Improved Ultrasonic Robot Navigation

Ramin Gharizadeh

Master's Thesis August 2017

I would like to dedicate this thesis to my dear parents and dear wife for tolerating all difficulties for raising me up mentally and physically.

Abstract

The purpose of this research is to employ the array processing methods to advance the steering of a robot within a room range by using ultrasonic sensors. Array processing has been utilizing to engender the effective two and three-dimensional images. These 2D and 3D images can be detected and processed by a robot to find the way in a dark, smoked, or, normal room.

Ultrasound imaging is used in the medical application; it is used in cases as imaging of inside organs etc. Array processing methods are also used to enable the robots for underwater operations as well. Marine application of ultrasound is vast as communication, detection and Hydrography. HUGIN Autonomous Underwater Vehicle(AUV) is a program held by Kongsberg Maritime jointly with FFI and UiO. Other applications of ultrasound can be named as shape recognition, positioning systems, transferring focused sound etc.

This research implements the array processing methods for ultrasound imaging. Sensor array signal processing is known as an active area of research and has the ability to combine different data which is collected at several sensors in order to accomplish a given estimation task (space-time processing).

Array processing helps to get a better resolution. Resolution can be defined as the quality of determination. Modifying the HCSR04 module and using two sensors as trig instead of one sensor, we could improve resolution from 15 degrees to almost 9 degrees.

Contents

1 Introduction **1**
1.1 Application of Sensors . . . 3
1.1.1 Comparison of Sensors . . . 4
1.1.2 Performance Variance in Different Conditions . . . 6
1.1.3 Market Insight . . . 7

2 Previous work **9**

3 Theoretical Background **10**
3.1 Speed of sound . . . 10
3.2 Ultrasonic Sensors . . . 10
3.3 Ultrasound (Ultrasonic Waves) . . . 11
3.4 Array Processing . . . 11
3.5 Wave Equation . . . 12
3.6 Aperture Function . . . 12
3.7 Aperture Smoothing Function . . . 12
3.8 Resolution . . . 16
3.9 Beamforming . . . 18
3.10 The Fresnel Approximation . . . 19

4 Research and Results **21**
4.1 Introduction . . . 21
4.2 Basic principle . . . 22
4.3 Testing the Sensor . . . 22
4.4 Testing Procedure . . . 24
4.5 Testing one Sensor . . . 25
4.6 Testing Linearity . . . 27
4.7 Finding Accuracy (Standard model of the Estimate) . . . 28
4.8 Resolution . . . 28
4.9 Simulations . . . 30
4.10 Improving Resolution with two sensors . . . 32

5 Experimental Result **35**
5.1 General Overview . . . 35
5.2 Power . . . 36
5.2.1 Voltage regulator . . . 37
5.3 Ultrasonic Sensor . . . 37
5.3.1 ultrasound . . . 37
5.3.2 Module pin definition . . . 37
5.3.3 Electrical parameters . . . 38
5.3.4 Module operating principle . . . 38
5.4 Stepper motor . . . 38
5.5 Micro-controller . . . 40
5.6 Ultrasonic Scanner . . . 42

5.7 2D Scan 44
5.7.1 Comparing HC-SR04 with two and three 49
5.8 Robot Navigation 50
5.9 SLAM 51
5.9.1 Landmark 51
5.9.2 Odometry 51
5.9.3 Feature Extraction 52
5.9.4 Line Extraction (RANSAC) 52
5.9.5 Algorithm 53
5.9.6 Random variable 54
5.10 Transfer function for Coordinate system 55
5.11 Navigating Robot through potential outway 57

6 Conclusion **59**

Appendices **60**

What we observe is not nature itself, but
nature exposed to our method of questioning.
Werner Heisenberg

Acknowledgement

Thank God creating this world with wisdom, freedom, and love, and I hope all human being can enjoy it.

I would like to express my special thanks of gratitude to Professor Sverre Holm and all my teachers from my first grade for all knowledge they taught me.

I also thank Professor Andreas Austeng, Kyrre Glette, Jim Tørresenand who helped me a lot with this research and also Informatikk Institute which made a great environment during my study.

I would also like to thank my parents and family who helped me while finalizing this research.

1 Introduction

The Ultrasonic sensor uses sonar to determine distance and direction to an object. It offers excellent non-contact range detection with high accuracy and stable readings[18]. Its operation is not affected by sunlight or black material. It comes with a complete package which contains ultrasonic transmitter and receiver module.

This research is conducted to invoke the ideas to achieve a better performance of ultrasound sensors. The concepts, principles, and applications of array processing are utilized in order to enhance the performance of ultrasonic sensors. Array structure can be defined as a set of sensors that are spatially separated, e.g. antennas [10] [33].

The main focus that is aimed to unravel by using array processing technique(s) is to[37]:

- Determine number and locations of energy-radiating sources (emitters).
- Enhance the signal to noise ratio SNR -"signal-to-interference-plus-noise ratio (SINR)".
- Track multiple moving sources.

The estimation gap of theoretical and practical statistics of various application have been the eminent interest for researchers. The parameter estimation has been an area of focus by statisticians and engineers because this area always lacks in performance and required improvements. Many techniques were the result of an attempt by researchers to go beyond the classical Rayleigh-limit[19].

Ultrasonic sensors are becoming very popular for robot navigation and range detection[15]. Due to their simplicity and low cost, they have overcome the competitors in educational and recreational applications. Other commonly used sensors are optical sensors, radar sensors, GPS sensors etc [20]. These sensors have vast commercial applications due to their accuracy (range and resolution). The commercial application of ultrasonic sensors have been restricted due to unacceptable qualities in range, and, especially resolution[36] [32]. Resolution is the quality of determination of an object. The higher the resolution is, the higher the accuracy of navigation. There have been numerous efforts to improve the resolution of ultrasound sensors to make them qualified for the applications with high precision. Resolution in such sensors can be improved by the quality of manufacturing and materials to avoid signal ambiguity[28]. But so far the main focus to improve the resolution has been to increase the size of sensors. These approaches cause a significant increase in the cost and complexity of sensors. In addition, the larger the sensor is, the more energy is consumed which is the prominent factor as robot applications are running on batteries. Another

approach is to use several small sensors and use them all to have a high quality resolution. The signal from different sensors should be assessed and analyzed to be applicable. Array processing can conduct the task by delay and sum the echo from several sensors[42]. By using this method, not only the resolution can be improved, but also the direction of echo source can be determined. Moreover, the range of detection increases[24]. Therefore this method can result in multiple benefits and increase in the applicability of the ultrasonic sensors. Many researchers have investigated this method in underwater detection and communication, medical applications etc using other types of sensors[17]. The main contribution of this research is to use array processing to optimize the resolution of the ultrasound sensors.

Objective of the research:

- To improve the accuracy of the object detection in a room.
- To use cost effective ways of improving robot's navigation.
- Widen the application of the robots; ultrasonic robots can be used to detect objects which light cannot. (light cannot detect glass, as light goes through it).
- Ultrasonic sensors improve the performance of the robot by using error eliminating rapid ultrasound firing (EERUF).

Scope of this research

This research intends to improve the performance of ultrasonic sensors. The future application could be:

- Localization and detection of robot or objects to improve the security of robot/humans environments.
- The robot can be programmed to pick the identified objects and place them at a proper place after cleaning the objects. This will help in cleaning and maintenance of the house as well.
- The robot can be further programmed to operate a washing machine, dish washers, etc.
- With advanced programming and use of advanced micro controllers, the robot can be used for cooking as well.

The motive behind selecting this research topic is to make life easier for everyone. Robots which can perform daily tasks like cooking, washing, cleaning, etc., can be manufactured at a low cost. The time saved from these kinds of engagements of humans can be utilized in other discoveries of human civilization.

Research method

The general goal of the project is to use HC-SR04 and modify the module to attain better resolution. It will be done by replacing one sensor as trig with two sensors.

This study uses a robot which will move around a room. Ultrasonic sensors will be used to send signals to all the corners of the room. The sensors will receive the echo, and, this echo will be processed by the micro-controller to detect the corners, and, other objects in the room. Simultaneous localization and mapping (SLAM) algorithm will be used to construct a map of the room; and, for simultaneous tracking of the robot. To find a better picture of the room Random Sample Consensus (RANSAC), an iterative method, would be deployed.

1.1 Application of Sensors

Increasing accidents in modern cities cost a lot for societies. A statistic from ASIRT (Association for safe international road travel) shows that there are nearly 1.3 million people who die in car accidents per year, on average 3287 deaths a day[3]. Additional 20-50 million become injured or disabled. The financial and immaterial cost of these numbers is noticeable for societies.

As the population and traffic of cities increase the cost of road accidents increase as well. Researchers are working to find new solutions to make modern and safer roads. According to (Dizikes, Peter) et al. research on a wireless highway, we can save many lives by making autonomous cars connected[8].

To decrease the accidents, self-driving cars is one of the encouraging solution catches a lot of attention and investment in last few years. Google and Tesla invest a lot in this field. These companies are already well known for introducing their first self-driving cars. Telenor in Norway is introducing his new 5G communication which will be coming in the market in 2018[35]. The 5G technology gives self-driving cars opportunity faster data transfer. Due to its 1Gbit/s speed, 5G technology provides self-driving cars and other connected systems (IoT) opportunity to exist. Sending and receiving a significant amount of data in a short time is necessary for IoT (Internet of Things), due to its extensive network and a lot of devices which want to communicate to each other. The short delay in data communication gives autonomous vehicles fast reaction time which improves failure rates and accident avoidance[4].

Ranging sensors are one of the important parts which make self-driving achievable. There are four main types of sensors which are used in range detection. Lidar, Radar, Ultrasonic and passive Visuals. Ultrasonic find range of an object by calculating time taken from sensor emit a sound pulse til echo is back

Figure 1: Autonomous cars detecting objects and humans in a wireless highway [12]

to the sensor. Lidar is similar to ultrasonic, except it uses light pulses instead of sound which make Lidar quite faster. Radar uses radio waves to determine range, angle or velocity of an object. Passive visuals use passive cameras and computer vision techniques and algorithms to determine object ranges in an image.

1.1.1 Comparison of Sensors

Google and Tesla are the top most developed companies in this field. Lidar systems are expensive compared to ultrasonics. Search on marked as Sparkfun, and other electronic providers show Lidar cost starts from 300$ where an ultrasonic sensor with module cost less than 1$(Ebay.com). The Lidar system Google uses for its cars cost $70,000 and 80KG weight. It should be placed on the top of the car, and due to current limitation, the system is not useful for detection of nearby objects. The range of these systems is 30-200 meters. Lidar works in all light condition but starts failing when snow, fog, rain and dust particle is increased. Biggest manufacturers of Lidar today are Continental AG, LeddarTech, Quanergy, and Velodyne.

The ultrasonic sensors have an edge over Lidar by introducing a very simple method to implement over a very low cost. They are very accurate for low ranges and can be used as a standalone navigation system for indoor robots or helping self-driving cars to sense small ranges and severe weather conditions. The ultrasonic sensor is used for detecting glass and other transparent objects because optical sensors are not always suitable for detecting them according to Ohtani K. research[23].

Figure 2: Lidar-attributes[5]

Figure 3: Ultrasonic-attributes [5]

Comparing figure 3 and 2 shows abilities of Lidar and Ultrasonic sensor strongly. We can conclude that using these 2 sensors as a sensing system can give us higher performance in object and range detection.

1.1.2 Performance Variance in Different Conditions

Figure 4 shows performance variance under different light and weather conditions. The Range is in meters. It is observed that Sonar is better than Lidar, under heavy rain, snow or fog condition.

Figure 4: sensor performance on different light and weather condition [5]

1.1.3 Market Insight

The global avoidance sensors market value in 2015 was USD 3.57 billion. An industry analysis from GVR (Grand View Research), Figure 5, shows market size for diverse types of sensors. Global demand of distance measurement sensor market has been increasing consistently. Analysts are of the view that the global demand of distance measurement sensor market will keep growing for the next six to seven years; with a CAGR of double digits[25].

Figure 5: Market size for different type of range sensors[25]

Distance Measurement Sensor: Market Dynamics

Distance measurement sensors can be applied in the automobile industry to increase the safety factor. Distance measurement sensors can be integrated with the automobile vehicles like cars, trucks, buses to provide the information of the surroundings to the driver. This information will help the driver to better understand the surrounding, and, execute accordingly. Thus, the safety of both the driver and the objects/human beings surrounding the vehicle can be increased. Apart from the automobile industry, distance measurement sensors find application in the construction industries (to trace the distance in various applications), and, military applications (for target detection; as in drone, and, air-crafts). However, high sensitivity towards environmental factors affects the accuracy of ultrasonic sensors, as the accuracy of ultrasonic sensors reduces in a noisy environment. Moreover, the accuracy of proximity sensor decreases in brighter environment.

Segmentation of Distance Measurement Sensors

Segmentation of distance measurement sensors can be done based on technology, operation type, applications, and, geography.

- Based on technology – the segmentation is done as infra-red led, capacitive sensor, inductive sensor, ultrasonic, laser diode, photo electric, draw wire, image sensor & others.

- Based on operational type – the segmentation is done as the time of flight, LASER triangulation, confocal chromatic imaging, and, photoelectric sensor.
- Based on photoelectric sensor – the segmentation is sub segmented into retroreflective, diffuse reflection sensing, and, through beam sensing.

Distance measurement sensors can also be segmented by application type; such as - manufacturing, robotics, defense aerospace and intelligence, automatic identification, packaging, consumer electronics, automobiles, and, others.

Distance measurement sensors have been segmented as geographically, as well. Seven geographically segmented regions are as follows - North America, Latin America, Western Europe, Eastern Europe, Japan, Asia Pacific Excluding Japan (APEJ), and the Middle East and Africa (MEA)[40].

2 Previous work

Ultrasonic systems, developed previously, function as a positioning system, only. These systems can be viewed as an indication systems with the ability to decide if a signal can be received or not. These systems can detect in which room the transmitter is located. A commercial application of such a system is the tracking of assets like a car, jewelry, etc [31]. This ultrasonic system is the reference system of this paper. Improving the room level accuracy, and, excellent positioning of the ultrasonic systems is the objective of this article. In this article, any reference to the robustness of positioning accounts for ranges of 5-10 meters, or, more.

These systems are in commercial use; however, they have significant shortcomings. The low update rate is a major challenge. The low bit rate of the ultrasonic communication channels results in a relatively low update rate of these systems. The chances of missing an object increase significantly, if many objects have to be located in a short span of time. Another important threat of ultrasonic systems is that portable ultrasonic transmitters may expose humans to near maximum recommended levels.

Many hybrid systems have been developed, such as Active Bat, Cricket, and Dolphin[1]. Cricket claims an accuracy in the 1-3 cm range; while Dolphin claims an accuracy of about 15 cm[22]. In real life, most of these systems are not useful because of their lower range of operation; even though the accuracy is high, even if the range is more than few meters these systems breakdown. Received signal level falls inversely with the distance; hence, for long ranges received signal to noise ratio is low for perfect detection beyond a certain range. Thus, most time-delay based systems breakdown in real life application.[27].

This research aims to improve range of ultrasonic sensors. We aim to achieve this by using error eliminating rapid ultrasonic firing. We will also improve the resolution. The emphasis in this paper is on parameter estimation methods in sensor array processing. A more systematic approach to estimating the number of signals is possible if the maximum likelihood estimator is employed. Most standard" approaches to array signal processing make no use of any available information about the signal structure. However, many human-made signals have a rich structure that can be used to improve the estimator performance. In digital communication applications, the transmitted signals are often cyclostationary, which implies that their auto correlation functions are periodic. This additional information is exploited.

3 Theoretical Background

3.1 Speed of sound

The speed of sound fluctuates and depends upon the medium in use (for instance, sound waves move faster in water than in air), plus with the characteristics of the medium, especially temperature. The word "speed of sound" is generally used to refer the speed of sound in air. The speed varies according to the atmospheric conditions; the most critical factor is the temperature.

According to Newton, the speed of sound in a substance is equal to square root of the pressure (STP) that is acting on it divided by the density of the medium [6]. If the speed of sound (c) in an ideal gas is given by the equation below:

$$c = \sqrt{(\gamma * P/\rho)} \tag{1}$$

where (P) is the ambient pressure and (ρ) is the gas density.

The physical properties are found to vary according to the change in ambient conditions. The speed of sound in the air will depend on the air temperature for example in 20C, the speed of sound in air is approximately 343 m/s. The speed of sound in dry air is approximately 331 m/s +- 0.05 m/s. The speed of sound varies when the temperature and humidity of the medium are changed[38]. The equation below defines how temperature affect speed of sound.

$$c = 331.1 + 0.606 * T \tag{2}$$

Where c is the speed of sound and T is temperature.

3.2 Ultrasonic Sensors

Ultrasonic sensors produce short and high-frequency sound pulses at regular intervals. These pulses transmit in the air at the velocity of sound. If they hit an object, they reflect back as echo signals to the sensor, which calculates the distance to the target among emitting the signal and receiving the echo based on the time-span.

The ultrasonic sensors are brilliant at suppressing background interface as the distance to an object is resolute by measuring the time of flight rather than the intensity of the sound. Practically all materials which reflect sound can be detected, in spite of their color. Even thin foils or transparent materials don't create any problem for an ultrasonic sensor. Ultrasonic sensors can see all the way through dust-laden air and ink mists. Even thin deposits on the sensor membrane do not weaken its function. Sensors with a blind zone of only 20mm and an exceptionally thin beam spread are making completely new applications possible today. Even lean wires are reliably detected.

3.3 Ultrasound (Ultrasonic Waves)

This study utilizes Ultrasonic waves or Ultrasound. Ultrasound is defined by the American National Standards Institute as "sound at frequencies greater than 20 kHz." In air at atmospheric pressure ultrasonic waves have wavelengths of 1.9 cm or less. Ultrasound is audio (sound) energy in the form of waves which consists of a frequency exceeding the human hearing range. The highest frequency detected by the human ear is around 20 thousand cycles per second (20,000 Hz). This is where the sonic range ends, and where the ultrasonic range begins.

3.4 Array Processing

There are four assumptions in array processing:

1. There is the uniform propagation of signals in all directions of the medium.
2. The radius of propagation is much greater than the size of the array, and, that there is plane wave propagation
3. Zero mean white noise and signal.
4. There is no coupling and the calibration is perfect.

Array structure can be defined as a set of sensors that are spatially separated. This study uses an array of sensors to first generate ultrasonic waves, and, then detect the echo to determine the objects in a room. The basic problem that we attend to solve by using array processing technique(s) is to:

1. Determine number and locations of the object(s).
2. Improve the signal to noise ratio (SNR).
3. Track multiple moving, or stable, objects.

In this study, an array of ultrasonic sensor module HC-RS04, which operates at 40kHz, is used. A sensor is a kind of transducer; which interacts with propagating energy, and, transforms it to an electrical signal. A sensor has a spatial extent and integrates energy from a spatial area and not a point. There are two crucial factors of a sensor - the linearity and the bandwidth. If a sensor is linear it means that the variation factor will not change when we transform energy from one type to another.

There are two types of sensors – directional sensors, and, omnidirectional sensors.

Directional sensors are focused on a direction; and, transform only signals that propagating from a certain direction, whereas omnidirectional sensors transform energy variations from all directions. Parabola is an example of a directional sensor.

In this study, directional sensors; with a limited angle of 15 degrees is used.

To calculate how the ultrasonic waves, interact with our sensors (HC-RS04) we need wave equation. The loss-less wave equation can be derived as below:

3.5 Wave Equation

The loss-less wave equation can be given as:

$$\boldsymbol{\nabla}^2 \vec{s} = \frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2} + \frac{\partial^2 s}{\partial z^2} \quad = \frac{1}{c^2}\frac{\partial^2 s}{\partial t^2} \tag{3}$$

Where ∇^2 stands for Laplacian operator s = s(x,y,z,t) is a general scalar field which in electromagnetic is : electric or magnetic field. and in acoustic it defines air pressure[10].

To find a function for a wave in Cartesian coordinate we can assume wave function has a complex exponential form as:

$$s(x, y, z, t) = A.exp^{j(wt - k_x - k_y - k_z)} \tag{4}$$

By substituting it inside wave equation we get:

$$s(\vec{x}, t) = A.exp^{j(wt - \vec{k}\vec{x})} \tag{5}$$

Plane wave means at any time t_0 the value of $s(x, y, z, t_0)$ are the same at all points lying in a plane given by $k_x x + k_y y + k_z z = C$, where C is a constant.

3.6 Aperture Function

The propagated waves will meet the sensor in a distinct area (called aperture), and, the sensor will integrate the energy of waves in this aperture area. The aperture function is given by:

$$w(x, y) = \begin{cases} 1, & \text{if } \sqrt{x^2 + y^2} \leq R \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Aperture function gives two valuable information about these sensors:

1. The spatial extent of aperture which explains shape, and, size of the sensor;
2. How the sensor integrates energy over its area.

A normal aperture will integrate signal equally over its extent, but it can give weights to each sensor. This will integrate over different areas of the sensor.

3.7 Aperture Smoothing Function

The sensor output when a field is observed through the finite aperture is given as:

$$z(\vec{x}, t) = w(x) * s(\vec{c}, t) \tag{7}$$

The aperture smoothing function, in terms of polar coordinates, can be derived as:

$$W(k_{xy}, \psi) = \int_0^{2\pi} \int_0^R exp\{jk_{xy}r(cos\psi cos\theta + sin\psi sin\theta)\}\, r\, dr\, d\theta \tag{8}$$

where Kxy represents radius, and, (pick it from pdf 35 page) represent an angle.

Because $cos(\theta - \psi) = cos\psi cos\theta + sin\psi sin\theta$, the integral over θ is independent of ψ and evaluate to $2\pi J_0(k_{xy}r)$, where $J_0(.)$ is a zero-order Bessel function of a first kind. From this simplification we obtain:

$$W(k_{xy}) = 2\pi \int_0^R rJ_0(k_{xy}r)\, dr \tag{9}$$

By using a Bessel function identity, findings of the aperture smoothing function of a circular array can be concisely be written as:

$$W(k_{xy}) = \frac{2\pi R}{k_{xy}} J_1(k_{xy}R) \tag{10}$$

Now plot the aperture smoothing function of sensors as a function of the angle which needs the dimension of the sensor. The figure 6 shows the dimension of the sensor.

Figure 6: HC-sr04 dimentions

R=10.3mm is the Radius of the sensor. The sensor is working with f = 40KHz and the speed of sound is c=340 m/s. From these, we can find λ which is the wave length of our sound waves

$$\lambda = \frac{c}{f} = \frac{340m/s}{40KHz} \quad = 0.0085m/rad \tag{11}$$

The aperture smoothing function of the sensor can be plotted as a function of angle

Figure 7: Element Responce

Element response shows us there are no side lobes on our plot. In this case, side lobes happen when aperture size is bigger than 10.3 mm. The side lobes also happen when frequency increase. To avoid side lobe frequency should not exceed 65 kHz. If we want to formulate it to avoid side lobes, w*R/c should be smaller than π.

Figure 8 shows how echo of sound waves will radiate from an object. Due to the fact that echo will travel different distances to reach each sensor. This issue is important for the calculation of the total response of the array of sensors.

As noticed output of sensors can be found relative to sensor 1. The difference of traveling distance of echo in relative to sensor 1 is calculable if the distances between each sensor to the next one is given. In this case, a uniform distance array will be used. $\triangle d$ is the distance between sensors and $\triangle d * (M - 1)$ is the distance between sensor one to sensor M. The difference between aperture smoothing function of sensor2 relative to sensor1 is a phase shift which corresponds to the extra distance wave travel relative to sensor1. Equation 12 shows how to calculate the total aperture smoothing function of a uniform distanced array of sensors.

$$W_{total}(k_{xy}) = W_e(k_{xy}) * W_a(k_{xy}) \tag{12}$$

Where $W_e(k_{xy})$ is aperture smoothing function of an element which is found from equation 10, now find $W_a(k_{xy})$ as follow:

Figure 8: Array of Sensors

$$W_a(k_{xy}) = \sum_{m=0}^{M-1} w_m \exp(jk_x) \tag{13}$$

The total aperture smoothing function will look like this:

$$W_{total}(k_{xy}) = \frac{2\pi R}{k_{xy}} J_1(k_{xy}R) * \sum_{m=0}^{M-1} \exp(-ik_{xy}md) \tag{14}$$

Where M is number of elements and d is distance between sensors. For 3 sensors where sensor 1 is at the center and sensor 2 and 3 are +-d on the right and left hand side of sensor 1. The Array response for this case will be:

$$W_{total}(k_{xy}) = 1 + 2 * \cos(kdsin\theta) \tag{15}$$

Figure 9 shows the Array response.

Discussion: To avoid side lobes we should have a bigger aperture. Bigger aperture better resolution. More cost.

Figure 9: Array Response* Element Response of 3 elements

3.8 Resolution

The resolution also means the minimum size of objects which we can distinguish between. In this study, range resolution will come under discussion later.

It is stated by Rayleigh criterion (for resolution) the minimum distance we can distinguish between two objects. Figure 10 shows interpretation of Rayleigh theorem [10] page 65

Aperture smoothing function of circular aperture fall on zero for the first time at radius

$$K_{xy} = \frac{1.22\pi}{R} \tag{16}$$

Resolution is depended on frequency and aperture size. Longer aperture gives better resolution and it means narrower main-lobe. But side lobes do not get lower. For circular apertures first side lobe which is highest occurs at $K_{xy} = 5.14/R$ which is equal $\frac{mainlobe}{sidelobe}$. It should be considered as a ratio which is equal 7.56

Figure 10: Rayleigh criterion[10]

Resolution is defined in some different ways or terms which are explained with the help of figure 11.

Figure 11: Azimuthal Resolution

Azimuth or lateral resolution of a system means the minimum distance r at which objects can be distinguished.

$$\Delta L = \theta * R = \frac{\lambda * R}{d} \tag{17}$$

Where d is aperture size, R is the length of the object to the aperture, λ is wavelength, and θ is the angle of two objects relative to the aperture center.

Aperture Ambiguities

Apertures are mostly symmetric. That means waves from some directions give same output from the aperture. This makes ambiguity which means we don't know which direction wave source is. For a circular aperture, we have

ambiguities for z axis which mean signals propagating from above and below can not be distinguished.

Characteristic	Linear Aperture	Circular Aperture
Aperture smoothing function	$\frac{\sin k_x D/2}{k_x/2}$	$\frac{2\pi R}{k_{xy}} J_1(k_{xy} R)$
Mainlobe width	$k_x = 4\pi/D$	$k_{xy} = 2.44\pi/R$
Resolution	$\delta\zeta_x = \lambda/D$	$\delta\zeta_{xy} = 0.61\lambda/R$
Relative sidelobe height	0.22	0.13
Ambiguity surface	Cone; axis parallel to array	Axis perpendicular to array plane

Figure 12: Aperture ambiguities [10]

Another important implication of array signal processing is a term which called grating lobes. While doing spatial sampling in array processing, some main lobes appear in aperture smoothing function. These lobes are called grating lobes which can not be distinguished from the main lobe and can cause inaccuracy and failures in our result. Grating lobes and main lobes accrue in same spatial frequencies. In figure 13 Signals that propagate from Kx=0 and Kx=2π/d can not be distinguished because they have the same intensity. Where Kx is wave-number and d is aperture size. λ is wavelength.

The data sheet of HC-SR04 provides 15 degrees of angular resolution. By studying the figure 7 notice that the sensor which has R=10.3mm lies good with this data around -6dB. It can be concluded that there is a threshold on -6dB. Data over -6dB will be counted as one (object in range) and data under -6dB as zero means no object detected in range. We will show how using array of sensors will give us better resolution.

The resolution, in this case, is the width of the main lobe at -6dB. It can be seen that width of main-lobe at -6dB on figure 7 compared to figure 9 is much bigger. This means using an array of sensors will give a better resolution. In theory, one sensor gives 15 degrees of resolution and two sensors gives 8 degrees of resolution but due to uncertainties in reality it gets around 16 degrees resolution with one sensor and around 9 degrees resolution with two sensors.

3.9 Beamforming

When waves propagate through a sensor, it contains information about source location which produces it. Temporal and spatial characteristics of the source of a signal with laws of physics allow determining the location of wave source. The

Figure 3.20 The aperture smoothing function magnitude $|W(k)|$ for uniform shading is plotted for a nine-sensor regular linear array. This spatial spectrum has period $k = 2\pi/d$. The visible region of the aperture smoothing function is that part for which $-2\pi/\lambda'' \leq k_x'' \leq 2\pi/\lambda''$. What might be called secondary mainlobes—those not located at the origin—are termed grating lobes.

Figure 13: Grating Lobes[10]

presence of unwanted wave sources or noise is always problematical. Beamforming will help to reduce noise or unwanted signals which propagate from other directions. It means we can focus to read signals from a particular direction. One easy task of beamforming is Array gain which reduces noise and amplifies our signal from a particular direction.

Delay and sum is one simple, strong array signal processing algorithm. It is also the heart of array signal processing. The algorithm can be expressed in term of array pattern for M equally spaced sensors separated by distance d:

$$W(k) = \frac{sin\frac{M}{2}k_x d}{sin\frac{1}{2}k_x d} \tag{18}$$

3.10 The Fresnel Approximation

Near-field and Far-field are two regions where some concepts should be respected when we calculate our result in each region.

Near Field can be defined as a region where the angular field distribution is dependent on the distance from the antenna. Here, Fersnels approximation can be used to determine the area of operation.

In the Far-Field region wavefronts from a wave source no longer retain their curvature and appear as plane waves. Figure 14 shows an illustration of near-field and far-field waves.

Figure 14: Near-Field and Far-Field region[16]

Fresnel approximation define distance from source which defines near-field and far-field region.[16]

Fresnel approximation states that the near-field far-field transition occurs at $D^2/4 * \lambda$ Where D stands for aperture size and λ for wavelength.

In our case where D is 6mm and λ is equal 0.0085 meters. The Fresnel range for near field is 0.001meter which is 1 cm. So for distances longer than 1cm, we can do our calculation in far-field concept which states that waves type are as plane waves and not curvature anymore.

4 Research and Results

Testing Ultrasonic Sensors

4.1 Introduction

Ultrasonic sensors transmit a pulse of sound which travels in a conical shape at the speed of sound (340 m/s). After being reflected by an object, the sound wave (i.e. echo) gets back to the range finder. The time interval between sending the pulse, and, receiving the echo is used to determine the distance of the object. In simple notation:

$$Distances = \frac{Elapsestime * speedofsound}{2} \tag{19}$$

Figure 15: HC-sr04 dimentions[39]

Features of Ultrasonic Sensor -HC-SR04 [39]

- Power Supply: +5V DC
- Quiescent Current: ¡2mA
- Working Current: 15mA
- Effectual Angle: ¡15
- Ranging Distance: 2cm – 400 cm/1ft – 13ft
- Resolution: 0.3 cm
- Measuring Angle: 30
- Trigger Input Pulse width: 10uS
- Dimension: 45mm x 20mm x 15mm

Pins of Ultrasonic Sensor - HC-SR04

- VCC: +5VDC
- Trig: Trigger (INPUT)
- Echo: Echo (OUTPUT)
- GND: GND

4.2 Basic principle

The system primarily consists of an ultrasonic sensor array unit, a signal processing unit, and an identification unit. In this study, the x coordinate position on the xz plane is defined as the width direction position of the measured object, and the z coordinate position on the xz plane is defined as the depth direction position of the measured object in the coordinate system.

The slope of the base of the object with respect to the xy plane is defined as the pose of the object. The ultrasonic sensor array unit has two-dimensional ultrasonic receivers and one transmitter. The transmitter positions the center of the receiver's array and irradiates an ultrasonic signal to the measured object.

The signal processing unit extracts the features of the measured object from the sensor array outputs using the signal processing circuit. The extracted data is inputted into the identification unit. The identification unit consists of two types of neural networks that perform shape identification and material identification.

4.3 Testing the Sensor

To test the sensors it needs to connect Raspberry Pi 3 microcontroller to the ultrasonic sensor(HC-SR04). Linux has been installed on the microcontroller Raspberry Pi 3. Python as defined in Linux, installed in the microcontroller, is used to test the sensors.

Figure 16: Raspberry Pi 3 with all input and output ports

The ultrasonic sensor(HC-SR04) is connected to the microcontroller as shown below:

Figure 17: Connecting HCSR04 to Rapsberry Pi

The ultrasonic sensor (HC-SR04) has 4 pins - VCC, GND, TRIG, ECHO. We should connect VCC pin, and, GND pin of the sensor module to the 5V pin, and, GND pin of Raspberry Pi, respectively. The Trig pin can directly be connected to Raspberry Pi's digital outputs (I chose pin no. 12). However, the Echo pin can not be connected directly. The Echo output is of 5V. The input pin of Raspberry Pi GPIO is rated at 3.3V. So, 5V cannot be directly given to the unprotected 3.3V input pin. Therefore, we use a voltage divider circuit using appropriate resistors to bring down the voltage to 3.3V.

Figure 18: 5V to 3.3V Voltage Divider

4.4 Testing Procedure

Figure 19: Connecting HCSR04 to Rapsberry Pi[39]

From Raspberry Pi, a 10us pulse is sent to the Trigger pin. At the same time, it starts a pulse to calculate time which can be named as a Time Pulse. After a pulse goes through trig, the module will create eight 40KHz pulses and transmit it through the TX sensor. The RX sensor will listen to the echo. When the RX sensor detects a 40KHz signal, it will send a pulse to the Echo Pin. When Raspberry Pi detects pulse from the Echo Pin, it ends the Time Pulse.

The pulse duration of the Time Pulse can be found by calculating the difference of End time – Start time of the Time Pulse.

The time is taken by the sound wave to travel to object's surface, and the time taken by the echo to reach the sensor can be calculated.

With this information, it will be easy to find the distance of the object from the module.

Distance Calculation:

Time taken by the pulse is for the to and fro travel of the ultrasonic signal. However, we need only half of this. Therefore, time is taken as time/2.

Hence,

$$Distance = \frac{Time * Speed}{2} \tag{20}$$

Speed of sound at sea level = 343 m/s or 34300 cm/s

Thus, Distance = 17150 * Time (cm)

Ultrasonic sensors have an acoustic transducer which is vibrating at ultrasonic frequencies. The pulses are emitted in a cone-shaped beam and aimed at a target object. Pulses reflected from the target to the sensor are detected as echoes. The device measures the time delay between each emitted and echo

pulse to determine the sensor-to-target distance.

Ultrasonic Position Sensors can detect target objects made of any material. Transparent, or, opaque objects of any color can be detected. For example, a glass bottle, as well as a rubber tire, can be accurately detected by ultrasonic sensors. Moreover, ultrasonic sensors can recognize wrapping material - shiny, as well as, Matt. Ultrasonic sensors perform well in, almost, any environment - dusty, noisy, etc.

This research paper uses the ultrasonic sensor HC-SR04. HC-SR04 is a low-priced stable ultrasonic sensor with high accuracy. Sunlight, and, other noises do not affect the working of ultrasonic sensor HC-SR04. With an accuracy of 3 mm, ultrasonic sensor HC-SR04 functions properly between 2 cm – 400 cm. Module – HC-SR04 has a control unit, transmitter, and, receiver[39].

4.5 Testing one Sensor

One sensor is connected to the module to test it. So, one sensor is connected to the trig(TX), and, one sensor to the echo(RX).

Figure 20: Testing one Sensor

An object is placed 10cm apart to test this sensor.

The result of the python program shows a reasonable result. It's worth to note that the result shows distance of objects surface to the outer part of the

Figure 21: Detecting Object with one Sensor

sensor!

Taking average of 10 results we can find average=10 Result/10

Figure 22: Output of Python Program

Average= 10.2cm, which is reasonable (inaccuracy = ± 2 %)

4.6 Testing Linearity

Linearity error is the deviation of the sensor output curve from a specified straight line over the desired pressure range. The linearity error value is normally specified as a percentage of the specified pressure range.

Figure 23: Testing Linearity

For testing linearity object is placed at different distances, and the result of the python program is recorded. MATLAB is used to fit line over data and check how accurate the result is.

The result of the MATLAB program (see appendices) is sown below:

Figure 24: Best Fit Line

The red dots show result from the sensor, and, the blue line shows the best fit line. The results show that the sensors produce linear output.

The accuracy of our sensor can be determined by Standard Error of the Estimate procedure.

4.7 Finding Accuracy (Standard model of the Estimate)

The accuracy of predictions can be estimated utilizing the Standard Error of Estimate procedure. Thus, the accuracy of the sensor can be determined using Standard Error of Estimate procedure. Sum of squared deviations of predictions can be minimized using the regression line. Standard Error of Estimate, and, the regression line are closely related; and it can be defined as below:

$$\sigma_{est} = \sqrt{\frac{\sum(Y - Y')^2}{N}} \tag{21}$$

where σest is the standard error of the estimate, Y is an actual score, Y' is a predicted score, and N is the number of pairs of scores. The numerator is the sum of squared differences between the actual scores and the predicted scores.

We use a MATLAB code (see appendices) to find this accuracy.

$\sigma_e st$ = 0.02 cm (error ± 2%).

The linearity of the sensor can be estimated as 98% accurate.

4.8 Resolution

Figure 25: Resolution of 1 sensor

As shown in the figure above, two objects are used to find the resolution of the sensors. These two objects are used to calculate the angular resolution of the sensors.

First, an object A is placed directly in front of the sensors. A new object B is placed in between the sensors, and, the object A. The object B is moved parallel to the object A until the object B is detected by the sensors. After this, the object B is moved back to find the border in which the object A is detected. The same procedure is repeated with object C from the other side. Thus, the minimum distance between object B and object C can be found in which the object A can be detected.

The resolution of this sensor is calculated at three different positions - 5cm, 10cm, and, 15cm. Calculation shows that the sensor has a linear resolution.

Figure 26: Resolution of 1 sensor

Angle =15.1 °

Figure 27: Best Fit Line - Resolution of the Sensor

We use MATLAB code to calculate the angle

matlabcode:

```
a=[6, 11, 16];
b=[5.5, 7, 8.2];

p=polyfit(a,b,1);

x=linspace(4,20,1000);
y=polyval(p,x);

scatter(a,b,'ro')
plot(x,y,'b')
axis([5 20 5 10]);

angle=(atan(p(1))*180)/pi
```

4.9 Simulations

Sensing multiple objects using one sensor can be challenging. Echo from different objects will interfere each other, and, it would be extremely difficult for the sensor to uniquely identify an object. To illustrate this challenge let's consider the example below: The figure below shows an object in front of a speaker (which acts as a transmitter). The red circle is the detector which will detect the object A. For simulation, the java program named wave interference simulator has been used. [30].

(a) Transmiting one pulse (b) Echo of one pulse

Figure 28: [30]

Figure 28(a) and 28(b) show the transmitted signal, and, echo received.

One object can be detected easily but the problem arises when there are multiple objects are present for detection.

Figure 29(a) shows two objects placed in front of the speaker(which acts as a transmitter) - object A, and object B. It can be seen in Figure 29(a), the object B is placed at an angle to the speaker, and, it is closer to the speaker as compared to object A.

Figure 29(b) shows that the echo the from object B has reached the detector while the echo from object A has not reached yet. This means that the distance

(a) Transmiting one pulse (b) Echo from object B (c) Echo from object A

Figure 29: Echo from two objects

of the object from the sensor, as well as, the angle of the object determines the time it takes for the echo to reach the sensor.

Figure 29(c) shows that the echo from object B has reached the sensor while the transmitted signal has just reached the object A

Figure 30(a), and Figure 30(b) show object A, and object B at different distances from the sensor, but at different angle.

(a) Trasmiting one pulse (b) Constructive echo

Figure 30: Angle of object A and B make constructive waves reach detector.

In this case, the echo is detected at the same time. Both the objects send the echo at the same time to the sensor. The sensor will not be able to differentiate between object A, and object B.

Hence, angle, shape, and distance of the objects will have an impact on the measurement and, identification of objects by one sensor.

This issue can be addressed by array processing.

4.10 Improving Resolution with two sensors

The goal is to be able to detect objects uniquely. To achieve the goal, it is mandatory to understand the behavior of waves, and, their interference.

Figure 31 shows how waves interact with each other; when the waves are transmitted from two different transmitters (or sensors).

Figure 31: Constructive interference and Destructive interference

Constructive interference, and, destructive interference patterns on the wall of a room is shown in figure 32 below:

Figure 32: Interference Patterns on the wall of a room

Figure 32 shows white, and, black areas on the wall. Constructive interference shows white spots, and, destructive interference shows black spots. Moreover, there are linear areas where variations cannot be detected. Thus, the sensors would not receive echo from the wall.

Figure 33(a) shows waves produced by two transmitters (or sensors). The detector is in between the two transmitters.

(a) Producing waves with two sources

(b) Echo of interfered waves

(c) Detector A and B getting different variations

Figure 33: Interfering waves and echo

Figure 33(b) shows echo of waves from the wall.

Figure 33(c) shows positions of detection of echo - at A, and, at B. At A the echo of constructive waves would be detected, whereas, at B no echo would be detected.

Now utilize this behavior of the interfering waves to make the main lobe narrower; and, thus get better resolution. However, grating lobes must be managed as well.

Figure 34(a) shows two transmitters (or sensors) producing waves to detect a wall using a single slit. The position of the detector is in between the two transmitters (or sensors).

(a) Producing waves with two sensors (b) Echo from S1 and S2 (c) Echo from the wall (W)

Figure 34: Detecting the wall of a room using single slit

Figure 34(b), and, Figure 34(c) show how the destructive, and, the constructive behavior of waves will help us in not detecting the echo waves from other directions.

Testing HCSR04 Sensor

As shown in the figure 35, two sensors are used as transmitters, and, the sensor in between these two sensors is used as a detector.

Figure 35: Improvement of resolution

The method described in section 5.6 is used to find the resolution of the sensors. The Matlab code gives a resolution of 9 degrees using two sensors as transmitters; which is better than the resolution of a single sensor which is 15 degrees.

Figure 36: Finding resolution of two sensors with matlab

5 Experimental Result

In this section, an actual robot is configured to demonstrate the results of this study.

5.1 General Overview

Figure 37: General Overview

5.2 Power

Circuits and sensors usually require a dc power supply that can maintain a fixed voltage while supplying enough current to drive a load. To design a power supply, some factors should come under consideration to avoid any harm to the parts of the circuit. The datasheet of 12V Lipo battery which is going to use shows max3amp output. The raspberry-pi need 5-volt dc to work. Here, voltage regulator should be used to deliver right voltage to microcontroller.

5.2.1 Voltage regulator

Voltage regulators are 2 types. Linear regulators and a Switching regulator. An integrated component named L7805CV is used to design a linear regulator. It will regulate 12V to 5V dc. Figure 38 design of the regulator.

Figure 38: Voltage Regulator

C1 and C2 are coupling capacitors which will avoid distortion and hold voltage stability on 5V dc. The other usage of coupling capacitors is needed when AC current will be used as an input and want regulated DC current as an output. Then C1 and C2 will work as a filter. The advantage of LM78## components is that they are very chip and very easy to use. The disadvantage is that if the input voltage is much higher than output voltage the difference voltage will produce heat which is a waste of energy.To find how much heat will be produced as wasted energy can be given as:

$$P_{7805} = (V_{in} - V_{out}) * I_{out} \tag{22}$$

for example if our circute use 1 amper the P = (12-5)*1 = 7Watts.

5.3 Ultrasonic Sensor

5.3.1 ultrasound

In this study, the ultrasonic sensor module HC-RS04 is going to be used which operates at 40kHz. A figure of the sensor is shown in figure 39.

5.3.2 Module pin definition

Pin Symbol	Pin Function Description
VCC	5V power supply
Trig	Trigger pin
Echo	Echo pin
GND	Power ground

Figure 39: HC-SR04 Ultrasonic sensor module

5.3.3 Electrical parameters

Electrical Parameters	HC-SR04 Ultrasonic Module
Operating Voltage	DC-5V
Operating Current	15mA
Operating Frequency	40kHz
Farthest Range	4m
Nearest Range	2cm
Measuring Angle	15 Degree
Input Trigger Signal	10us TTL pulse
Output Echo Signal	Output TTL level signal, proportional with range
Dimensions	45*20*15mm

5.3.4 Module operating principle

Set low the Trig and Echo port when the module initializes, firstly, transmit at least 10us high level pulse to the Trig pin (module automatically sends eight 40K square wave), and then wait to capture the rising edge output by echo port, at the same time, open the timer to start timing. Next, once again capture the falling edge output by echo port, at the same time, read the time of the counter, which is the ultrasonic running time in the air. According to the formulae: test distance = (high level time * ultrasonic spreading velocity in air)/2, you can calculate the distance to the obstacle.

5.4 Stepper motor

For mapping the room the process should be repeated to rotate and stop the array of sensors in a way to get a 2D information about the range of objects in two dimensions. The stepper motor is used for this purpose. Stepper motors provide a high amount of torque at low speed, making them suitable for applications where low speed and high precision position control is needed. Stepper motors are DC motors which move in discrete steps. By using multiple coils which are organized in groups called phases. By sending current through each phase in sequence, the motor will rotate. Accurate positioning and speed control can be achieved with a computer. In this case, we use a raspberry pi and USBtoDynamixel to control the stepper motor[41].

Figure 40: Stepper motor

The step motor used is Dynamixel AX-12A. The AX-12A robot servo has the ability to track its speed, temperature, shaft position, voltage, and load. The control algorithm used to maintain shaft position on the ax-12 actuator can be adjusted individually for each servo, allowing us to control the speed and strength of the motor's response. All of the sensor management and position control is handled by the servo's built-in microcontroller. This distributed approach leaves us main controller free to perform other functions. The Dynamixel actuator is operated by writing values to its control table and its status is checked by reading values from its control table. The data values for the RAM area will be set to the default initial values whenever the power is turned on. However, the data values for the EEPROM area are non-volatile and will still remain even after the power is turned off.

Figure 41 shows control table of out stepmotor.

3-4. Control Table

	Address	Item	Access	Initial Value
EEPROM Area	0(0X00)	Model Number(L)	RD	12(0x0C)
	1(0X01)	Model Number(H)	RD	0(0x00)
	2(0X02)	Version of Firmware	RD	?
	3(0X03)	ID	RD,WR	1(0x01)
	4(0X04)	Baud Rate	RD,WR	1(0x01)
	5(0X05)	Return Delay Time	RD,WR	250(0xFA)
	6(0X06)	CW Angle Limit(L)	RD,WR	0(0x00)
	7(0X07)	CW Angle Limit(H)	RD,WR	0(0x00)
	8(0X08)	CCW Angle Limit(L)	RD,WR	255(0xFF)
	9(0X09)	CCW Angle Limit(H)	RD,WR	3(0x03)
	10(0x0A)	(Reserved)	-	0(0x00)
	11(0X0B)	the Highest Limit Temperature	RD,WR	85(0x55)
	12(0X0C)	the Lowest Limit Voltage	RD,WR	60(0X3C)
	13(0X0D)	the Highest Limit Voltage	RD,WR	190(0xBE)
	14(0X0E)	Max Torque(L)	RD,WR	255(0XFF)
	15(0X0F)	Max Torque(H)	RD,WR	3(0x03)
	16(0X10)	Status Return Level	RD,WR	2(0x02)
	17(0X11)	Alarm LED	RD,WR	4(0x04)
	18(0X12)	Alarm Shutdown	RD,WR	4(0x04)
	19(0X13)	(Reserved)	RD,WR	0(0x00)
	20(0X14)	Down Calibration(L)	RD	?
	21(0X15)	Down Calibration(H)	RD	?
	22(0X16)	Up Calibration(L)	RD	?
	23(0X17)	Up Calibration(H)	RD	?
RAM Area	24(0X18)	Torque Enable	RD,WR	0(0x00)
	25(0X19)	LED	RD,WR	0(0x00)
	26(0X1A)	CW Compliance Margin	RD,WR	0(0x00)
	27(0X1B)	CCW Compliance Margin	RD,WR	0(0x00)
	28(0X1C)	CW Compliance Slope	RD,WR	32(0x20)
	29(0X1D)	CCW Compliance Slope	RD,WR	32(0x20)
	30(0X1E)	Goal Position(L)	RD,WR	[Addr36]value
	31(0X1F)	Goal Position(H)	RD,WR	[Addr37]value
	32(0X20)	Moving Speed(L)	RD,WR	0
	33(0X21)	Moving Speed(H)	RD,WR	0
	34(0X22)	Torque Limit(L)	RD,WR	[Addr14] value
	35(0X23)	Torque Limit(H)	RD,WR	[Addr15] value
	36(0X24)	Present Position(L)	RD	?
	37(0X25)	Present Position(H)	RD	?
	38(0X26)	Present Speed(L)	RD	?
	39(0X27)	Present Speed(H)	RD	?
	40(0X28)	Present Load(L)	RD	?
	41(0X29)	Present Load(H)	RD	?
	42(0X2A)	Present Voltage	RD	?
	43(0X2B)	Present Temperature	RD	?
	44(0X2C)	Registered Instruction	RD,WR	0(0x00)
	45(0X2D)	(Reserved)	-	0(0x00)
	46[0x2E)	Moving	RD	0(0x00)
	47[0x2F)	Lock	RD,WR	0(0x00)
	48[0x30)	Punch(L)	RD,WR	32(0x20)
	49[0x31)	Punch(H)	RD,WR	0(0x00)

Figure 41: Dynamixel AX12 Control Table[11]

5.5 Micro-controller

A Micro-controller is a small computer without keyboard, mouse, and monitor on a single integrated circuit. Micro-controller houses onboard central processing unit (CPU) that performs the function as a computer (Microprocessor) performs a logical operation, Input/Output control operations, etc. It also houses extra on-chip goodies as ROM, RAM, serial communication port, often A/D converter and more. Microcontrollers can be used to build robots, TV, VCR,

Generate sound via a speaker, toys, infrared sensors, control systems, security systems and any device that requires program-like control. In this case, the micro controller going to take control of

- Stepper motor rotation
- Ultrasound sensor chip -Trig (start HC-SR04 to function)
- Echo (Read the data and check range of object detected)
- Take control of robot movements (Control motor driver to start DC motors)

The microcontroller which is going to use is Raspberry Pi model 3.

Figure 42: Raspberry pi 3 [21]

Raspberry pi 3 feature SoC on-chip, CPU with speed 1.2 GHz, Graphic Process Unit (GPU), on board 1GB memory RAM, SD card, Bluetooth and WiFi, 4 USB port, HDMI port and I/O.

Figure 111 shows Input/Output pins. Analog/Digital input outputs and a Serial communication port.

Figure 43: Raspberry pi 3 I/O [29]

5.6 Ultrasonic Scanner

In this section, the rotating ultrasonic scanner will be implemented. A tested sensor is now to place above the robot by using stepper motor Dynamixel AX12a. Figure 40 and 41 shows image and control table of this stepper motor.

The Dynamixel AX12 is a smart actuator with integrated part such as : DC motor, reduction gear head, controller, driver, network in one DC servo module. AX12 is with advanced functions like precise control, PID control, 360 degrees of position control. It also has high-speed communication, advanced durability, degree of precision and wider control zone. It is compatible with raspberry pi and python. Using USBtoDynamixel adapter to connect raspberry pi to Dynamixel actuator. Figure 48 shows an overview for connecting different parts.

3 (0X03)	ID	ID of Dynamixel	RW	1 (0X01)
43 (0X2B)	Present Temperature	Current Temperature	R	-

Figure 44: Dynamixel AX-12A Register Address 3 and 43 corresponds to ID of Dynamixel and Current Temperature[11]

Now use Python code to rotating the actuator by rotating sensors 10 degrees on each step.

Run the simulation with some different degree values to check functionality. Since the resolution of sensors is around 10-15 degrees, rotate 15 degrees for each sampling, but it's more accurate to take 10 degrees for each step due to uncertainties and environment complexity.
It is decided to rotate the actuator only 150 degrees to scan the front view of the robot, but of course, this method can be used to get 360 degrees scan of the room. For scanning side and back of robot, the robot self should be rotated.

For each step, it can read the state of the actuator and get data from the sensor. By using these two data, a 2D visualization of the room can be plotted.

From e-manual (figure 41) all register-address information and firmware version can be found. Using Dynamixel library, a Python class object can be setup for controlling the actuator.
Dynamixel Dxl(1); Following code will define Dxl object. Dynamixel control registers can be accessed via the class object. Dxl.readByte and Dxl.writeByte gives full control over Dynamixel AX12A. It can read all information as temperature, position if moving etc with Dxl.readByte. Functions are easy to use. Dxl.readByte(ID,Address) gets two parameters as input. ID number and Address of register to be read. Dxl.writeByte(ID,Address,Param[]) function get three parameters as input. The ID register identifies which Dynamixel is to be controlled. Finally, the value of params will be written to given Address.

The actuator model is 12 and version is 24. The temperature can be read from address 0x43. The ID of the Dynamixel actuator is available on address 0x03. The first ID of Dynamixel should be set as for example one.

The state of address 46(0x2E) defines if the actuator is under movement. If the state is 1 it means motor is moving and if 0 it means it is not moving.

Then we try to rotate the AX-12A Dynamixel to a location of 100 degrees. In normal mode, AX-12A has a rotation range of 300 degrees. Register address for Goal Position should be used. The goal position is composed of 2 bytes. One lower byte and one higher byte. Address 30(0x1E) and 31(0x1F) corresponds to Goal Position(L) and Goal Position(H). The figure below shows an interpretation of AX12A rotation range. Goal Position can be set to a number range from 0 to 1000. The number zero corresponds to degree 0 and number 1000 corresponds to degree 300. As an example for rotating AX-12A to degree 150, it should give value 150 to address of Goal Position(31 and 32). 150 in binary form is equal [0x200].

Figure 45: Dynamixel AX-12A rotation range

The rotation can also be restricted by using Address 12 and 13. It can be restricted to 150 degrees which are quite enough to see the front of our robot. Another address register which is usable is Max-torque on address 14 and 15.

5.7 2D Scan

When humans navigate to a destination, they use vision as sensing system to get an illustration of the environment around, to find a possible way to the destination. A robot also needs a sensing system to collect information about the environment around it. In this part, the goal is to use an actuator with build sensors on it, to make a 2D room scanner. It will help the robot to get feelings about possible out ways around it self. Figure 46 shows how to fix the sensors above Dynamixel actuator. A python code which has three parts is written and compiled in Raspberry-pi. It will take care of rotation of actuator, reading sensors data and navigate the robot.

Figure 46: 3 Sensors fixed on Dynamixel AX-12A

Python Code for Rotating actuator: A python library which is provided by GitHub, is used to control Dynamixel actuator. The library has two functions,

one for reading AX12 registers and one for writing to a particular register of AX12a. Dynamixel AX12a can rotate 300 degrees in normal mode as discussed. There are some variables defined which adopt rotation of AX12a actuator as desired.

- MaxAngle: Maximum angle which actuator is allowed to rotate
- MinAngle: Minimum angle which actuator is allowed to rotate
- Torque: Define torque of our actuator
- Speed: Speed of rotation
- Step: A number which defines how many degrees actuator rotate for takings a sample

Figure 47 illustrates how AX12a actuator operates.
The rotation of actuator is limited by 300 degrees in normal mode, which is shown as a blue area in figure 47. The two variables (Maximum and Minimum angle) can be used to set a limit to rotation of Dynamixel AX12a. Step variable defines how many degrees the stepper motor rotates on each sampling step.

Figure 47: Control variables for Dynamixel AX-12A

Another part of python code will take care of sensors sampling as it rotates. Due to a complex program which has many parts, the processing time also becomes longer; The time delay which is provided in python code should be set to propitiate time to make scanner run smoothly. The code is included in appendices.

Connecting stepper motors to raspberry pi happen via a motor driver chip usually. But I have used a DynamixelToUSB adapter and an SMPS2Dynamixel Adapter to connect Dynamixel to raspberry pi 3. DynamixelToUSB can control a network of Dynamixel actuators from PCs USB port.

Figure 48: Connecting Dynamixel AX-12A to Raspberry pi via Dynamixel-toUSB adapter

In next step, a python library named Turtle is used to visualize 2D scan of the room. A python code is developed to draw a line corresponding to each sample. The length of the line is determined by the value of distance which is measured by sensors. The Turtle library is elementary and efficient to use. The object 'dw' is defined from Turtle class. It gives us access to several functions of the class. For example, dw.home() will return drawer to the center of the plot. dw.hset(angle) will define in which angle should the line be drawn from the current position. Finally, dw.forward(distance) will draw a line from current position with the specified angle. The variable distance' determines the length of the line. An illustration of the drawer is shown in figure 49

Before setting up the scanner some experiments are necessary. The first experiment is to find out if our system will give us a better resolution. As the figure 50 shows, there is set objects around sensors with an out way in front [9].

Figure 49: An illustration of Turtle drawer. (A Python library)

Figure 50: Room with two potential out ways

The Dynamixel actuator will begin to rotate and scan 225 degrees around it. The speed can be set to an average value. Rotation speed and minimum and maximum angle of the rotation can be adjusted. We also adjust the torque at its maximum. Tryomh some delays with time.sleep() which is available on Python will help us get a smooth rotation of the actuator. The speed should not be so fast which affect the result.

Figure 51 shows the result of the Python code. Noticing that two out ways are detected shown as long lines.

The size of outway can be calculated. As the room is getting are scanned, distance to each angle can be stored as a variable named current-distance. The previous result should also be saved in a different variable as previous-distance. The difference between current-distance and previous distance could be stored in a variable diff. If the difference is higher than for example 10cm It can indicate

Figure 51: 2d Scanner

a potential out way. The angle of current distance will be stored in a variable named angle-1. Angle-2 is also found by repeating last procedure. By using the difference of angle-1 and angle-2, and corresponding distance size of out-way can be found. Figure 5 shows an illustration of the procedure for determining the size of outway.

Figure 52: Finding length of out way

Equation 5.7 shows the calculation of side length of a triangle when the length of two sides and size one angle are known.

$$c = \sqrt{a^2 + b^2 - 2 * a * b * cos(\theta)} \tag{23}$$

Where :

- c is length of outway.
- a is dist1 (Distance of corresponding angle1)
- b is dist2 (Distance of corresponding angle2)
- θ is difference between Angle1 (ang1) one and Angle2 (ang2)

5.7.1 Comparing HC-SR04 with two and three

In this part HC-SR04 original and modified are compared. Notice that HC-SR04 has one sensor on the trig pin, and modified HC-SR04 has two sensors on trig and one sensor on echo pin. An illustration of HC-SR04 original and modified is shown in figure 53. Wiring procedure is also shown in figure 53 where two sensors are wired in series and connected to the trig pin on the HC-SR04 module.

Figure 53: Modified and Unmodified HCSR04 module

Figure 54 also shows environment of our experiment where sensors are surrounded by objects around with an out way which is 20 cm length.

Figure 54: Experiment Environment with one out way

Running Python code to scan the room around with modified and unmodified HC-SR04 module. Figure 55(a) and Figure 55(b) shows the result of our experiment.

(a) Experiment for unmodified and modified module

(b) Result of HC-SR04 modified module

Figure 55: Result from HC-SR04 modified

Notice modified method can detect outway but HCSR04 original module can not detect the outway. This experiment shows clearly that the resolution is improved. Python code used for this experiment is included in appendix named Scanning_Room.py

5.8 Robot Navigation

The robot navigation has two main parts

Hardware:

The hardware consists of two main parts also. A rotation system for sensors and Robot. The Radar has a motor which will rotate the array of sensors in 180 degrees to send and receive the echo from sensors. The robot consists the skeleton tank and a micro-controller (Raspberry pi)

Software:

1. A program which takes care of radar rotation and ultrasound scanning.
2. Analyzing the data we get from the first part to navigate robot to next available point which can be a potential outway. Then step 1 and 2 can be repeated

5.9 SLAM

Simultaneously Localization and Mapping is a main topic in Robot Navigation. To make any movement to the Robot environmental data through sensors are needed. Probably the environmental data needs to be updated after robot movement and from these data new position of the robot can be found. There are many algorithms for solving this problem. The most popular is A EKF algorithm based on Extended Kalman Filter theory. Kalman Filter is ideal for systems which are continuously changing. Kalman filters can be used in any dynamic system and you can make an educated guess about what the system is going to do next.

5.9.1 Landmark

A landmark is a feature which can be found from sensors data. Landmarks are re-observable which can help to define the position of the robot. One of the most important factors when choosing a landmark is that a landmark should have a unique feature so it does not get confused with a wrong landmark. In other words, if you re-observe two landmarks at a later point in time it should be easy to determine which of the landmarks is previously seen. If two landmarks are very close to each other this may be hard. Landmark extraction is a huge topic in computer vision, but only some part is mentioned which is used in this research.

5.9.2 Odometry

Odometry means finding position of a robot from rotation of wheels in our case. So this data will not be exact but an approximation. Since then our position will be a random variable with a Gaussian distribution. This variable will define the position of the robot with the highest probability. The step motors used has a predefined function which can be used to estimate the position of the robot by finding degrees of wheel rotation.

Figure 56: The robot initially measures using its sensors the location of the landmarks (sensor measurements illustrated with lightning).

5.9.3 Feature Extraction

After deciding which landmarks will be utilized, a method is needed to find a reliable extract from data. A method named RANSAC (Random Sampling Consensus) is used for this purpose. RANSAC is an iterative method to estimate the parameters of a mathematical model which contain outliers. For example finding Circles or Lines with this method is very efficient.

5.9.4 Line Extraction (RANSAC)

For efficient indoor navigation in a room detecting the corners, and, the walls of the room are essential. RANSAC (Random Sampling Consensus) method can be used to detect the corners, and, the walls. RANSAC uses voting mechanism to find the best fit line out of a data set consisting of inliers, and outliers. Each element of the data set is used to vote for one or more models. There are two assumptions in this voting mechanism:

1. The noisy elements will not consistently vote for a model,
2. There is enough information available to find a good model.

RANSAC method produces a result in which the outliers do not influence the result. Whereas, simple least square methods produce results in which both inliers, as well as outliers, have an impact on the result. Thus, RANSAC method provides a better result as compared to simple least square methods.

5.9.5 Algorithm

1. Given:
   - data – a set of observed data points
   - model – a model that can be fitted to data points
   - n – the minimum number of data values required to fit the model
   - k – the maximum number of iterations allowed in the algorithm
   - t – a threshold value for determining when a data point fits a model
   - d – the number of close data values required to assert that a model fits well to data
2. Return:
   - bestfit – model parameters which best fit the data (or nul if no good model is found)

Figure 57: RANSAC shows Inliers and Outliers

For fitting a line, defining a value d is needed. This value defines maximum distance between the line and a point which can be counted as inlier. Points which has a longer distance than d from the line will be counted as an outlier. The line with most inlier will be counted as the best fit line. Figure 57 shows one iteration and corresponds inliers and outliers.

There are several methods to iterate different lines with data to see which match better and get the highest vote. A simulation program is written in C (Ransac_ line_finder.c) to illustrate line fitting. The line is angular modeled with two parameters $l(\theta, \rho)$

As figure 58 shows any line can be defined with these two parameters θ and ρ. θ can be defined as an array from $-\pi/2$ to $\pi/2$ and ρ from 0 to Max. By calculating all combination of two parameters with defined range, the line with most inliers can be found. Variable d is used for deciding point corresponds as an inlier or outlier. Figure 57 shows an illustration of inliers and outliers.

Figure 58: Angular Line

5.9.6 Random variable

The position of robot and landmark position can be defined as given function:

xk=(p,v)

The measurements are never exact, and it's just an approach to real value. Gaussian distribution helps to implement variables as a random variable. Gaussian distribution is a continuous function which instead of giving us exact value tells us the probability of getting value in the range. In this case, the position of the robot is implemented as an approach to the real value. It can be defined as a random variable.

A random variable with a Gaussian distribution has a mean (μ) and variance (σ^2). As the first step, two random variables should be defined for the robot movement(Position and Velocity). Correlation between these two variables gives dependency or independence of them. A covariance matrix (Σ) is set where each element of the matrix (Σ_{ij}) is a degree of correlation between the ith and jth state variables.

Figure 59: The SLAM problem – An estimate of the robot, as well as, landmark location is required. The exact locations are not measured directly. Observations are made between the robot and the landmark locations.

5.10 Transfer function for Coordinate system

The position of the robot can be estimated in 3d worlds coordinate with a relative origo as x=y=z=0. The center of the robot then has a length, height and width relative to Origo. The sensors which measure distances to objects have its own coordinate system. Distance to a point in real world relative to sensor measurements can be found with help of matrix calculation. The pose of sensor relative to pose of Robot can be represented by a homogeneous transformation[34]

$$ {}^{s}\xi_r = \begin{bmatrix} {}^{s}R_r & {}^{s}t_r \\ 0_{3\times1} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & {}^{s}t_{rx} \\ r_{21} & r_{22} & r_{23} & {}^{s}t_{ry} \\ r_{31} & r_{32} & r_{33} & {}^{s}t_{rz} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24} $$

Where s = Sensor , r = Robot , R = 3x3 rotation matrix

The distance to a point object relative to Tank robot will be represented as below:

$$ {}^{r}p = {}^{s}\xi_r * {}^{s}p \tag{25} $$

$$\begin{bmatrix} {}^r x \\ {}^r y \\ {}^r z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^s R_r & {}^s t_r \\ 0_{3\times1} & 1 \end{bmatrix} * \begin{bmatrix} {}^s x \\ {}^s y \\ {}^s z \\ 1 \end{bmatrix} \tag{26}$$

Where R defines relative rotation of sensor coordinate system with robot coordinate system and t is distance from sensor to robot coordinate system.

Using equation 26 we can find the distance from the robot to object. The distance can be estimated by multiplying distance of point to sensor with pose transfer matrix of robot relative to sensor.

R matrix which is rotation matrix can be found with given matrix multiplication: Assuming $\theta_x, \theta_y, \theta_z$ as the rotation of coordinate system of the sensor in x, y and z axis relative to robot's coordinate system, then R matrix can be defined as below:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta_x) & -sin(\theta_x) \\ 0 & sin(\theta_x) & cos(\theta_x) \end{bmatrix} \tag{27}$$

$$R_y = \begin{bmatrix} cos(\theta_y) & 0 & sin(\theta_y) \\ 0 & 1 & 0 \\ -sin(\theta_y) & 0 & cos(\theta_y) \end{bmatrix} \tag{28}$$

$$R_z = \begin{bmatrix} cos(\theta_z) & -sin(\theta_z) & 0 \\ sin(\theta_z) & cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{29}$$

$${}^s R_r = R_x * R_y * R_z \tag{30}$$

Using R and t we can find transfer matrix. By multiplying transfer matrix with distance which we get from sensor data we can find the distance from an object to the robot. Illustration of distances are shown in figure 60

Using the transfer function in Python code we can simply find the distance from an object to robot corners, which we can use to determine if the size of outway is big enough.

Figure 60: Transfer function of robot to object

5.11 Navigating Robot through potential outway

For testing the robot, through an easy task, an outway can be made to examine how the robot detects and find a way through it. Each time robot discovers an outway, we can determine the size of it. Techniques for determining the size of outway in discussed at page 57. After scanning room around robot can choose the best candidate as an outway and navigate through it and repeat the procedure again. The robot is also tested to check if it can detect same features from two different points and make correspondences.

An illustration of the process for robot navigation is shown in the figure 61.

Figure 61: Transfer function of robot to object

Distance from object to the robot can be found using transfer function explained in 5.10. These data are collected in a file and get processed for detection of futures like outway and corners. Future with their specification will be saved in the file. After this part, the robot will make movement and try to scan the

room again. Data collected from scanning the room will again be processed to find previous or new futures. Processing these data gives an estimation of the position of robot and estimation of the right path to a possible outway. By using Python code scanning_room.py we can write a program to navigate the robot through outway by repeating steps in figure 61. Feature detection and robot navigation also can be done with help of theory in chapter 5.9.

(a) Bottom 1 (b) Bottom 2

Figure 62: Building The Robot

(a) Top (b) Top right

Figure 63: Building The Robot

6 Conclusion

Results of the experiments presented in this paper show that the resolution of ultrasonic sensors can be improved by array processing. This increased resolution facilitates the use of array processing of ultrasonic sensors to solve real world problems – road safety being one of them. Beamforming could also be used to improve resolution, however, owing to higher cost, and, electronic complexity, Ultrasonic Sensor Module - HCSR04 has been used in this research paper to demonstrate the results of the experiments.

The Rotating Scanner introduced in this paper can be integrated with other systems to achieve superior results. For example, the objects which the LiDAR system cannot detect can be detected, if the rotating scanner is connected to it.

Two sensors, connected in series, have been used in the experiments, in this paper, to serve as trigger sensors. This resulted in better resolution. Increasing the number of sensors, and, connecting them in unique ways, to form an array, can serve a specific purpose. For example – connecting several sensors in a circle can present a 360° view of the objects in a room; with enhanced resolution. Further research can be conducted to examine the results of diverse ways of arranging the sensors. Even the shape of the objects detected by the sensors can be determined. However, that is beyond the scope of this research paper. Ultrasonic sensors can be effectively utilized to determine the shape of transparent objects; as optical sensors fail to effectively detect transparent objects.

Appendices

Matlab code (Aperture smoothing function)

```
%% Aperture smoothing function for circular elemnt
clear all;
theta=-90:0.001:90;

c=340;
f=40000;
w=2*pi*f;
lambda = c/f;


R=[0.02,0.0104,0.0063];
R=R/2;

for i=1:length(R)

    X=(w/c)*sin(theta*pi/180);
    J = besselj(1,X*R(i));
    %J = airy(X*R);
    J = ((2*pi*R(i))./X ) .* J;
    J = J/max(J);
    J = 20*log10(J);
    %W= (2*pi*D) / sin((w/c)*theta);
    %W=W*J;
    plot(theta,J)
    hold on

end

plot(theta,ones(length(theta),1)*-3,'--')
plot(theta,ones(length(theta),1)*-6,'--')
hold off
xlabel('\theta')
ylabel('W(\theta) dB')
```

Other Matlab codes used

```
%% Wave equation
clear all;
x=-30:0.1:30;
y=zeros(size(x));

N=100;
temp=0:2*pi/N:2*pi;
temp=sin(temp);

for i=1:601-N

    y(i:N+i)=temp;
    plot(x,y)
    pause(0.001)

end



%% Sinx/x function 3d
clear all
A=-4*pi:0.2:4*pi;
N=numel(A);
X=repmat(A,N,1);
Y=X';

Z=zeros(N,N);
for i=1:N
    for j=1:N
        x=X(i,j);
        y=Y(i,j);
        Z(i,j)=abs(sin(sqrt(x^2+y^2)))/(sqrt(x^2+y^2));
        if(sqrt(x^2+y^2)>12 )
            Z(i,j)=0;
        end
    end
end


figure(1)
surf(X,Y,Z);
xlabel('\theta')
ylabel('\theta')
```

```
saveas(figure(1),'2DAperture.jpg')


%% Aperture smoothing function for circular elemnt
clear all;
theta=-20:0.001:20;

c=340;
f=40000;
w=2*pi*f;
lambda = c/f;


%R=[0.02,0.0104,0.0063];
R=[0.02];
R=R/2;

for i=1:length(R)

    X=(w/c)*sin(theta*pi/180);
    J = besselj(1,X*R(i));
    %J = airy(X*R);
    J = ((2*pi*R(i))./X ) .* J;
    J = J/max(J);
    J = 20*log10(J);
    %W= (2*pi*D) / sin((w/c)*theta);
    %W=W*J;
    plot(theta,J)
    hold on

end

plot(theta,ones(length(theta),1)*-3,'--')
plot(theta,ones(length(theta),1)*-6,'--')
hold off
xlabel('\theta')
ylabel('W(\theta) dB')


%% Array Responce of m elements
clear all;
theta=-90:0.001:90;

c=340;
f=40000;
w=2*pi*f;
```

```
lambda = c/f;


R=0.0104;

R=R/2;

d=0.017;


X=(w/c)*sin(theta*pi/180);

W=0;
for m=-2:2
    W=W+exp(-j*X*m*d);
end

J = besselj(1,X*R);
J = ((2*pi*R)./X ) .* J;
J = W.*J;
J = J/max(J);
J = 20*log10(J);

plot(theta,J)
hold on



%plot(theta,ones(length(theta),1)*-3,'--')
%plot(theta,ones(length(theta),1)*-6,'--')
hold off
xlabel('\theta')
ylabel('W(\theta) dB')


%% Geometrical transfer function
clear;


Thetax = pi/4;
Thetay = pi/3;
Thetaz = pi/8;

T=[1 3 4];

Rx = [1 0 0; 0 cos(Thetax) -sin(Thetax); 0 sin(Thetax)
```

```
cos(Thetax)];
Ry = [cos(Thetay) 0 sin(Thetay); 0 1 0; -sin(Thetay) 0 cos(Thetay)];
Rz = [cos(Thetaz) -sin(Thetaz) 0; sin(Thetaz) cos(Thetaz) 0; 0 0 1];

Rxyz= Rx*Ry*Rz

A_T_B = [Rxyz T';0 0 0 1]

P= [1 2 3 1];

P * A_T_B;



%% Testing linearity of Sensor
clear all

testdata = [10, 15, 20, 30, 40];
sensordata=[10.2, 15.2, 20.2, 30.3, 40.5];

p=polyfit(testdata,sensordata,1);

x=linspace(0,41,1000);
y=polyval(p,x);

scatter(testdata,sensordata,'ro')
hold 'on'
plot(x,y,'b')
hold 'off'

xlabel('x / testdata')
ylabel('y / sensordata')

modeldata= polyval(p,testdata);

saveas(figure(1),'linefit.jpg')

sum(((modeldata - sensordata).^2))
%% Resolution angle of sensor
clear all

a=[6, 11, 16];
b=[5.5, 7, 8.2];

```

```
p=polyfit(a,b,1);

x=linspace(0,40,1000);
y=polyval(p,x);

scatter(a,b,'ro')
hold 'on'
plot(x,y,'b')
%hold 'off'

xlabel('x');
ylabel('y');
axis([0 25 0 25]);

angle=(atan(p(1))*180)/pi
%saveas(figure(1),'find_angle_sensor.jpg')
% Resolution of 2 sensors echo
%%
clear all

%a=[ 10, 20, 30]; 2 sensor on trig one sensor in between on exho
%b=[ 4.2, 6, 7.5];
a=[5, 20];
b=[5, 7.3];

p=polyfit(a,b,1);

x=linspace(0,40,1000);
y=polyval(p,x);

scatter(a,b,'bo')
hold 'on'
plot(x,y,'b')
hold 'off'

xlabel('x');
ylabel('y');
axis([0 40 0 15]);

angle=(atan(p(1))*180)/pi
saveas(figure(1),'find_angle_sensor2.jpg')



%% Angular resolution Theoretical
```

```
lambda=c/f
((8.5*10^-3)*1.9)/0.2618

A = 110;
x = linspace(0,4*pi,1000);
y1 = A*sin(x);
phase = 120 * (2*pi / 360)
y2 = A*sin(x+phase)
y3 = A*sin(x+2*phase)

plot(x,y1)
hold on
plot(x,y2)
plot(x,y3)


plot(x,(y1+y2+y3))
```

Python Code (Testing HCSR04)

```
#import modules
import RPi.GPIO as GPIO
import time
GPIO.setwarnings(False)

#pin Definition:
ledPin_G = 20 # Led Green
ledPin_R = 21 # Led Red
trig = 12     # Trig pin
echo = 13     # Echo pin

#pin Setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin_G, GPIO.OUT) #Led pin green set as output
GPIO.setup(ledPin_R, GPIO.OUT) #Led pin  red  set as output
GPIO.setup(trig, GPIO.OUT)     #Setup trig pin as output
GPIO.setup(echo, GPIO.IN)      #Setup echo pin as input

#initial state for LEDs
GPIO.output(ledPin_G, GPIO.LOW)
GPIO.output(ledPin_R, GPIO.LOW)
time.sleep(0.5)
GPIO.output(ledPin_G, GPIO.HIGH)
GPIO.output(ledPin_R, GPIO.HIGH)
time.sleep(2)
GPIO.output(ledPin_G, GPIO.LOW)
GPIO.output(ledPin_R, GPIO.LOW)
time.sleep(0.5)

print "Distance measurement in progress"

while True:
    GPIO.output(trig, False)
    print "waiting for sentor to settle"
    time.sleep(1)

    GPIO.output(trig, True) #Set trig as HIGH so it sends 8 40HZ pulses
    time.sleep(0.000010)     #Wait 10 us as it stays in manual
    GPIO.output(trig, False)#Set trig LOW

```

```
    while GPIO.input(echo)==0:
        pulse_start = time.time()  #Save last known time where echo is LOW

    while GPIO.input(echo)==1:
        pulse_end = time.time()     #save last known time where echo is HIGH

    pulse_duration=pulse_end - pulse_start

distance = pulse_duration * 17150
    ditance = round(distance,2)
    if distance > 2 and distance < 400:
        print "Distance:", distance - 0.5, "cm"
else:
    print "Out of Range"
```

Python Code (Scanning Room)

```
from threading import *
import wx
from Dynamixel import dynamixel
import turtle

#import modules for ultrasonic sensor
import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)

#Dynamixel definition
ID = 1
SPEED_REG = 32
POS_REG = 30
PRESENT_POS = 36
MIN_ANGLE=400
MAX_ANGLE=600
step=5

# Button definitions
ID_START = wx.NewId()
ID_STOP = wx.NewId()



#pin Definition:
ledPin_G = 20 # Led Green
ledPin_R = 21 # Led Red
trig = 12      # Trig pin
echo = 13      # Echo pin

#pin Setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin_G, GPIO.OUT) #Led pin green set as output
GPIO.setup(ledPin_R, GPIO.OUT) #Led pin  red  set as output
GPIO.setup(trig, GPIO.OUT)     #Setup trig pin as output
GPIO.setup(echo, GPIO.IN)      #Setup echo pin as input



# Define notification event for thread completion
```

```
EVT_RESULT_ID = wx.NewId()

def EVT_RESULT(win, func):
    """Define Result Event."""
    win.Connect(-1, -1, EVT_RESULT_ID, func)

class ResultEvent(wx.PyEvent):
    """Simple event to carry arbitrary result data."""
    def __init__(self, data):
        """Init Result Event."""
        wx.PyEvent.__init__(self)
        self.SetEventType(EVT_RESULT_ID)
        self.data = data

# Thread class that executes processing
class WorkerThread(Thread):
    """Worker Thread Class."""
    def __init__(self, notify_window):
        """Init Worker Thread Class."""
        Thread.__init__(self)
        self._notify_window = notify_window
        self._want_abort = 0

        #initialize dynamixels
        speed = 255
        self.ax12 = dynamixel()
        self.ax12.set_ax_reg(ID, SPEED_REG, ([(speed%256),(speed>>8)]))
        #set Angle Limit
        self.ax12.set_ax_reg(ID, 6, ([(MIN_ANGLE%256),(MIN_ANGLE>>8)]))
        self.ax12.set_ax_reg(ID, 8, ([(MAX_ANGLE%256),(MAX_ANGLE>>8)]))


        # This starts the thread running on creation, but you could
        # also make the GUI thread responsible for calling this
        self.start()

    def run(self):
        """Run Worker Thread."""
        #Paintin 2D SCAN
        dw = turtle.Turtle()

```

```
        dw = turtle.Turtle()
        dw.pensize(5)
        dw.speed(100)
        dw.hideturtle()
        dw.color("red")

        # This is the code executing in the new thread. Simulation of
        # a long process (well, 10s here) as a simple loop - you will
        # need to structure your processing so that you periodically
        # peek at the abort variable

        #read Current Position
        val=0
        time.sleep(0.3)
        chc = self.ax12.get_reg(ID, 2, PRESENT_POS, 2)
        if(chc):
            val= (256*chc[1])+chc[0]

        if(val<MIN_ANGLE):
            self.ax12.set_ax_reg(ID, POS_REG, ([(MIN_ANGLE%256),(MIN_ANGLE>>8)]))

        dw.penup()
        flag=True
        while(1):
            chc = self.ax12.get_reg(ID, 2, PRESENT_POS, 2)
            if(chc):
                val= (256*chc[1])+chc[0]

            #print val

            dw.seth((val/(1024.0-MIN_ANGLE))*360.0)
            if(flag):
                val=val+step
                if(val>=MAX_ANGLE):
                    flag=False
            else:
                val=val-step
                if(val<=MIN_ANGLE):
                    flag=True
                    dw.clear()
            self.ax12.set_ax_reg(ID, POS_REG, ([(val%256)
```

```
                ,(val>>8)]))


            GPIO.output(trig, False)
            #print "waiting for sentor to settle"
            time.sleep(0.2)

            GPIO.output(trig, True) #Set trig as HIGH so it sends 8 40HZ pulses
            time.sleep(0.000010)      #Wait 10 us as it stays in manual
            GPIO.output(trig, False)#Set trig LOW

            while GPIO.input(echo)==0:
                pulse_start = time.time()  #Save last known time where echo is LOW

            while GPIO.input(echo)==1:
                pulse_end = time.time()     #save last known time where echo is HIGH

                pulse_duration=pulse_end - pulse_start

                distance = pulse_duration * 17150
                ditance = round(distance,2)


                if distance > 2 and distance < 400:
                print "Distance:", distance - 0.5, "cm"
                dw.pendown()
                dw.forward(distance*2)
                dw.penup()
                    dw.back(distance*2)
                else:
                dw.pendown()
                dw.forward(200)
                dw.penup()
                dw.back(200)
                    print "Out of Range"

                time.sleep(0.2)
                    if self._want_abort:
                # Use a result of None to acknowledge the abort (of
                # course you can use whatever you'd like or even
```

```
                    # a separate event type)
                    wx.PostEvent(self._notify_window, ResultEvent(None))
                    turtle.bye()
                            return
                                # Here's where the result would be returned (this is an
                                # example fixed result of the number 10, but it could be
                                # any Python object)
                                wx.PostEvent(self._notify_window, ResultEvent(10))

                                def abort(self):
                                    """abort worker thread."""
                                    # Method for use by main thread to signal an abort
                                    self._want_abort = 1



# GUI Frame class that spins off the worker thread
class MainFrame(wx.Frame):
    """Class MainFrame."""
    def __init__(self, parent, id):
        """Create the MainFrame."""
        wx.Frame.__init__(self, parent, id, 'Thread Test')

        # Dumb sample frame with two buttons
        wx.Button(self, ID_START, 'Start', pos=(0,0))
        wx.Button(self, ID_STOP, 'Stop', pos=(0,50))
        self.status = wx.StaticText(self, -1, '', pos=(0,100))

        self.Bind(wx.EVT_BUTTON, self.OnStart, id=ID_START)
        self.Bind(wx.EVT_BUTTON, self.OnStop, id=ID_STOP)

```

```
195         # Set up event handler for any worker thread
                results
196         EVT_RESULT(self, self.OnResult)
197
198         # And indicate we don't have a worker thread yet
199         self.worker = None
200
201
202     def OnStart(self, event):
203         """Start Computation."""
204         # Trigger the worker thread unless it's already
                busy
205         if not self.worker:
206             self.status.SetLabel('Starting computation')
207             self.worker = WorkerThread(self)
208
209 def OnStop(self, event):
210     """Stop Computation."""
211         # Flag the worker thread to stop if running
212         if self.worker:
213             self.status.SetLabel('Trying to abort
                    computation')
214             self.worker.abort()
215
216                 def OnResult(self, event):
217                     """Show Result status."""
218 if event.data is None:
219     # Thread aborted (using our convention of None return
            )
220     self.status.SetLabel('Computation aborted')
221         else:
222             # Process results here
223             self.status.SetLabel('Computation Result: %s'
                    % event.data)
224     # In either event, the worker is done
225     self.worker = None
226
227
228 class MainApp(wx.App):
229     """Class Main App."""
230     def OnInit(self):
231         """Init Main App."""
232         self.frame = MainFrame(None, -1)
233         self.frame.Show(True)
234         self.SetTopWindow(self.frame)
235         return True
```

```

if __name__ == '__main__':
    app = MainApp(0)
    app.MainLoop()
```

C code (RANSAC line finder (simulation))

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <unistd.h>


#define WIDTH 100
#define HEIGHT 50
#define X 0
#define Y 0
#define XMAX WIDTH
#define XMIN 0
#define YMAX HEIGHT
#define YMIN 0
#define PI 3.14159265358979323846


char grid[WIDTH][HEIGHT];
int plot(int x, int y);
void init_grid(void);
void show_grid(void);
void makerandomarray(int *temp,int N);
void printArray(double *temp, int N);
void countNumber(int *temp, int number, int N);
void plot_xy(int *xtemp, int *ytemp, int N);
void add_line_pattern(int *xtemp, int *ytemp, int N);
void find_line();
void make_line_on_xy(int *xtemp, int *ytemp, double a,
    double b, int N);

int *x_array;
int *y_array;




int main(int argc, char** argv) {

    int N = 100;
    //printf("Answer is  = %d\n", b);
    //printf("xarray = %d \n", &x_array);

    x_array = (int *) malloc(N*sizeof(int));
```

```
    y_array = (int *) malloc(N*sizeof(int));

    makerandomarray(x_array,N);  // Make random array
    makerandomarray(y_array,N);  // Make random array
    add_line_pattern(x_array, y_array, N);
    //printArray(x_array, N); //Prints Array elements
    //printArray(y_array, N);

    init_grid();
    plot_xy(x_array,y_array, N);

    //plot(1,1);
    //plot(rintf(x*10),rintf(y*8));

    show_grid();
    usleep(300000);
    find_line(x_array, y_array);


    //printf("%d", sizeof(int));
    usleep(300000);
    free(x_array);
    free(y_array);

    return (0);
}


void makerandomarray(int *temp, int N)
{
    for (int i = 0; i < N; i++) {
        *(temp+i) = rand() %100 ;
    }
}

void printArray(double *temp,int N)
{
    for(int i = 0; i < N; i++){
        printf("%2d : %f \n", i+1, temp[i]);
    }
}

void countNumber(int *temp, int number, int N)
{

    int counter=0;
```

```
    for(int i=0; i < N; i++){
        if (number == temp[i])
            counter++;

    }
}


int plot(int x, int y)
{
    if( x > XMAX || x <= XMIN || y > YMAX || y <= YMIN )
        return(1);
    //if(x != 0 && y != 0)
    else
        grid[x][y] = '*';
    return(1);
}


void init_grid(void)
{
    int x,y;

    for(y=0;y<HEIGHT;y++)
        for(x=0;x<WIDTH;x++)
            grid[x][y] = ' ';
    /* draw the axis */
    for(y=0;y<HEIGHT;y++)
        grid[0][y] = '|';
    for(x=0;x<WIDTH;x++)
        grid[x][0] = '-';


}


void show_grid(void)
{
    int x,y;

    for(y=0;y<HEIGHT;y++)
    {
        for(x=0;x<WIDTH;x++)
            putchar(grid[x][HEIGHT-y-1]);
        putchar('\n');
    }
```

```
}

void plot_xy(int *xtemp, int *ytemp,int N)
{

    for(int i=0; i<N; i++)
        plot(xtemp[i],ytemp[i]/2);
}

void add_line_pattern(int *xtemp, int *ytemp, int N)
{
    for(int i=1; i<N-10; i=i+5){
        xtemp[i]=i;
        ytemp[i]=i+25;
    }
}


void find_line(int *xtemp, int *ytemp)
{

    int N=100;
    double theta[N];
    double r[N];
    double x, y;
    double a, b;
    double distance;
    int vote;
    int best_vote=1;
    double best_theta;
    double best_r;
    double best_a;
    double best_b;

    for(int i=-N/2; i<N/2; i++){
        theta[i+N/2] = i*(PI/N);
        //printf("%2f\n", cos(theta[i])/sin(theta[i]));
        //printf("%2f\n", sin(theta[i]));
    }
    //printArray(theta,N);
    for(int i=0; i<N; i++)
        r[i] = i;

    for(int i=0; i<N; i++){
        for(int j=0; j<N; j++){

```

```
            x=r[i] * cos(theta[j]);
            y=r[i] * sin(theta[j]);
            a=x/y;
            b=y+a*x;
            vote=0;
            for(int m=0; m<N; m++){

                distance = ( ytemp[m] - a*xtemp[m] - b ) / sqrt(pow(a,2)+1);
                if(fabs(distance)<2 && fabs(distance)>0.01){
                    vote++;
                    //printf("%2f\n", fabs(distance));
                }
            }
            if(vote>best_vote){
                best_vote = vote;
                best_theta = theta[j];
                best_r = r[i];
                best_a=a;
                best_b=b;
            }

            //printf("%2f\n",theta[i]);

        }

    }
    printf("Best vote : %d \n", best_vote);
    printf("Best theta : %f = %f degree \n", best_theta, best_theta*180/PI);
    printf("Best r : %f \n", best_r);

    int *x_array2 = (int *) malloc(N*sizeof(int));
    int *y_array2 = (int *) malloc(N*sizeof(int));
    make_line_on_xy(x_array2, y_array2, best_a, best_b, N);
    sleep(1);
    free(x_array2);
    free(y_array2);


}


void make_line_on_xy(int *xtemp, int *ytemp, double a,
```

```
    double b, int N)
{
    for (int i = 0; i < N; i++) {
        xtemp[i]=i;
        ytemp[i]=i*a+b;
    }
    init_grid();
    plot_xy(xtemp,ytemp, N);
    show_grid();



}
```

References

[1] A. Jones A. Ward and A. Hopper. A new location technique for the active office. (English). *IEEE Personal Communications*, 4(5):42–47, 1997.

[2] Fahrudin Smajlovic Aladin Carovac. Application of Ultrasound in Medicine. (English). *AIM*, 19(3):168–171, 2011.

[3] asirt.org. *Annual Global Road Crash Statistics*. ASIRT, 2017.

[4] F. Barfield. Autonomous collision avoidance: the technical requirements. (English). *Proceedings of the IEEE 2000 National Aerospace and Electronics Conference*, pages 808–813, 2000.

[5] Michael Barnard. *Tesla & Google Disagree About LIDAR — Which Is Right*. https://cleantechnica.com/2016/07/29/tesla-google-disagree-lidar-right/, 2017.

[6] Dennis A. Bohn. Environmental Effects on the Speed of Sound. (English). *Rane Corporation,*, 36(4), 1988.

[7] Andrew K. Dennis. *Raspberry pi home automation with arduino*. Packt Publishing, 2013.

[8] Peter Dizikes. Wireless highway: networked-car safety research hits the road.(TRANSPORTATION). (English). *Cambridge, Mass.*, 109(1):25–27, 2006.

[9] Elizabeth Vidal Duarte. Teaching the First Programming Course with Python's Turtle Graphic Library. (English). *ACM*, pages 244–248, 2016.

[10] E. Dudgeon and Dan E. Dudgeon. *Array Signal Processing: Concepts and Techniques*. Prentice Hall, 1993.

[11] MOUSER ELECTRONICS. *Stepper Motors Make the Right Moves with Precision, Ease and Smarter Drivers*. MOUSER ELECTRONICS, 2015.

[12] Elyes Ben Hamida. Security of Cooperative Intelligent Transport Systems: Standards, Threats Analysis and Cryptographic Countermeasures. (English). *www.mdpi.com/journal/electronics*, 4(1):380–423, 2015.

[13] Monson H. Hayes. *Statistical Digital Signal Processing and Modeling*. Wiley, 1996.

[14] Liu K. R. Haykin, S. *Handbook on array processing and sensor networks*. M, 2010.

[15] S. Haykin and K.J.R. Liu. *Handbook on Array Processing and Sensor Networks , Adaptive and Learning Systems for Signal Processing, Communications, and Control Series*. Wiley, 2010.

[16] Oleg V. Minin Igor V. Minin. *Basic Principles of Fresnel Antenna Arrays*. Springer, 2008.

[17] kongsberg.com. *Autonomous Underwater Vehicle - AUV The HUGIN Family*. kongsberg, 2017.

[18] J. ; Bilík P. Koval, L. ; Vaňuš. Distance Measuring by Ultrasonic Sensor. (English). *IFAC PapersOnLine*, 49(25):153–159, 2016.

[19] M. Krim, H.; Viberg. Two Decades of Array Signal Processing Research. (English). *IEEE Signal Processing Magazine*, pages 67–94, 2010.

[20] labelsensors.com. *Label Sensor Types and Technologies*. labelsensors.com, 2017.

[21] Simon Monk. *Raspberry Pi Cookbook: Software and Hardware Problems and Solutions*. O'Reilly Media, 2016.

[22] A. Chakraborty N. Priyantha and H. Balakrishnan. The cricket location-support system. (English). *Proceedings of the 6th annual international conference on mobile computing and networking*, pages 32–43, 2000.

[23] K. Ohtani. Shape recognition for transparent objects using ultrasonic sensor array. (English). *SICE Annual Conference*, pages 1813–1818, 2007.

[24] G. ; Piracci E. Petrochilos, N. ; Galati. Array processing of SSR signals in the multilateration context, a decade survey. (English). *IEEE , Tyrrhenian International Workshop on Digital Communications*, pages 1–5, 2008.

[25] Grand View Research. *Collision Avoidance Sensors Market Analysis*. GVE, 2017.

[26] Peter E. Hart Richard O. Duda and David G. Stork. *Pattern Classification 2nd Edition*. Wiley-Interscience, 2000.

[27] S. Rostad S. Holm, O. B. Hovind and R. Holm. Indoors data communications using airborne ultrasound. (English). *Proceedings*, 3:957–960, 2005.

[28] David ; Little Jim Se, Stephen ; Lowe. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. (English). *The International Journal of Robotics Research*, 21(8):735–758, 2002.

[29] Timothy Short. *Raspberry Pi 3: Beginner to Pro Guide:: (Raspberry Pi 3, Python, Programming)*. CreateSpace Independent Publishing Platform, 2017.

[30] PhET Interactive Simulations. *Simulator for Wave Interference*. PhET CAROLINA, 2017.

[31] J. Pozo P.L.M.J. van Neer B. Snyder P. O'Brien H.P. Urbach N. de Jong S.M. Leinders, W.J. Westerveld and M.D. Verweij. A sensitive optical micro-machined ultrasound sensor (OMUS) based on a silicon photonic ring resonator on an acoustical membrane. (English). *Scientific Reports (Nature Publisher Group)*, 5:14328, 2015.

[32] Matthew ; Cheeseman Peter Smith, Randall ; Self. Estimating Uncertain Spatial Relationships in Robotics. (English). *Computer Science - Artificial Intelligence. Cornell University*, 1986.

[33] E.G. ; Gershman A.B. Stoica, P. ; Larsson. Signal Processing and Analysis. (English). *IEEE Signal Processing Letters*, 8(5):148–150, 2007.

[34] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2011.

[35] Telenor.com. *5G demo and pilot testing*. Telenor.com, 2017.

[36] E. Tuncer and B. Friedlander. *Classical and Modern Direction-of-Arrival Estimation*. Academic Press, 2010.

[37] H. L. Van Trees. *Optimum Array Processing*. Wiley, 2002.

[38] George S. K. Wong. Effects of humidity on the characteristic impedance in air. (English). *Program Of The 111th Meeting Of The Acoustical Society Of America*, 79:20, 1986.

[39] www.mouser.com. *HC-SR04 Ultrasonic Range Finder*. MOUSER ELECTRONICS, 2015.

[40] www.persistencemarketresearch.com. *Distance Measurement Sensor Market: Global Industry Analysis and Forecast*. www.persistencemarketresearch.com, 2010.

[41] www.robotis.com. *Dynamixel AX12A e-manual v 1.29.00*. www.robotis.com, 2017.

[42] L.J. Zunich, G.T. ; Griffiths. A Robust Method for Adaptive Array Processing. (English). *IEEE Conference Publications, Conference Record Twenty-Fourth Asilomar Conference on Signals, Systems and Computers*, 1:23, 1990.

[39, 41, 11, 38, 6, 31, 1, 27, 22, 40, 24, 42, 28, 32, 36, 20, 15, 19, 37, 33, 23, 4, 35, 8, 3, 30, 17, 2, 16, 9, 18, 14, 10, 13, 34, 26, 7, 29, 21, 12, 5]