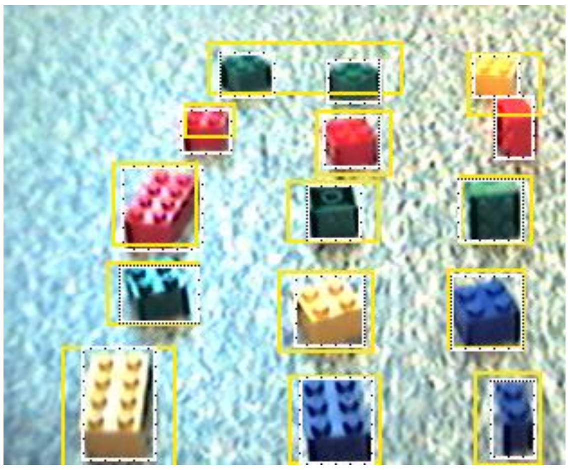


Figure 6.1: Ground Truths with dotted rectangles and detection with solid lines.

These ground truth rectangles are compared with the detected rectangles obtained by each vision algorithm. An object is considered as detected if the overlap region between the ground truth rectangle and the detected rectangle is eighty percent of the area of the ground truth rectangle. The relative true positive, false positive and false negative rates are determined for each image against each algorithm. A trend line of precision and recall for each algorithm is drawn as shown in figure 6.2.

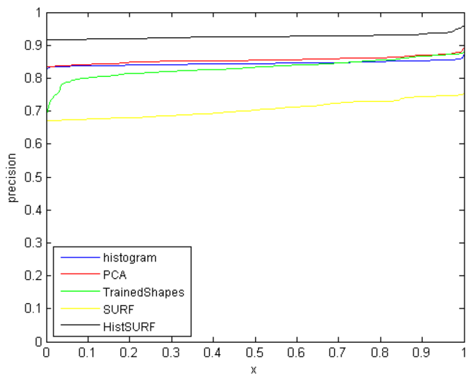
The results of precision and recall obtained for each dataset (D1-D5) are averaged and are presented in a tabular form as shown in table 6.1. In order to get a combined measure of performance, an F-score is used. A balanced F- score combines precision and recall with equal weight. The F-score F_α, also called the harmonic mean of the precision and recall, is described as follows:

$$F_{(1+\propto^2)} = \frac{2 * (precision * recall)}{\propto^2 * precision + recall}$$

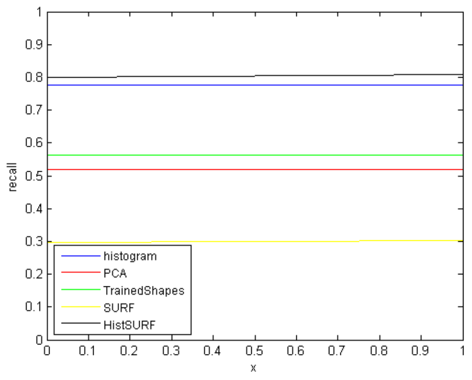
where α is the weight that determines the preference of precision over recall or vice versa. For α=1, the F-score is said to be balanced because both precision and recall are equally weighted. Two other commonly used F-scores are F_2 which weights recall twice as much as precision and $F_{0.5}$ which weights precision twice as much as recall.

Table 6.1 : Results of the vision experiment.

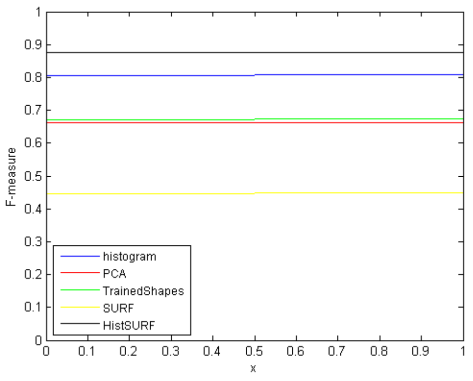
	Avg Precision	Avg Recall	Avg F-score (F_1)	Avg F-score ($F_{0.5}$)
Histogram	0.844	0.826	0.840	0.834
PCA	0.854	0.643	0.799	0.731
TrainedShapes	0.833	0.748	0.809	0.781
SURF	0.705	0.427	0.617	0.522
HistSURF	0.926	0.874	0.914	0.898



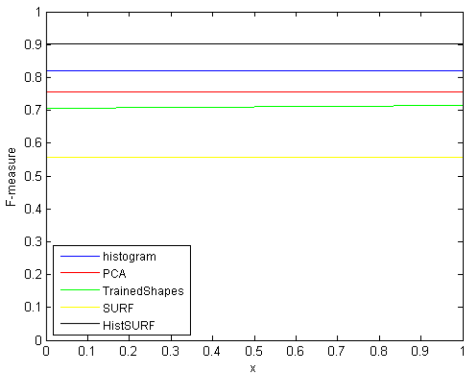
(a)



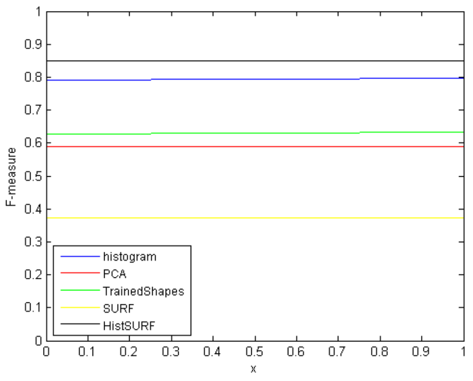
(b)



(c)



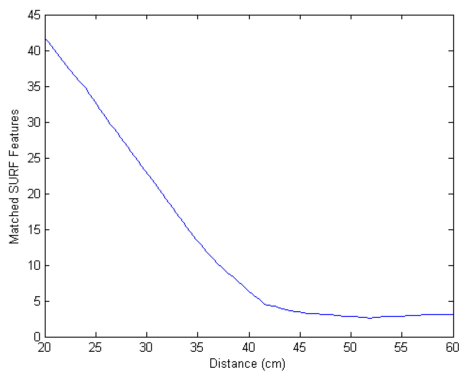
(d)



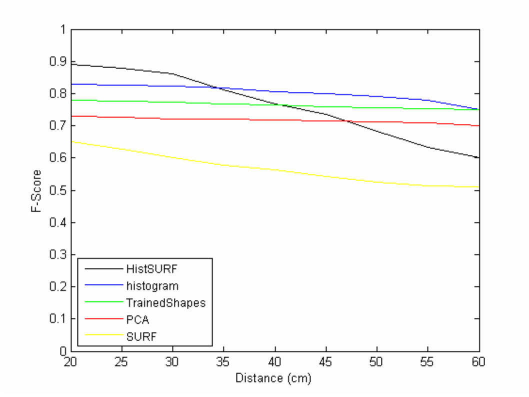
(e)

Figure 6.2: (a) precision graph for each algorithm, (b) Recall graph (c) Balanced F-measure α =1 (d) $F_{0.5}$ measure (e) F_2 measure.

The result of the experiment conducted in order to find the relation between the matched SURF features and the distance of the camera from the target object is presented in the form of a graph as shown in figure 6.3. All images that are used in the experiment are of size 640x480 and are taken from the camera that uses PAL-BG encoding. It is to be noted that the camera used in the experimentation has manual focus with no zoom capabilities. The horizontal and vertical angles of view of the camera are 58° and 40° respectively.



(a)



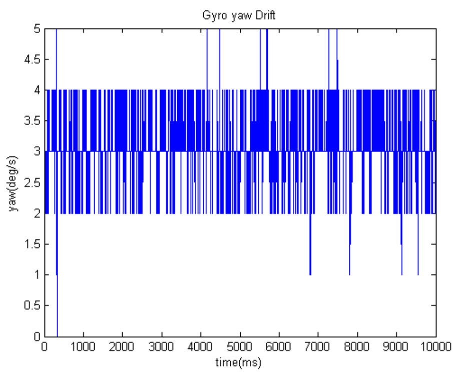
(b)

Figure 6.3: (a) Effect of distance on SURF features (b) Distance effect on performance.

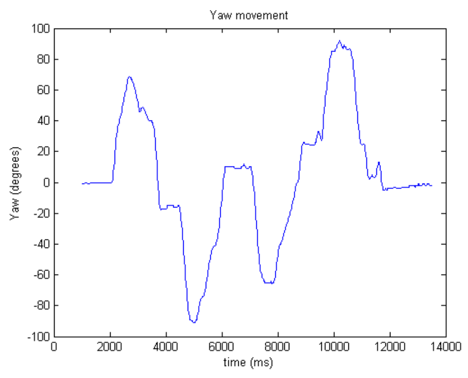
It is clear from the result that number of matched features reduces significantly with the increase in distance from the target object. This implies that detection rate would increase and would stabilize if the object is near to the camera. In comparison with other detection techniques, the detection rate of the HistSURF is more affected by the variation in the distances.

6.2 Robotic Locomotion Results

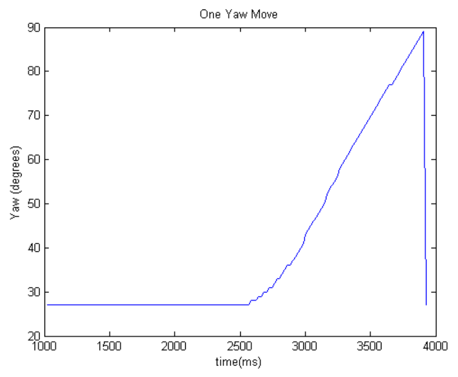
In this section results obtained from the experiment performed on robotic movement sensing are presented and described. An experiment has been carried out in order to determine the benefit of using a Gyro in measuring the absolute angle of the robot. In order to determine the Gyro behaviour, some readings are plotted while keeping the robot stationary. This can be seen in figure 6.4 (a). It is quite visible that the Gyro contains a small noise signal on the line but this can be neglected since it is quite steady noise and can be removed by simple thresholding. But this is not the same when the Gyro is in movement. Another set of readings are obtained by creating a predefined move. A move is made by turning the robot 90° towards the left and then rotating it 90° towards the right. As it can be seen from figure 6.4(b), each move of the robot is followed by the formation of a very strong offset in the opposite direction. Another result obtained from the single move is presented in figure 6.4(c) which indicates that the jitter can be isolated and the original move can be detected by keeping a gap between the individual moves.



(a) Gyro is stationary.



(b) 90° left and 90° right.



(c) One Gyro move.

Figure 6.4: Gyro noise and drift at stationary and at movement.

The results obtained from the experiment performed using the Gyro for the absolute angle measurement, are presented in table 6.5. The results of the speed model are presented in table 6.3. In order to compare the fault values from the both methods, an absolute error measure '*E*' is taken which is given as follows:

$$E = \sum_{D} \frac{|F_D|}{n}$$

where 'F_D' is the fault value for each desired angle and '*n*' is the number of angles taken in the experiment. This results in E_g=0.18 for the Gyro and E_m=0.06 for the speed model. This shows that angles attained through the speed model have smaller error rates for the case of the spinning turn.

Table 6.2: Results of the Gyro angle measurement (D=desired angle,A=attained angle,t=time of movement).

Desired Angle (D°)	**t_1 (sec)**	**t_2 (sec)**	**t_3 (sec)**	**$t=\sum t_i/3$ (sec)**	**a_1°**	**a_2°**	**a_3°**	**$A=\sum a_i/3$**	**F= (A/D)-1**
30	0.28	0.28	0.29	0.28	25	20	23	22.67	-0.24
45	0.52	0.52	0.52	0.52	57	55	54	55.33	+0.23
60	0.70	0.73	0.71	0.71	70	75	73	72.67	+0.21
80	1.00	0.96	0.95	0.97	85	89	90	88.00	+0.10
90	1.10	1.12	1.20	1.14	95	97	100	97.33	+0.08
135	1.78	1.77	1.80	1.78	155	150	160	155.00	+0.15
180	2.59	2.50	2.60	2.56	210	200	230	213.33	+0.20
225	3.57	3.60	3.59	3.59	280	287	290	285.87	+0.27

Table 6.3: Results of the angle measurement using model (D=desired angle,A=attained angle,t=time of movement).

Desired Angle (D°)	t_1 (sec)	t_2 (sec)	t_3 (sec)	$t=\sum t_i/3$ (sec)	$a_1°$	$a_2°$	$a_3°$	$A=\sum a_i/3$	F= (A/D)-1
30	0.33	0.34	0.31	0.33	28	28	26	27.33	-0.08
45	0.51	0.54	0.52	0.52	42	40	41	41.00	-0.08
60	0.66	0.69	0.68	0.68	57	58	60	58.33	-0.03
80	0.88	0.86	0.87	0.87	79	75	78	77.33	-0.03
90	1.00	1.00	1.00	1.00	90	87	90	89.00	-0.01
135	1.50	1.51	1.50	1.50	125	122	127	124.67	-0.08
180	2.00	2.10	1.99	2.00	170	169	160	166.33	-0.08
225	2.50	2.47	2.51	2.50	210	195	205	203.33	-0.10

Figure 6.5 shows a comparison of the angle measurement techniques with respect to time taken by each method . This indicates that Gyro measurement suffers from noise and drift problem which is not the problem with simple speed model being used.

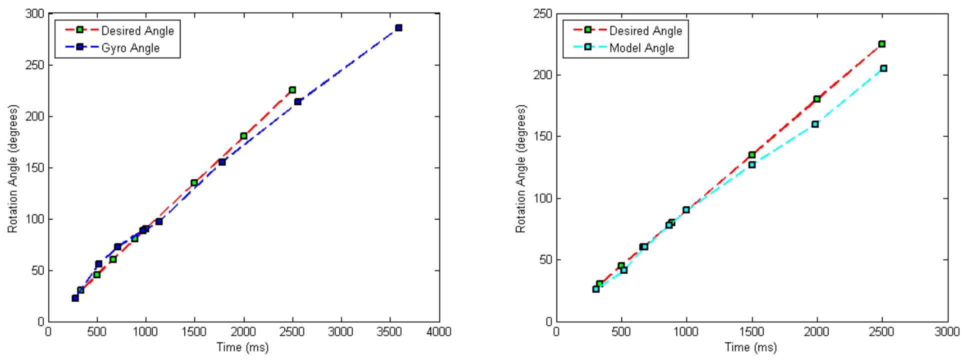


Figure 6.5: Comparison of Gyro and model angle measurement in reference to desired angle.

The results of turning with large diameter when the wheel speeds are different are presented in table 6.4. The robot is moved at various speeds with the desired turn angle fixed at 90° and the diameter of the achieved angle is measured. Measurements with the Gyro for the large diameter case are not presented since it resulted in erroneous angles due to drift that accumulates over time.

Table 6.3: Turning with large diameter.

No.	Approx. Achieved Diameter (cm)	V_R (cm/sec)	V_L (cm/sec)	Rotation Time (ms)
1	37	60	85	3970
2	28	60	109	2440
3	19	60	154	1522
4	24	109	154	1990
5	22	85	109	1640

CHAPTER 7: ANALYSIS AND INTERPRETATION

In this section, an analysis on the results presented in the previous section is made and interpretations from the results are presented.

The results from the vision experiment are presented in the form of tabular data and also in the form of graphs in the previous section. The average values indicate that HistSURF and PCA are better in precision while HistSURF and Histogram are better in recall. There is always a trade-off between precision and recall, so a combined measure can explain better. The combined measure (F-score) with balanced weight as well as weighting precision twice as much as recall, suggest that HistSURF achieves a better overall detection rate.

In order to determine the significance of the results, a pairwise t-test is used with a 0.05 confidence level since the result is an average of individual values obtained from five datasets. The test is repeated for all participating techniques keeping ‘HistSURF’ as the reference technique. The results of the analysis can be seen in the table 7.1.

Table 7.1: Results of pairwise t-test on F-scores.

HistSURF	$t_{4,0.05}$	p-value
Histogram Comp.	5.71	4.6e-3
Trained Shapes	7.60	1.6e-3
PCA	10.52	4.6e-4
SURF	20.12	4.0e-5

The results from analysis show that null hypothesis can be rejected for each comparison which means ‘HistSURF’ performs significantly better than the other tested object detection techniques.

The results from the vision experiment conducted to determine the relation between distance and number of matched features indicate that the performance of SURF depends on the distance from the object. This implies that performance of the HistSURF will also depend on the distance from the target object. The participation of SURF in the hybrid technique can be adjusted by changing the value of the combining factor as explained in the methodology section.

Although all measures are favouring the new detection algorithm, there are still limitations of the vision system. Since the proposed algorithm is also based on image segmentation that itself is an unsolved problem in general, it also suffers from lighting variations and is not very robust at estimating the pose of an object. Training of shapes does not help much as can be seen in the results, because of its dependence on the pre-processing stage that involves colour segmentation. The SURF algorithm alone performs poorly since most of the spatial information is lost in the process of segmentation. The combined algorithm HistSURF performs better than the other participating methods but is still vulnerable to distance from the camera. The accuracy of HistSURF is good if the distance from the camera is small.

A major advantage for object detection from images obtained from a camera mounted on a mobile robot is that the motion of the robot can be used to decrease the distance to objects in order to increase the accuracy of the HistSURF technique. Decreasing object

distance requires effective motion control and navigation, which in turn depends upon good control of orientation and rotational motion. The motion experiments on the robot indicate that the Gyro alone is not a very accurate measure, for the selected Gyro. There is Gyro drift that creates difficulty in measuring the correct angle. However, a Gyro with a good high pass filter to suppress noise along with a magnetometer can be used to achieve better accuracy [51].
The absolute average error coefficient measure defined in the results section show that the Gyro solution has an 18.3% error rate while the speed model solution has a 6.1% error rate. This clearly indicates that the Gyro has failed to improve the error rate. It can also be seen from figure 6.5 that the Gyro deviates more from the desired time and angle line than the speed model.

Future Work:

There can be many future directions keeping in view that object detection is used as an initial step in vision based solutions. An important direction for ongoing work could be to use object recognition as a tool for obtaining visual perception (e.g. 3D Perception) in mobile robots. This can lead to a study of visual servoing in which interaction with the environment could be studied. Object tracking could also be used to provide additional data for robot motion and position determination. There has been a recent activity regarding object recognition which involves using of Bag-of-Visual-Words for the classification of objects in the scene. Although it has been successful in offline testing performed on the large databases but it would be interesting to see its implications for the real-time applications such as visual manipulation for mobile robot. Another important direction could be studying the use of object recognition for the visual SLAM (Simultaneous Localization and Manipulation) of mobile robots.

The motion systems of the robots usually use more than one measuring devices for the determination of correct orientation and attaining the balance (e.g. in case of UAVs). In the case of more than one measuring devices (e.g. a Gyro and a magnetometer), Kalman filters can be used to reduce the error in one of the measuring devices and a more reliable estimate of the true angle attained by the robot can be determined [52].

Summary

There are many functional systems in a human that are needed to be mimicked in humanoid and other robots. One such system is a vision system that is often used for detecting objects and understanding scenes. Vision processing has been a primary focus in robotics and some very successful solutions have been fielded for the automation of tasks in industry. Although there has been enormous research in solving vision related problems, every solution has its application in limited contexts. This study focuses on the problem of identifying small sized coloured objects placed in a scene. The reliable detection of small sized objects is not without problems specifically in the context of mobile robots with limited processing power, obviating the need for off-board processing and a robust communications link between the robot and an image processing and object recognition host. The use of colour segmentation alone as a solution is not reliable because of the size of the objects and poor image quality that is usually the case with mobile robots.

There have been attempts at using hybrid techniques for the solution of challenging vision related problems. Such studies have indicated that combining key-point based techniques with colour segmentation based approaches yield better detection rates due to the reduction of false positives. A novel algorithm 'HistSURF' has been proposed and implemented in this study and is compared by experimental trials with other vision techniques. The results are favourable for the histogram/SURF combination and it can be deduced that for small sized coloured objects, it is advisable to combine a key-point based technique with colour segmentation to achieve a better detection rate.

Another important part of a mobile robot is its navigation system that is crucial for the completion of its autonomous tasks. It is very important for an intelligent robot to make sure that it moves as intended and can actively compensate for errors. This has been the focus of the navigation related experiment described in this thesis. An angle measurement has been taken as a benchmark and various angular positions are tested for two motion solutions. The hardware solution consisted of a Gyro which is a device that can determine the rotation rate of the body in which it is contained. This can be of great help in determining the correct orientation of the object but unfortunately, it turned out by experimentation that there is a problem of line noise as well as a variable drift that accumulates while calculating the angle attained by the Gyro for the particular (low cost) Gyro that was tested. An experiment conducted to compare the angle measurement by two solutions indicated that using a Gyro alone doesn't help in reducing the error in the movement of the robot, compared with dead reckoning. However, it is possible to attain the better results by using more than one measuring device, for example, using a Gyro along with a magnetometer or digital compass. This arrangement could then be used to get more accurate estimates of the orientation of the robot using Kalman filters.

An important direction for ongoing work could be to use object recognition as a tool for obtaining visual perception in mobile robots. This can involve visual servoing in which interaction with the environment could be studied. Object tracking could also be used to provide additional data for robot motion and position determination. However, these issues are beyond the scope of the current work.

References

[1] Lowe, D. G., 2004, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60(Copyright 2005, IEE), pp. 91-110.
[2] Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L., 2008, "Speeded-Up Robust Features (SURF)," Computer Vision and Image Understanding, 110(3), pp. 346-359.
[3] Ke, Y., and Sukthankar, R., "PCA-SIFT: A more distinctive representation for local image descriptors," Proc. Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, June 27, 2004 - July 2, 2004, Institute of Electrical and Electronics Engineers Computer Society, pp. II506-II513.
[4] Abdel-Hakim, A. E., and Farag, A. A., "CSIFT: A SIFT descriptor with color invariant characteristics," Proc. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2006, June 17, 2006 - June 22, 2006, Institute of Electrical and Electronics Engineers Computer Society, pp. 1978-1983.
[5] Grabner, M., Grabner, H., and Bischof, H., "Fast approximated SIFT," Proc. Computer Vision - ACCV 2006. 7th Asian Conference on Computer Vision. Proceedings, Part I, 13-16 Jan. 2006, Springer-Verlag, pp. 918-927.
[6] Viola, P., and Jones, M., "Rapid object detection using a boosted cascade of simple features," Proc. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 8-14 Dec. 2001, IEEE Comput. Soc, pp. 511-518.
[7] Donghoon, K., and Dahyot, R., "Face components detection using SURF descriptors and SVMs," Proc. 2008 International Machine Vision and Image Processing Conference (IMVIP), 3-5 Sept. 2008, IEEE, pp. 51-56.
[8] Ruihua, L., and Yanguang, W., "SAR image matching base on speeded up robust feature," Proc. 2009 WRI Global Congress on Intelligent Systems, GCIS 2009, May 19, 2009 - May 21, 2009, IEEE Computer Society, pp. 518-522.
[9] Shan, A., Xin, M., Rui, S., and Yibin, L., "Face detection and recognition with SURF for human-robot interaction," Proc. 2009 IEEE International Conference on Automation and Logistics (ICAL), 5-7 Aug. 2009, IEEE, pp. 1946-1951.
[10] Svab, J., Krajnik, T., Faigl, J., and Preucil, L., "FPGA based Speeded Up Robust Features," Proc. 2009 IEEE International Conference on Technologies for Practical Robot Applications. TePRA 2009, 9-10 Nov. 2009, IEEE, pp. 35-41.
[11] Zhang, N., "Computing parallel speeded-up robust features (P-SURF) via POSIX threads," Proc. 5th International Conference on Intelligent Computing, ICIC 2009, September 16, 2009 - September 19, 2009, Springer Verlag, pp. 287-296.
[12] Anderson, J. K., Iftekharuddin, K. M., Threlkeld, E., and Montgomery, B., "Single camera-based object detection and tracking for mobile robots," Proc. Optics and Photonics for Information Processing II, 13 Aug. 2008, SPIE - The International Society for Optical Engineering, p. 70720T (70712 pp.).
[13] Fujiwara, K., Peters, R. A., II, and Kawamura, K., "Colored-object detection for a mobile robot," Proc. Applications of Digital Image Processing XVII, 26-29 July 1994, pp. 457-469.
[14] Wajima, N., Takahashi, S., Itoh, M., Satoh, Y., and Kaneko, S., "Robust object detection based on radial reach filter for mobile robots," Proc. 2006 SICE-ICASE International Joint Conference, 18-21 Oct. 2006, IEEE, p. 4 pp.
[15] Xiaochun, W., and Wilkes, D. M., "Visual novel object detection for mobile robots," Proc. 2008 International Conference on Data Mining, 14-17 July 2008, CSREA Press, pp. 407-413.
[16] Yu, J.-x., Cai, Z.-x., and Duan, Z.-h., 2007, "Detection and tracking of moving object with a mobile robot using laser scanner," Chinese Journal of Electron Devices, 30(Copyright 2008, The Institution of Engineering and Technology), pp. 2301-2306.

[17] Chen, T.-W., Chen, Y.-L., and Chien, S.-Y., "Fast image segmentation based on K-means clustering with histograms in HSV color space," Proc. 2008 IEEE 10th Workshop on Multimedia Signal Processing, MMSP 2008, October 8, 2008 - October 10, 2008, Inst. of Elec. and Elec. Eng. Computer Society, pp. 322-325.
[18] Morshidi, M. A., Marhaban, M. H., and Jantan, A., "Color segmentation using multi layer neural network and the HSV color space," Proc. International Conference on Computer and Communication Engineering 2008, ICCCE08: Global Links for Human Development, May 13, 2008 - May 15, 2008, Inst. of Elec. and Elec. Eng. Computer Society, pp. 1335-1339.
[19] Calderon, C., Zhou, C., Kong Yue, P., Wong, M., Rajesh Elara, M., Yu, G., Keong Ang, C., and Wu, B., 2007, "Robo-erectus junior – a kidsize soccer playing humanoid robot," Dept. Electrical and Electronic Eng.,Singapore Polytechnic., Singapore.
[20] Bruce, J., Balch, T., and Veloso, M., "Fast and inexpensive color image segmentation for interactive robots," Proc. Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000), 31 Oct.-5 Nov. 2000, IEEE, pp. 2061-2066.
[21] Brown, M., and Lowe, D., "Invariant features from interest point groups," Proc. BMVC2002: British Machine Vision Conference 2002, 2-5 Sept. 2002, British Machine Vision Assoc, pp. 253-262.
[22] Diplaros, A., Gevers, T., and Patras, I., 2006, "Combining color and shape information for illumination-viewpoint invariant object recognition," IEEE Transactions on Image Processing, 15(Copyright 2006, IEE), pp. 1-11.
[23] van de Weijer, J., and Schmid, C., "Coloring local feature extraction," Proc. Computer Vision-ECCV 2006. 9th European Conference on Computer Vision. Proceedings, 7-13 May 2006, Springer, pp. 334-348.
[24] Quelhas, P., and Odobez, J. M., "Natural scene image modeling using color and texture visterms," Proc. Image and Video Retrieval. 5th International Conference, CIVR 2006. Proceedings, 13-15 July 2006, Springer-Verlag, pp. 411-421.
[25] Farag, A., and Abdel-H, A. E., 2004, "Detection, Categorization and Recognition of Road Signs for Autonomous navigation," Advanced Concepts in Intelligent Vision Systems, pp. 125-130.
[26] Schugerl, P., Sorschag, R., Bailer, W., and Thallinger, G., "Object re-detection using SIFT and MPEG-7 color descriptors," Proc. Multimedia Content Analysis and Mining. International Workshop, MCAM 2007, 30 June-1 July 2007, Springer-Verlag, pp. 305-314.
[27] Borenstein, J., 1995, "Control and kinematic design of multi-degree-of-freedom mobile robots with compliant linkage," IEEE Transactions on Robotics and Automation, 11(Compendex), pp. 21-35.
[28] Yongoug, C., Chongkug, P., and Harashima, F., 2001, "A position control differential drive wheeled mobile robot," IEEE Transactions on Industrial Electronics, 48(Copyright 2001, IEE), pp. 853-863.
[29] Rezaei, S., Guivant, J., and Nebot, E. M., "Car-like robot path following in large unstructured environments," Proc. 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, 27-31 Oct. 2003, IEEE, pp. 2468-2473.
[30] Yong, L., Xiaofei, W., Jim Zhu, J., and Jae, L., "Omni-directional mobile robot controller design by trajectory linearization," Proc. Proceedings of 2003 American Control Conference, 4-6 June 2003, IEEE, pp. 3423-3428.
[31] Fox, D., Burgard, W., and Thrun, S., "Controlling synchro-drive robots with the dynamic window approach to collision avoidance," Proc. Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96, 4-8 Nov. 1996, IEEE, pp. 1280-1287.
[32] Robert W. Erikson, M. D., 2001, Fundamentals of Power Electronics, Springer.
[33] "Pololulu Robotics and Electronics," http://www.pololu.com/catalog/product/777.
[34] Azad, P., 2009, Visual Perception for Manipulation and Imitation in Humanoid Robots, Springer.

[35] Peng, C., and Krumm, J., "Object recognition with color cooccurrence histograms," Proc. Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 23-25 June 1999, IEEE Comput. Soc, pp. 498-504.

[36] A. Nakhaei and F. Lamiraux, “Motion planning for humanoid robots in environments modeled by vision,” in Book Motion planning for humanoid robots in environments modeled by vision, *Series Motion planning for humanoid robots in environments modeled by vision*, Editor ed.^eds., City: IEEE, 2008, pp. 197-204.

[37] P. Vadakkepat, S. Ng Buck, D. Goswami, X. Zhang Rui, and Y. Tan Li, “Soccer playing humanoid robots: processing architecture, gait generation and vision system,” *Robotics and Autonomous Systems*, vol. 57, (no. Copyright 2009, The Institution of Engineering and Technology), pp. 776-85, 2009.

[38] S. Yu-Te, T.H.S. Li, H. Chia-Ling, L. Ming-Feng, H. Chun-Yang, and L. Shao-Hsien, “Omni-directional vision-based control strategy for humanoid soccer robot,” in Book Omni-directional vision-based control strategy for humanoid soccer robot, *Series Omni-directional vision-based control strategy for humanoid soccer robot*, Editor ed.^eds., City: IEEE, 2007, pp. 2950-5.

[39] D. Zhang and G. Lu, “A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures,” in Book A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures, *Series A Comparative Study on Shape Retrieval Using Fourier Descriptors with Different Shape Signatures*, Editor ed.^eds., City, 2001, pp. 1-9.

[40] S. Russell and P. Norvig, *Artificial Intelligence A modern Approach*: Prentice Hall, 2002.

[41] D. Clark and O. Michael, *Building Robot Drive Trains*: McGraw Hills, 2008.

[42] arduino, "http://www.arduino.cc"

[43] Z. Zhu, M.P. Naing, and A. Al-Mamun, “A 3-D simulator using ADAMS for design of an autonomous gyroscopically stabilized single wheel robot,” in Book A 3-D simulator using ADAMS for design of an autonomous gyroscopically stabilized single wheel robot, *Series A 3-D simulator using ADAMS for design of an autonomous gyroscopically stabilized single wheel robot*, Editor ed.^eds., City: Institute of Electrical and Electronics Engineers Inc., 2009, pp. 4455-4460.

[44] G.W. Lucas, “An Elementary Model for the Differential Steering System of Robot Actuators,” *Journal*, [online], (Date 2000).

[45] A. Robotics, http://www.acroname.com/robotics/parts/R48-IR12.html, 1994.

[46] MaxBotix, http://www.maxbotix.com.

[47] Xbee, http://www.digi.com/products/wireless/point-multipoint/xbee-series1-module.jsp.

[48] V. Manohar, P. Soundararajan, H. Raju, D. Goldgof, R. Kasturi, and J. Garofolo, “Performance evaluation of object detection and tracking in video,” in Book Performance evaluation of object detection and tracking in video, vol. 3852 LNCS, *Series Performance evaluation of object detection and tracking in video*, Editor ed.^eds., City: Springer Verlag, 2006, pp. 151-161.

[49] V.Y. Mariano, M. Junghye, P. Jin-Hyeong, R. Kasturi, D. Mihalcik, L. Huiping, D. Doermann, and T. Drayer, “Performance evaluation of object detection algorithms,” in Book Performance evaluation of object detection algorithms, vol. vol.3, *Series Performance evaluation of object detection algorithms*, Editor ed.^eds., City: IEEE Comput. Soc, 2002, pp. 965-9.

[50] J.C. Nascimento and J.S. Marques, “Performance evaluation of object detection algorithms for video surveillance,” *IEEE Transactions on Multimedia*, vol. 8, (no. Copyright 2006, The Institution of Engineering and Technology), pp. 761-74, 2006.

[51] K.J. O'Donovan, R. Kamnik, D.T. O'Keeffe, and G.M. Lyons, “An inertial and magnetic sensor based technique for joint angle measurement,” *Journal of Biomechanics*, vol. 40, (no. Copyright 2007, The Institution of Engineering and Technology), pp. 2604-11, 2007.

[52] L. Xue, C.-y. Jiang, H.-l. Chang, G.-m. Yuan, and W.-z. Yuan, “MIMU/magnetometer attitude estimation filtering algorithm based on state constraint,” *Journal of Chinese Inertial Technology*, vol. 17, (no. Copyright 2010, The Institution of Engineering and Technology), pp. 338-43, 2009.

Appendix A1

```
#include <Servo.h>
#include <Wire.h>
///////////////////////Gyro/////////
#define yawpin 0
#define NUMREADINGS 5
int Greadings[NUMREADINGS];
int Gindex = 0;
float Gtotal = 0;
float Gavg = 0;
const int lyawoffset=27;
const int ryawoffset=30;
long yaw=0;
int time, last_time;
float delta_t;
int last_yaw[3];
float yaw_deg;
int curcameraYaw=0,cameradir=0;
//////////////////////

//////////Sensors///////////
const char unit = 'c';
float USReading =0;
float US2Reading =0;
const int sonarpin=1;
const int sonarpin2=4;

const int lIR = 3;
const int rIR= 2;
int LIRValue = 0;
int RIRValue = 0;
const int numOfReadings = 5;
const int NumIR=100;
int LIRreadings[numOfReadings];
int RIRreadings[numOfReadings];
int USreadings[numOfReadings];
int US2readings[numOfReadings];
int LIRarrayIndex = 0;
int RIRarrayIndex = 0;
int USarrayIndex = 0;
int US2arrayIndex = 0;
int LIRtotal = 0;
int RIRtotal = 0;
int LIRavg = 0;
int RIRavg = 0;
int UStotal = 0;
int US2total = 0;
int USavg = 0;
int US2avg = 0;

int lastUStime=0;
int lastUS2time=0;
int lastIRtime=0;
/////////////////////
const int rStationaryPoint=85;
const int lStationaryPoint=86;
const double axle=38;
const int stepSize=2;

Servo cameraSteer;
Servo cameraTilt;
Servo lChannel;
Servo rChannel;

int IRleft=0,IRright=0;

 void setup() {
  // initialize serial communication:
  Serial.begin(9600);

     cameraSteer.attach(8);
     cameraTilt.attach(9);
     lChannel.attach(10);
     rChannel.attach(11);

     cameraSteer.write(90);
     cameraTilt.write(90);
     lChannel.write(lStationaryPoint);
     rChannel.write(rStationaryPoint);

    //Wire.begin(10);
    //Wire.onReceive(WireReceive);

   init_sensors();
 Serial.println("End of Setup...");
 //while(Serial.available()<=0);
 }

 void init_sensors()
{
  for(int i=0;i<3;i++)
       last_yaw[i]=0;
   //delay(6000);
   for (int i = 0; i < NUMREADINGS; i++)
   {
   Greadings[i] = analogRead(yawpin);
   Gtotal+=Greadings[i];
   }

 for (int thisReading = 0; thisReading < numOfReadings;
thisReading++)
 {
   LIRreadings[thisReading] = 0;
   RIRreadings[thisReading] = 0;
   US2readings[thisReading]=0;
 }
  resetUS();

  //I2C Sensors
  /*Serial.println("Before changing I2C address.");
  FrontSonar.connect();
  FrontSonar.changeAddress(CommandRegister,
New_Address);
  New_Address += 4; */

}
void resetUS()
{
  for (int i = 0; i < numOfReadings; i++)
  {
  USreadings[i] = 0;
  US2readings[i]=0;
  }
  UStotal=0;
  US2total=0;
  USarrayIndex=0;
  for(int i=0;i<numOfReadings;i++)
  {
    readMaxUS();
    //read_USreading();
    delay(20);
  }

}

 boolean sonarsent=false;

void readMaxUS()
{

  USReading =
((5000.0f/1024)*analogRead(sonarpin))/9.8;
  US2Reading =
((5000.0f/1024)*analogRead(sonarpin2))/9.8;
  if(unit=='c')
    USReading*=2.54;

  UStotal= UStotal - USreadings[USarrayIndex];
  US2total= US2total - US2readings[USarrayIndex];
  USreadings[USarrayIndex] = USReading;
  US2readings[USarrayIndex] = US2Reading;
  UStotal= UStotal + USreadings[USarrayIndex];
  US2total= US2total + US2readings[USarrayIndex];
  USarrayIndex = USarrayIndex + 1;
  if (USarrayIndex >= numOfReadings)  {
    USarrayIndex = 0;
  }
  USavg = UStotal / numOfReadings;
  US2avg = US2total / numOfReadings;
  lastUStime=time;
}


 void readIR()
{
  LIRValue = read_gp2d12_range(lIR);
  LIRtotal= LIRtotal - LIRreadings[LIRarrayIndex];
  LIRreadings[LIRarrayIndex] = LIRValue;
  LIRtotal= LIRtotal + LIRreadings[LIRarrayIndex];
  LIRarrayIndex = LIRarrayIndex + 1;
  if (LIRarrayIndex >= numOfReadings)  {
    LIRarrayIndex = 0;
  }
  LIRavg = LIRtotal / numOfReadings;

  RIRValue = read_gp2d12_range(rIR);
  RIRtotal= RIRtotal - RIRreadings[RIRarrayIndex];
  RIRreadings[RIRarrayIndex] = RIRValue;
  RIRtotal= RIRtotal + RIRreadings[RIRarrayIndex];
  RIRarrayIndex = RIRarrayIndex + 1;
  if (RIRarrayIndex >= numOfReadings)  {
    RIRarrayIndex = 0;
  }
  RIRavg = RIRtotal / numOfReadings;
  lastIRtime=time;
}

float read_gp2d12_range(byte pin) {
int tmp;

tmp = analogRead(pin);
if (tmp < 3)
return -1; // invalid value

return (6787.0 /((float)tmp - 3.0)) - 4.0;
}

void ManageObstacles()
{
  boolean isClear=true;
  int frontdist;
  if(unit=='i')
  frontdist=USavg*2.54;
  else
  frontdist=USavg;

  if(curcameraYaw<105 &&curcameraYaw>75 &&
US2avg<20)
   {
    lChannel.write(lStationaryPoint);
    rChannel.write(rStationaryPoint);
    if(curcameraYaw>90)
      Turn(45,'R');
    else
      Turn(45,'L');
   resetUS();
   return;
   }

  if(frontdist<25)
  {

   lChannel.write(lStationaryPoint);
   rChannel.write(rStationaryPoint);
   cameraSteer.write(100);
   //delay(150);
   for(int a=100;a>70;a-=3)
   {
   cameraSteer.write(a);
   for(int i=0;i<5;i++)
   {
   readIR();
   delay(15);
   }
   if(LIRavg<30)
   {
     isClear=false;
     break;
   }
   delay(15);
   }

   if(!isClear)    //if left side is also blocked turn right
   {
   cameraSteer.write(80);
   delay(150);
   for(int a=80;a<110;a+=3)
   {
   cameraSteer.write(a);
   for(int i=0;i<5;i++)
   {
   readIR();
   delay(15);
   }
   if(RIRavg<30)
   {
     isClear=false;
     break;
   }
   }

   if(isClear)
     Turn(15,'R');
   else
     {
          lChannel.write(lStationaryPoint-9);
          rChannel.write(rStationaryPoint+9);
          delay(500);
          Turn(15,'L');
     }
   }
   else
   Turn(15,'L');
   resetUS();
   //cameraSteer.write(90);
   //last_time=millis();
  }

}

 int angle=90,d1=0,d2=0,d3=0;

 int channelNum=0;
 int rSpeed = rStationaryPoint;
 int lSpeed = lStationaryPoint;

 void loop() {
       time=millis();

        if(time-last_time > 15)
       {
         readIR();

        Serial.print("US: ");
        Serial.print(USavg);
        Serial.print(",");
        Serial.print(US2avg);
        Serial.println(" (cms), ");

         Serial.print(" LIR:");
         printDouble(LIRavg,10);
         Serial.print(" RIR:");
         printDouble(RIRavg,10);
         Serial.println();

         readMaxUS();
        ManageObstacles();
        scanObject();
          last_time=millis();
          }

  anticipateCmds();

}

void scanObject()
{
cameraTilt.write(100);

  switch(cameradir)
  {
  case 0:
  if(curcameraYaw >= 140)
    cameradir=1;
  else
    cameraSteer.write(curcameraYaw);
  curcameraYaw++;
  break;
  case 1:
    if(curcameraYaw <= 40)
    cameradir=0;
  else
    cameraSteer.write(curcameraYaw);
  curcameraYaw--;
  break;
  }

  rSpeed = rStationaryPoint - 2;
  lSpeed = lStationaryPoint + 2;
  //rChannel.write(rSpeed);
  //lChannel.write(lSpeed);

}


void anticipateCmds()
{

  if (Serial.available() > 1)
  {
    int inByte = Serial.read();
     //Serial.println(inByte);

    switch (inByte) {
    case 'U':

      angle = 90-10*(Serial.read()-48);

      Serial.print("Camera Up Angle= ");
      Serial.println(angle,DEC);

      cameraTilt.write(angle);
      delay(15);
      break;
    case 'D':
      angle = 90+10*(Serial.read()-48);
      Serial.print("Camera Down Angle= ");
      Serial.println(angle,DEC);
      cameraTilt.write(angle);
      delay(15);
  break;
case 'L':
  angle = 90-10*(Serial.read()-48);
  Serial.print("Camera Left Angle= ");
  Serial.println(angle,DEC);
  cameraSteer.write(angle);
  delay(15);
  break;
case 'R':
    angle = 90+10*(Serial.read()-48);
    Serial.print("Camera Right Angle= ");
    Serial.println(angle,DEC);
    cameraSteer.write(angle);
    delay(15);
    break;
case 'W' :
case 'w':
  angle = Serial.read()- 48;
  rSpeed = rStationaryPoint - stepSize * angle;
  lSpeed = lStationaryPoint + stepSize * angle;
  Serial.print("Channel Speeds= (");
  Serial.print(rSpeed,DEC);
  Serial.print(",");
  Serial.print(lSpeed,DEC);
  Serial.println(")");
  rChannel.write(rSpeed);
  lChannel.write(lSpeed);
  delay(15);
  break;
case 'X' :
case 'x':
  angle = Serial.read()- 48;
  rSpeed = rStationaryPoint + stepSize * angle;
  lSpeed = lStationaryPoint - stepSize * angle;
  Serial.print("Channel Speeds= (");
  Serial.print(rSpeed,DEC);
  Serial.print(",");
  Serial.print(lSpeed,DEC);
  Serial.println(")");
  rChannel.write(rSpeed);
  lChannel.write(lSpeed);
  delay(15);
  break;

case 'A' :
case 'a':
  while(Serial.available()<3) delay(5);
  d1=Serial.read()-48;
  d2=Serial.read()-48;
  d3=Serial.read()-48;
  Turn(d1*100+d2*10+d3,'L');
  delay(15);
  break;
case 'S' :
case 's':
  while(Serial.available()<3)  delay(5);
  d1=Serial.read()-48;
  d2=Serial.read()-48;
  d3=Serial.read()-48;
  Turn(d1*100+d2*10+d3,'R');
  delay(15);
  break;
case 'F' :
```

```
    case 'f':
      Serial.print(USavg,DEC);
      delay(15);
      break;

    case 'Q':
    case 'q':
         channelNum = Serial.read()-48;
         Serial.print("Channel ");
         Serial.print(channelNum,DEC);
         Serial.println(" reset");
         switch(channelNum)
         {
           case 0:
             lChannel.write(lStationaryPoint);
           break;
           case 1:
             rChannel.write(rStationaryPoint);
           break;
           case 2:
             cameraSteer.write(90);
           break;
           case 3:
             cameraTilt.write(90);
           break;

         }
         break;
    }
     delay(150);
  }

}
void Turn(int angle,char dir)
{
  float totTime=0,t=0;
  const float motspeed=60.6;
  switch(dir)
  {
    case 'L':
    t= (angle*PI/180)/(PI/2)*1000;
    last_time=millis();
    while(time-last_time < t)
    {
      time=millis();
      lChannel.write(lStationaryPoint- 10);
      rChannel.write(rStationaryPoint - 10);
      //if(time-lastUStime > 30)
        //readMaxUS();
      //delay(15);
    }

    break;
    case 'R':
    t= (angle*PI/180)/(PI/2)*1000;
    last_time=millis();
    while(time-last_time < t)
    {
      time=millis();
      lChannel.write(lStationaryPoint+ 10);
      rChannel.write(rStationaryPoint + 10);
      //if(time-lastUStime > 30)
        //readMaxUS();
    //delay(15);
   }
   break;
  }
  lChannel.write(lStationaryPoint);
  rChannel.write(rStationaryPoint);
  Serial.print("Total Time=");
  Serial.println(time-last_time,DEC);
  //while(Serial.available()<=0);
}
void TurnGyro(int Angle, char dir)
{
   //int angle = 90+10*angleStep;
   //resetyaw();
   int totTime=0;
   float angle=Angle;
   int lspeed=0,rspeed=0;
   switch(dir)
   {
    case 'L':

    last_time=millis();
    while(yaw_deg + lyawoffset < angle)
    {
    time=millis();
    delta_t=(time-last_time);

    if(delta_t>=35)
    {
    yaw=getInput();     //yaw angle in deg/s
    calcAngle();
    /*
    lspeed=int((stepSize/2)*(angle/9));
    rspeed=int(angle/9);*/
    lspeed=rspeed=10;
    lChannel.write(lStationaryPoint- max(lspeed,10));
    rChannel.write(rStationaryPoint - max(rspeed,10));
    last_time=time;
    totTime+=delta_t;
    Serial.print(yaw,DEC);
    Serial.print("\t");
    Serial.println(yaw_deg,DEC);
    }

    }


    break;
    case 'R':
    last_time=millis();
    while(yaw_deg + ryawoffset < angle)
    {
    time=millis();
    delta_t=(time-last_time);

    if(delta_t>=35)
    {
    yaw=getInput();     //yaw angle in deg/s
    calcAngle();
    /*
    lspeed=int((stepSize/2)*(angle/9));
    rspeed=int(angle/9);*/
    lspeed=rspeed=10;
    lChannel.write(lStationaryPoint + max(lspeed,10));
    rChannel.write(rStationaryPoint + max(rspeed,10));
    last_time=time;
    totTime+=delta_t;
    Serial.print(yaw,DEC);
    Serial.print("\t");
    Serial.println(yaw_deg,DEC);
    }

    }

   break;
  }

  Serial.print("Total Time=");
  Serial.println(totTime,DEC);
  Serial.println("Angle Attained=");
  Serial.print(yaw_deg+lyawoffset,DEC);

  lChannel.write(lStationaryPoint);
  rChannel.write(rStationaryPoint);
  resetyaw();
  //while(Serial.available()<=0);
}
void resetyaw()
{
  last_yaw[2]=0;
   last_yaw[1]=0;
   last_yaw[0]=0;
   yaw = 0;
   yaw_deg = 0;
   Gtotal=0;
   Gavg=0;

   Gindex=0;
   delay(10000);
   for(int i=0;i<NUMREADINGS;i++)
   {
     Greadings[i]=analogRead(yawpin);
     Gtotal+=Greadings[i];
   }
}
void calcAngle()
{
  if(abs(yaw) > 5)
 {

 yaw_deg+=rk4Integrate(yaw, last_yaw[0], last_yaw[1],
last_yaw[2], delta_t);
 // yaw_deg+= trapIntegrate(rateold,rate,delta_t);
 //yaw_deg+=delta_t*yaw/1000;
 last_yaw[2]=last_yaw[1];
 last_yaw[1]=last_yaw[0];
 last_yaw[0]=yaw;

 }
}

void printDouble( double val, unsigned int precision){
   Serial.print (int(val));  //prints the int part
   Serial.print("."); // print the decimal point
   unsigned int frac;
   if(val >= 0)
```

```
                        frac = (val - int(val)) * precision;
   else
                        frac = (int(val)- val ) * precision;
   Serial.print(frac,DEC) ;
}
float rk4Integrate(int y4, int y3, int y2, int y1, float
deltax){
  float area=0;
  area=((y4+2*y3+2*y2+y1)/6)*deltax/1000;
  return area;
}

float getInput(){

   Gtotal -= Greadings[Gindex];
   Greadings[Gindex] = analogRead(yawpin);
   Gtotal += Greadings[Gindex];
   Gindex = Gindex + 1;

   if (Gindex >= NUMREADINGS)
     Gindex = 0;

    Gavg = Gtotal / NUMREADINGS;
   // Gavg=analogRead(yawpin);



  static float tmpf;       //temporary variable
  tmpf = Gavg * 5000.0 / 1023.0f;  //voltage (mV)
//  tmpf = Gavg * 4900.0 / 1023.0f;  //voltage (mV)
  tmpf -= 1230;  //voltage relative to zero level (mV)
  //tmpf -= yaw0*5000.0/1023.0f;  //voltage relative to
zero level (mV)
  tmpf /= 3.33;    //input sensitivity in  mV/deg/s(gyro)
  //tmpf /= 0.83;    //input sensitivity in  mV/deg/s(gyro)


  return tmpf;

}
```