Project Number: YR-11E1

Obstacle Avoidance Robot

A Major Qualifying Project Report

Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for

The Degree of Bachelor of Science Date:

2011-8-3

WPI Project Team

Paul Kinsky

Quan Zhou

Advisor: Professor YimingRong

HUST Project Team

Zhaoliang Yang

Min Li

Weijie Zhang

Advisor: Professor Lingsong He

Co-Advisor: Michael A. Gennert

Abstract

A robot using computer vision to avoid obstacles was built for Depush, a Chinese company specializing in educational robotics. We used the Open Computer Vision library to implement stereo vision for obstacle detection. We then sent commands to the motors using a microcontroller. This robot successfully detected and avoided different kinds of obstacles such as bottles, chairs and walls.

Acknowledgements

Professor Rong: For setting up the MQP in China and making sure that we were taken care of every step of the way. Nice comments of our work and necessary meetings during the process.

Professor He: For helping us to work with the HUST facilities and enthusiastically supporting our work

Professor Michael A. Gennert: For helping us with computer vision.

Mr. Zhang of Depush: For financial support and manufacturing our components

Lei Bei: helped us a lot from the beginning to the end of this project.

Yiming Wu: For help testing the microcontroller

Fengjin Chai For assistance with motor control programming

East 3 Lab: For providing us a place to work with air-conditioning in the summer.

East 1 Manufacturing Facility: For manufacturing our components

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

Robotics is a growing field. This has caused many universities to offer classes and programs in the field of robotics that combine elements of electrical engineering, mechanical engineering and computer science. Additionally, project-based learning is an important part of learning an engineering discipline.

For that reason, many of these schools use educational robots as experimental platforms. These robots are built to perform basic functions such as line following and obstacle avoidance. Students can then program them to perform tasks such as collecting small balls or travelling from one area to another. The concepts behind a robot that carries radioactive fuel rods and one that carries red Ping-Pong balls are very similar.

Our sponsor Depush is a relatively young engineering educational platform supplier. They offer a range of educational robotics platforms. Depush wants to add robots that use Computer Vision to their products. Computer vision is the area of computer science that tries to let computers see.

Objective 1: Provide Depush with a project that shows the potential of OpenCV and computer vision

Computer vision is the science of interpreting images and video with computers. The potential functions of OpenCV include tracking the position of objects and detecting obstacles in a scene.

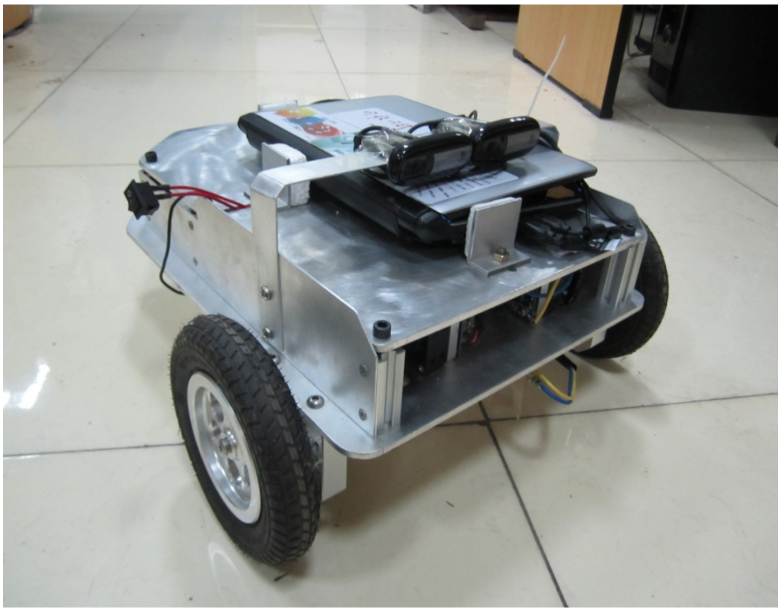
Objective 2: Develop a program in OpenCV that can detect different types of obstacles using video captured by webcams

Although the control part of different robots might be different, the vision part and decision-making part could be compatible with any other kind of robot. We endeavor to develop a program with universality which could be taken advantage by different robots.

Objective 3: Build a robot that can use this program to avoid different types of obstacles

The different types of obstacles include chairs, desks, bottles, walls etc. Despite their differentappearances, the principles of their detection with stereo vision are the same.

Depush currently has no experience with OpenCV. This project gives them a demonstration of some uses of computer vision in the field of robotics. Our can react rapidly to navigate from one point to another point with no touch of any of the obstacles we set out.



Our robot

Chapter 2: Background

Depush

Our sponsor Depush is a relatively young engineering educational supplier for which is founded in 2001. Currently, they have headquarters in ShenzhengFutian District and a subsidiary company in Wuhan. As a pioneer in this market, the average age of their engineers and technical employees is 27 years old. This gives them a great deal of flexibility and energy. As a new entry into their market, their plan is to be a creative engineering education supplier with new methods and innovative experimental platforms.

From the birth of the company, they have been dedicated to exploring new methods for higher engineering education. They absorbed the advanced education concepts from the USA and Hong Kong and developed Prof. Jiang's theory of "process-based teaching method" into an objective-based method.

Most products of Depush have a magnificent reputation in the market as a result of this objective-based method. To get practical skills and comprehensive quality in the courses, engineering students usually need real objectives to study and operate on. Depush's products let them work on real projects, with real objectives.

Over 300 colleges and universities purchase their products and solutions for engineering courses on the mainland. As time goes by, more and more schools cooperate with Depush to chase after the steps of advanced techniques.

Products of Depush

Depush’s products can be categorized into two categories: one is measurement and control experiment product series; the other is Depush educational robotic product series. Our work is about educational robots and Depush has four kinds of educational robots, namely elementary educational robots, professional educational robots, innovative research-based robots and robot extension series.

Table 1: Products of Depush

	name	function	technology		
			MCU	Servos/motor control	sensors
Elementary educational robots	Walking robot	Walk by shifting weight center	BS microcontroller teaching board	2 servo motors	IR sensors
		Turn by sliding feet			
	Graffiti robot	Draw the track it passes	BASIC STAMP microcontroller	2 DC motors	2 IR sensors
		Light following; obstacle avoidance & detection; track guidance			
	BoeBot-C51/BS2/AVR version	Obstacle avoidance; track guidance	BS microcontroller teaching board; AT89S52 microcontroller	2 imported successively turning servo motors	IR sensor; light dependent resistor; whisker sensor
	Sumo robot	Used in contest	BS	2 imported	QTI line

		Detect opponent/boundary; push opponent out	microcontroller teaching board; BS2 microcontroller	successively turning servo motors	following sensor; IR sensor
Professional educational robots	Outdoor heavy-load armor robot	Bionic robot’s behavior control and gait research	BS microcontroller teaching board; 2 PSC motors controller	18 servo motors	IR distance detection suite
		Complex electromechanical system control study			
	5 degrees of freedom mechanical hand-BS2 version	Mechanical hand control and research	BS microcontroller teaching board; 1 PSC motor controller	5 servo motors	IR distance detection suite
		Complex electromechanical system control study			
	Hexcrawler robot	Bionic robot’s behavior control and gait research	BS microcontroller teaching board	12 servo motors	IR distance detection suite
		Complex			

		electromechanical system control study			
	Quadcrawler robot	Bionic robot’s behavior control and gait research	BS microcontroller teaching board	8 servo motors	IR distance detection suite
		Complex electromechanical system control study			
	Outdoor heavy-load armor robot-BS2 version	Bionic robot’s behavior control and gait research	BS microcontroller teaching board; 2 PSC motor controllers	18 servo motors	IR distance detection suite
		Complex electromechanical system control study			
	Hexcrawler columnar armor robot-BS2 version	Bionic robot’s behavior control and gait research	BS microcontroller teaching board; 1PSC motor controller	12 servo motors	IR distance detection suite
		Complex electromechanic			

		al system control study			
	Quadcrawler columnar armor robot-BS2 version	Bionic robot’s behavior control and gait research	BS microcontroller teaching board; 1PSC motor controller	8 servo motors	IR distance detection suite
		Complex electromechanical system control study			
	Humanoid robot	Walk like humans	MR-C3024 robotic control board	16 servo motors	
		Imitate humans’ highly difficult movements			
	6 degrees of freedom mechanical hand-BS2 version	Bionic robot’s behavior control and gait research	BS microcontroller teaching board; 1PSC motor controller	6 servo motors	IR distance detection suite
		Complex electromechanical system control study			
Innovative research-ba	Medium sized football	Used in football	It has been sold out		

sed robots	robot	robot contest	

Some conclusions about Depush’s products:

1. Most of Depush’s products focus on microcontroller and motor control
2. Depush’s products do not use cameras as sensors, thus OpenCV technology is useful for Depush
3. Depush does not have products to train students to program using Python language

OpenCV

OpenCV (Open Computer Vision) is a library that implements many algorithms commonly used in the field of computer vision. Computer vision is the area of computer science that focuses on extracting structured information from images. Images as they are stored on computers are large, unstructured, two dimensional arrays of pixels. Computer vision techniques can also be applied to videos, which are stored as sequences of images. OpenCV provides algorithms that can be used for tasks such as locating faces in an image, recognizing predefined objects and shapes and detecting movement in a video. OpenCV also provides the infrastructure necessary for working with images and videos. OpenCV is regularly updated, with documentation in several languages including English and Chinese.[5]

Applications

Many computer vision algorithms are implemented by OpenCV, with a wide range of applications. Instead of attempting to list every algorithm and give a detailed technical explanation, we will list some common computer vision problems that OpenCV can be used to solve.

OpenCV provides many functions useful for recognizing objects within an image. These range from the simple task of finding an object of one color against a background of another to the more complicated process of finding the faces of people in an image. OpenCV also provides algorithms for machine learning, where the program is shown a very large set of images of one type of object and uses this information to detect instances of this object in other images. For example, a program can be trained to recognize letters and numbers in an image. Such a program could then be used to transform the scanned pages of a book into text files.

OpenCV also provides functions for analyzing motion between frames of a video. These functions are mostly concerned with determining which parts of the image moved between frames, as well as the direction and distance of this movement. One application of these functions is separating those parts of an image that are in motion from those that are not. This has applications in automated security – a camera could be programmed to send an alert only after it detects motion past a certain threshold.

OpenCV can also be used to extract three-dimensional information from two or more views of an object. This technique is called stereo vision when used with two cameras. This is useful for mobile robots, which often must navigate an unknown environment. Several NASA rovers, for example, have used stereo vision to navigate the unknown surface of other planets.

OpenCV also provides the infrastructure necessary for working with images. It provides code objects for handling image capture from webcams, images, and matrices. It also provides functions that implement common operations such as inverting and smoothing images. OpenCV was originally written in C, a commonly used low-level programming language. It can also be used with Python and Java, two commonly used high-level programming languages.[5]

Python

Python is a high-level programming language. In this context, high level means that it handles many of the tasks that programmers need to handle in low-level languages such as C. This makes is much easier to for beginners to write code for, because it does not require them to learn the details of the computer architecture. Python also has many modules available that deal with tasks ranging from analyzing images to programming video games. This lets programmers focus on the problem they are trying to solve instead of forcing them to reinvent the wheel.

For an example of the differences between high and low level languages, a C programmer working with an array would first declare a variable. Then he would need to allocate memory, and then write more code to iterate over the array if to perform an operation on each of its members. If he forgot to deallocate the memory when he was done using the array, it would cause errors with his program. In Python, creating and allocating the memory for an array takes one line of code. The Python equivalent of an array is called a list, and it comes with built-in functions for sorting, removing or adding items. However, there is a tradeoff. Low-level languages allow the programmer to customize details such as when and how memory is allocated and deallocated. This lets them write code that takes less time and system resources to run. However, high-level languages like Python generally consume more system resources but free programmers from having to consider time-consuming tasks such as memory management. This can save hours of time. It also lets programmers write more concise, readable code. Both high and low-level languages are widely used for different applications.

Another benefit of Python is that it has a very large number of open-source modules. These modules can be freely downloaded and used, and provide functions for applications ranging from reading and writing CAD files (dxf-reader), to writing video games (pygame), to interpreting and analyzing images (OpenCV). Web pages can also be written in Python (Django). Python itself is also open-source, meaning that it can be freely used without payment of licensing fees. These modules, combined with the flexibility of Python, let programmers use it for a wide range of applications. They also hide many of the complicated details behind layers of abstraction. For example, one does not need to know precisely how shapes are stored in a CAD file to output in that format.

Because of Python's ease of use it is gaining wide acceptance in industry. For example, Google is powered by Python. Google has even hired the creator of the programming language as part of a team dedicated to improving the language. This shows that Python is a major programming language with a large user base. It also shows that Python will almost certainly continue to be updated and supported for the foreseeable future.

Value to Depush

Depush aims to be a comprehensive server for innovative engineering education. Until now, its products have mainly focused on micro controllers and motor control. As computer vision is expanding very quickly in many fields and becoming widely taught in engineering education, Depush is becoming more interested in developing computer vision products. OpenCV technology is widely used to process the images sent by cameras. At the same time, obstacle avoidance is one of the most important applications of computer vision, thus our core work—OpenCV and obstacle avoidance—is quite significant for Depush in the market. By the way, we are the first users of Depush's newly developed teaching board and we can provide experience and formal instructions on how to use it.

In conclusion, computer vision/image processing is a relatively new field and is going to play an important role in Depush's future market. Computer vision and OpenCV technology is new to Depush and thus our work is valuable to Depush.

The application of this robot is to teach basic engineering knowledge in a way that is applicable to real-world problems. For the mechanical part, small changes could be made by them to previous work to complete more functions. However, the core value for them is to study the programming of both the software and hardware.

For software, we take advantage of Keil4 to program the micro-controller and OpenCV to handle the image processing. Our product has the advantage that it is simple enough for beginners to study the AT89S52 chip with basic knowledge of the C language which is a freshman course in most engineering schools.

About the OpenCV part, students are strongly recommended to have a background of image processing with python. Even though some people will analyze the picture with C++, we choose to program with Python which is more concise. This whole part is a combination of many functions in OpenCV which could be modified with new functions to get a better effect.

For hardware, students can learn the interface and electrical schematic with the documentation of the development board. The interpretation of every program sentence could be made to clarify the control methods of the motors. Based on this, students can change the PWM duty ratio or try to control the speed based on their requirements.

Obstacle avoidance techniques

Two categories of obstacle avoidance technique were considered for this project. In the first category are monocular techniques. These techniques require only a single camera, and work by analyzing the image and dividing it into regions. The second category is stereo techniques. These techniques work by comparing images from multiple cameras with different viewpoints, and attempting to recreate the geometry of the scene from these images.

For all of these techniques, two types of possible error are considered. The first is false positives. A false positive is when a technique labels an area that is not an obstacle as an obstacle. The second type of error is false negatives. This type of error occurs when the technique fails to detect an obstacle and labels an obstacle as free space.

Monocular Techniques

Two monocular techniques were considered, an edge based technique and a color based technique.

The first technique worked by analyzing edges found in an image. For this technique, the camera would be pointed forwards and down. The area at the bottom of the image would be assumed to be the floor, and any edges found would be assumed to be the bottom edges of obstacles.

This technique made two assumptions. First, it assumed that edges correspond to obstacles, and only to obstacles. This caused it to fail in several situations. For example, it produced false positives when the floor was scratched or dirty or when the floor was made out of light colored tiles with dark lines between them. This caused it to fail in many real-world situations.

Second, this technique assumed that the area at the bottom of the image was part of the floor. This caused it to produce false negatives when an obstacle was close enough to the robot that the robot could not see the floor. It would then assume that the obstacle was the floor.

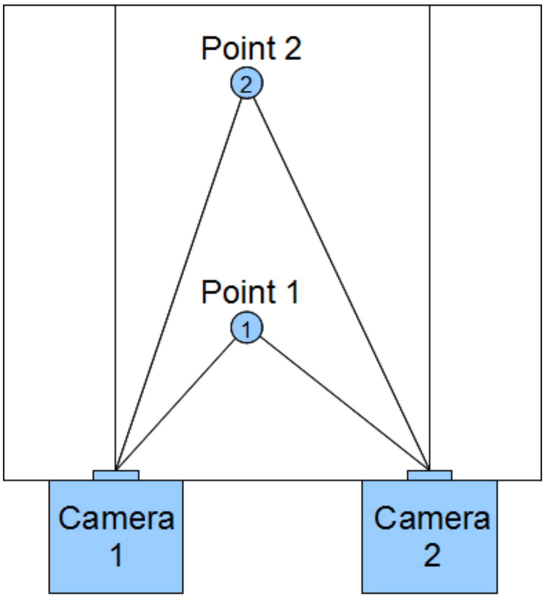
The second technique attempted to segment the image into the floor area and everything else. It worked by starting with a seed region at the bottom of the image that was assumed to be part of the floor. It would then attempt to grow this region, adding any pixels with a color similar to the average color of the seed region to that region, repeating this process until (hopefully) the region covered the part of the image corresponding to the floor.

This technique made two assumptions. First, it assumed that obstacles were a different color from the floor. This caused it to fail in several situations. For example, it produced false negatives when obstacles were the same color as the floor. It also produced false positives when the floor was composed of areas with sufficiently different color values.

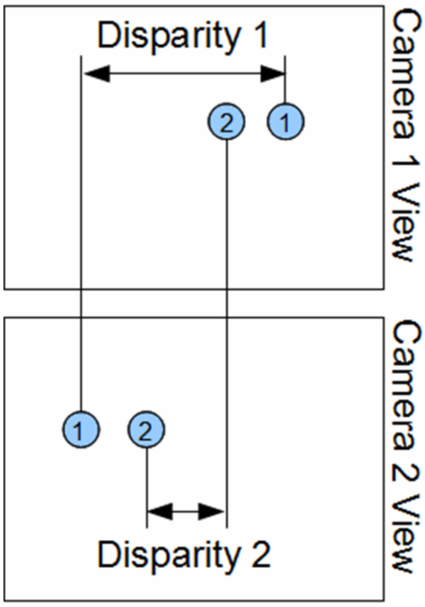
Second, it assumed that the region at the bottom of the image was part of the floor. This could cause it to produce false negatives when an obstacle blocked the robot’s view of this region. It would then label that obstacle as the floor.

Stereo techniques

Stereo vision works by comparing two or more views of the same scene taken from different perspectives. Objects that are close to the cameras have a large difference in position between the two views. Objects that are further away have smaller differences in position. With this technique, the actual distance of objects from the cameras can be detected. There are two techniques for implementing stereo vision.



Two points as seen by two cameras



Point 2 is further from the cameras. It therefore has a lower disparity between camera views.

The first technique, dense stereo matching, attempts to match every pixel in the left camera view to a corresponding pixel in the right camera view. It then assigns every pixel a distance based on the disparity between its positions in the two camera views. This technique is computationally intensive, but produces more detailed distance maps.

The second technique is called sparse stereo matching. It works by finding feature points in one of the two images. In this context, features are areas that are visually distinctive. The algorithm then attempts to find these feature points in the other image. It then computes the distance to those feature points based on the disparity between their positions in the two camera views. This technique is less computationally intensive than dense stereo matching, but produces less complete distance maps.

With either technique, the output can then be analyzed to separate the ground from obstacles. This requires (for the technique used in this project) making the further assumption that the ground is flat. The first step of this process is finding the ground plane. The second step is calculating the distance from the ground plane to each point or area with a known distance. Then, anything with a height more than a set amount above the ground can be tagged as an obstacle. This assumption, that the ground is flat, can cause false negatives or positives when the ground dips or rises in front of the robot.

This technique also makes the assumption that obstacles will have enough surface detail to match them between camera views. This can cause it to produce false negatives in cases where obstacles do not have enough surface detail to be matched.[5]

Importance of Obstacle Avoidance

In scientific exploration and emergency rescue, there may be places that are dangerous for humans or even impossible for humans to reach directly, then we should use robots to help us. In those challenging environments, the robots need to gather information about their surroundings to avoid obstacles. For outer space exploring robots this is even more important because there can be a delay of seconds or minutes between the control station on earth and the robot.. Nowadays, even in ordinary environments, people also require that robots can detect and avoid obstacles. For example, an industrial robot in a factory is expected to avoid workers so that it won’t hurt them. In conclusion, obstacle avoidance is widely researched and applied in the world, and it is probable that most robots in the future should have obstacle avoidance function.

Sensors for Obstacle Avoidance

A variety of sensor types can be used to detect obstacles. Among them, cameras occupy a middle ground. They provide a large amount of information, as opposed to cheaper options such as infrared sensors which only provide the distance of objects directly in front of the sensor beam. However, they provide much less accuracy and detail than costly options such as LIDAR, which can directly measure the distance of thousands to hundreds of thousands of points in its field of view. USB cameras have become increasingly common in the last decade, making them an attractive option for obstacle detection applications.

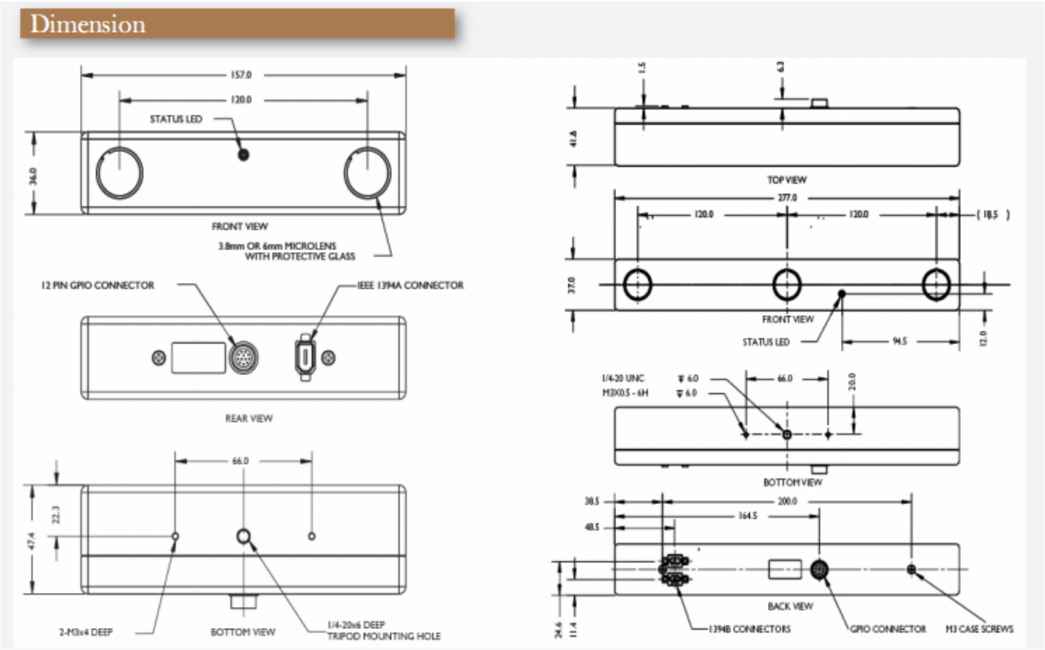
Table 2: A comparison of sensors for obstacle avoidance

Sensor Type	Cost	Information Gathered	Issues
Infrared	~20$	Distance of objects directly in front of sensor (thin beam width)	Interference between multiple IR sensors
Sonar	~30$	Distance of nearest object within viewing	Interference between multiple sonar sensors.

		angle	Errors depending on surface properties of object and angle of object to sensor
Camera (monocular) (after processing)	~40$ (depends on camera)	Areas of color thought to correspond to floor	Requires good lighting Assumes floor to be one color Assumes obstacles to be a different color from the floor
Camera (stereo) (after processing)	~80$ (depends on camera)	Distance of objects from cameras	Requires good lighting Requires objects with sufficient surface detail
LIDAR	~6000$	Distance of thousands of points in field of view with millimeter accuracy	High cost

Camera Selection

Existing products in the market[7]



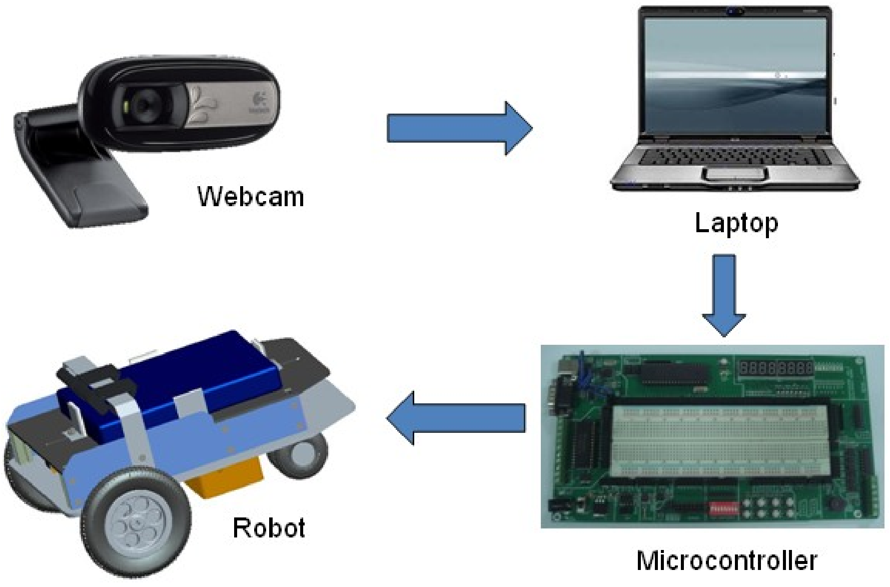
Two dimensional drawing of existing product[7]

We find in the market that many companies already have mature products for vision calibration. As an example, the cameras above have been assembled precisely to make sure both cameras point in exactly the same direction. Apart from that, most of them have integrated to control the cameras and process the images.

We considered taking advantage of these cameras directly. However, the high price prevents our application because we need to get a relatively high-value and low-cost design. And the ports are not usually USB which needs to adapt the communication methods. Actually, after a manual work to adjust the cameras, the precision of our work is equal to the obstacle avoidance. The program for calibration and image processing is the most challenging part that should be done by us.

Chapter 3: Methodology

The image below shows the information flow of our robot. The images taken by the cameras are sent to the laptop where the images are processed and decisions are made. The laptop then sends commands such as moving forward, turning left or right to the micro controller. The micro controller then sends commands to the motors to make the robot move forward, turn left or turn right.



Information flow of our robot

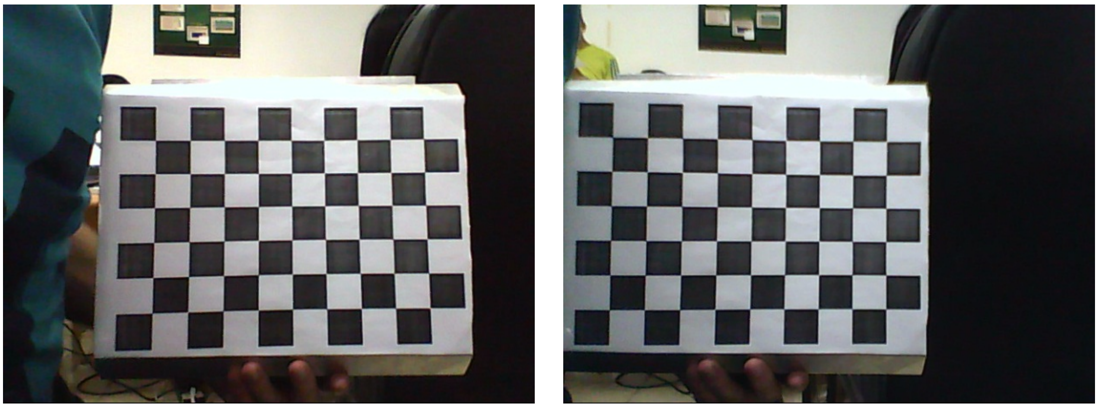
Computer Vision

The computer vision section of this project is required to analyze images using OpenCV and output the location of obstacles. It should also be usable by students and others with general technical backgrounds that do not have any experience with computer vision. Because of this, this section was written in Python, a widely used high level programming language that automatically handles tricky low-level tasks such as memory allocation. The benefits of Python for novice programmers are discussed more in detail in the background.

1. Collect Images: The first module has a simple purpose. It takes as input the IDs of the two cameras used and outputs frames from them. It also provides information about the images gathered, such as the size of the images.

2. Get Stereo Calibration: this module is used to collect a series of images of a calibration patterns, in this case a 9x6 chessboard. OpenCV supports chessboard patterns as calibration patterns, and provides several functions specifically for extracting points from images of chessboards and

working with those points. This module takes as input pairs of left and right images and saves them to disk for use in the Stereo Rectification module. This module only needs to be used when the position of the cameras relative to each other changes.

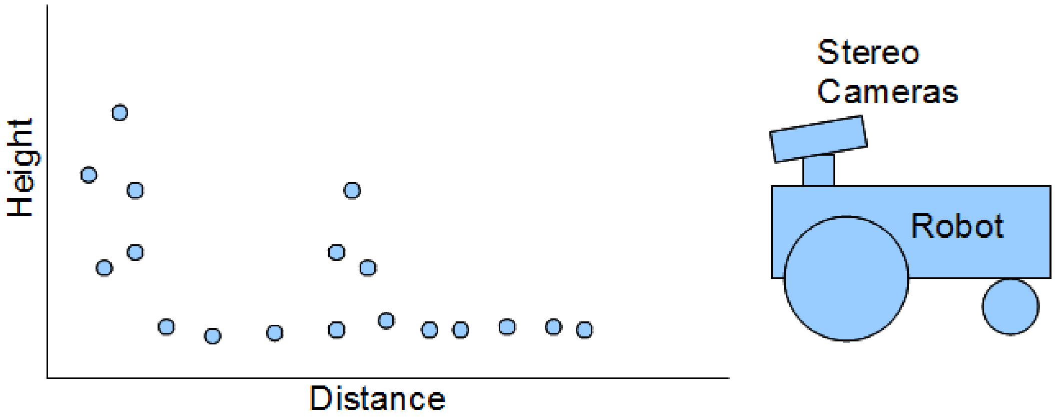


A pair images used for stereo calibration

3. Stereo Rectification: this module has two possible inputs. First, it takes as input a set of stereo calibration images, as produced by the Get Stereo Calibration module. From these it calculates the necessary transforms to stereo rectify a pair of images. This requires correcting for the intrinsic distortion of the cameras and aligning the images such that features in each image have the same height. OpenCV provides functions that handle the complicated math and matrix operations. After this is completed, it can take as input a pair of left and right images and applies the previously

calculated transforms to them. It then outputs a pair of stereo rectified images.

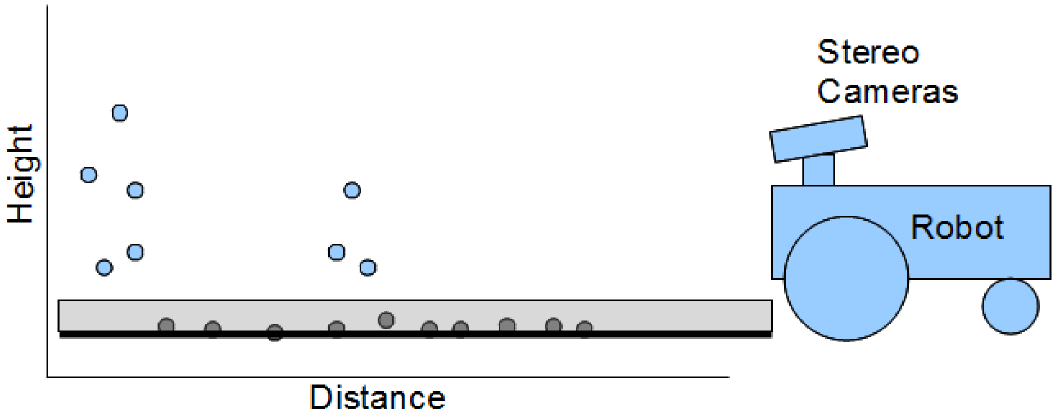
4. Sparse Stereo: This module takes as input a pair of stereo rectified left and right images. It then finds features in the left image and attempts to find the position of the corresponding features in the right image. It then uses the X and Y position of these points, as well their disparity, to map them into points in three-dimensional space. These points are the output of this module. OpenCV provides function for finding points and tracking them between images, specifically GoodFeaturesToTrack and CalcOpticalFlowPyrLK.



A side view of the points found in the previous step

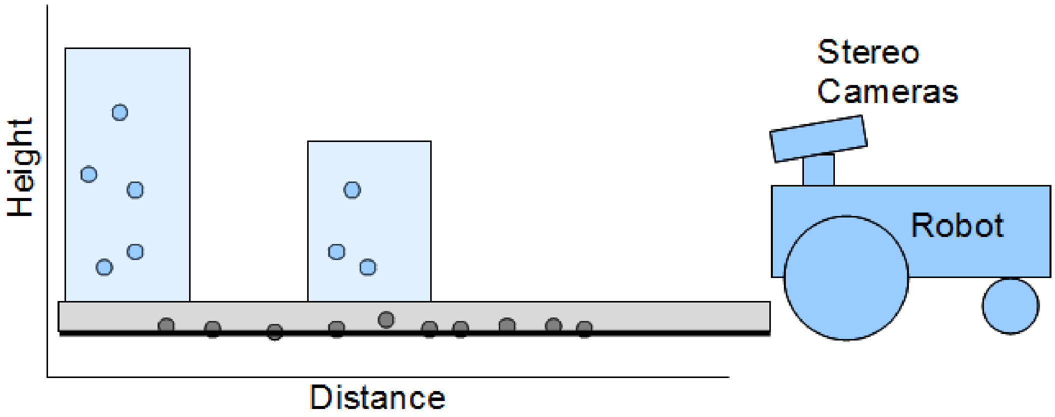
5. Find Plane: This module takes as input a pair of stereo rectified left and right images. It searches those two images for the same 9x6 chessboard pattern used in the Stereo Rectification module and matches points between the two images. It then uses the same technique as the Sparse Stereo module to map these points to three-dimensional space. Finally, it uses a linear least squares approach to fit a plane to those points. The object representing this plane is the output of this module. This module only needs to be used when the position of the cameras relative to the ground changes.

6. Find Obstacle Points: This module takes as input the set of three-dimensional points output by the Sparse Stereo module and the ground plane outputted by the Find Plane module. It compares those points to the plane and outputs all those points that are more than a preset distance above the ground plane as obstacle points.



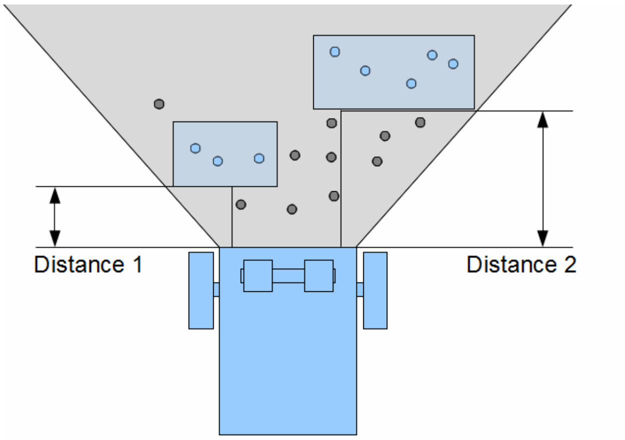
The black line is the ground plane; the grey rectangle is the area above the ground plane where points are still not classified as obstacles. Blue points are obstacle points.

7. Find Obstacle Bounds: This module takes as input the set of three-dimensional obstacle points output by the Find Obstacle Points module. It then attempts to group them into clusters, where each cluster corresponds to one obstacle, and the obstacle that cluster corresponds with is within the bounds of that cluster. This module then outputs the bounds of these clusters. Points are grouped into clusters using a hierarchical approach: each point starts as its own cluster, and the closest clusters are repeatedly merged.



The obstacle points are grouped into clusters

8. Serial Communication: This module takes as input the bounds of the obstacles found by Find Obstacle Bounds. It then interprets this data and sends commands to the microcontroller that tell it to turn left, turn right, stop, or go forward. These commands are then sent to the robot using a USB-to-Serial adapter.



Because the distance to each point is known, the distance to each cluster can be calculated

Control system design process

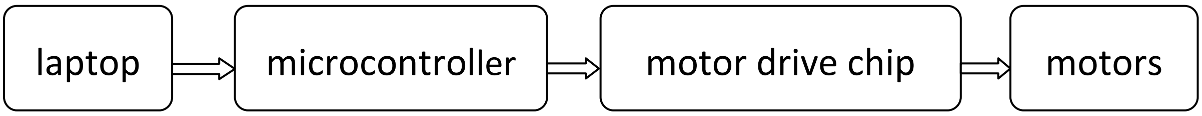
Requirements for control system:

(1) We can control the speed and direction of the motors by change the pulse width

(2) The microcontroller turns the signals into PWM wave when receive the laptop input and control the movement of the robot.

Principle of control system:

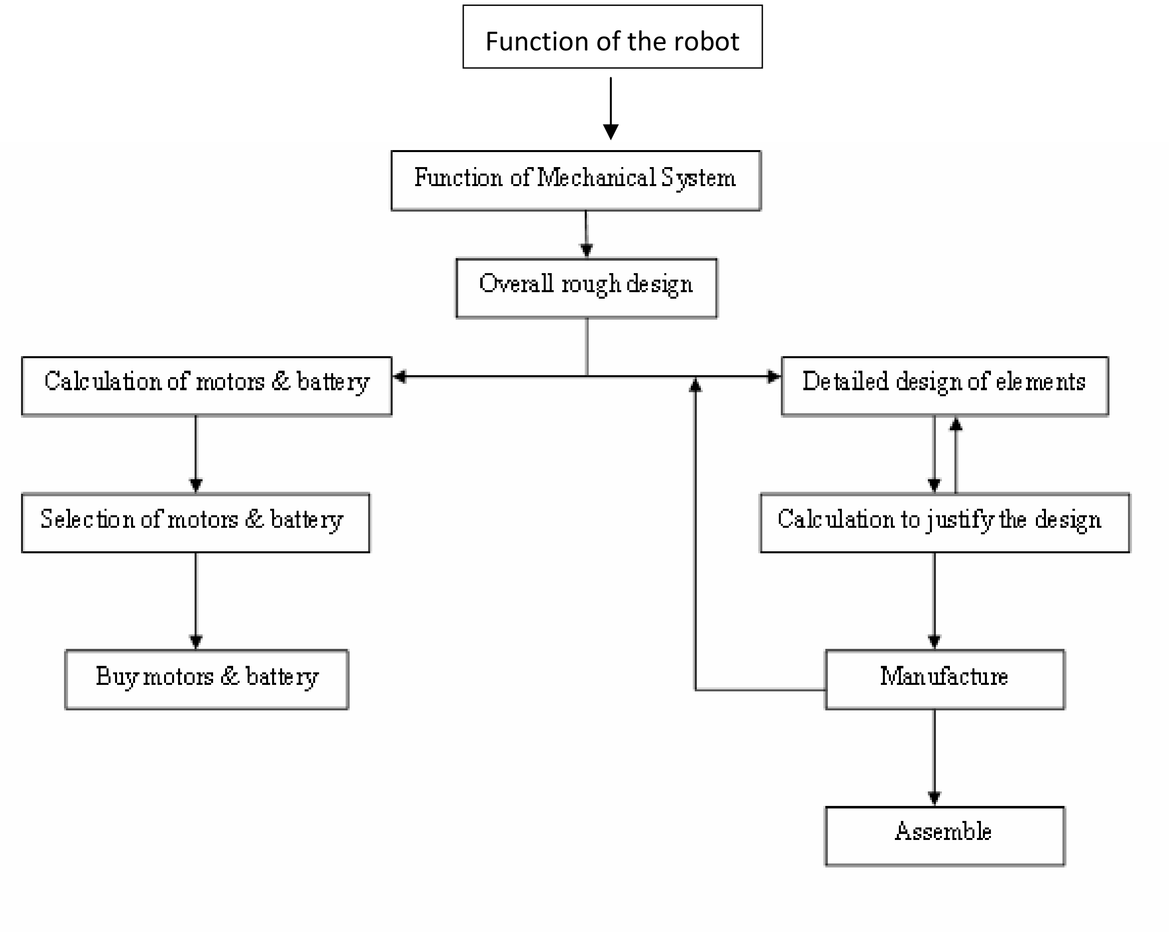
Laptop sends motion commands to microcontroller, and then the microcontroller sends out a PWM pulsecorresponding to the motion commands via p1.1 which will drive the motors by the HB-25,then the robot will realize the corresponding motion.The following flow chart shows the diagram of hardware design:



Systematic block diagram of hardware

Mechanical Design Process

Figure 1: Mechanical design process


Function of the robot
Function of Mechanical System
Overall rough design
Calculation of motors & battery
Detailed design of elements
Selection of motors & battery
Calculation to justify the design
Buy motors & battery
Manufacture
Assemble

From detailed design of elements to assemble, the detailed procedures should be like this:

Overall, the mechanical design has two parts

- Laptop holder
- Camera holder

Each part will be executed in this order:

1. Talk with Paul and learn about his requirements for mechanical part
    - We need to document his requirement and his justification for the requirement
2. Try all means to design suitable mechanism
    - Search information online and from text books
    - Think by ourselves
3. Draw rough design on a piece of paper and explain that to Paul to see if it fits his requirements well enough. If not, re-design
4. Do a rough feasibility analysis on the design
5. Draw three dimensional models on Pro/e. During this process, some elements need to be bought online, then we should search for enough information about those elements such as the dimensions, prices and important parameters
6. Do detailed analysis to our design
    - First think of what analysis is necessary and list them in a word document
    - Then think of the solutions to each item of analysis
    - Seek for help if any of the analysis is beyond us

7. Draw two dimensional designs for each element.
8. Send two and three dimensional designs to Depush and get feedbacks from Depush
9. Do some modifications according to Depush
10. Take two dimensional designs to machine shop for manufacturing
    - Maybe we still need to modify our designs so that it is easier to manufacture
    - Depush may help manufacture some elements
    - Other elements may be bought from the market
11. Stay in contact with machine shops and urge them to manufacture as quickly as possible
12. Assemble the elements and modify when necessary
13. Work together with Control group and Stereo vision group to make the whole robot perform well

Motor Calculations

About the calculation of motors: the parameters we need to know about the motors are minimum torque, maximum revolving speed, voltage and power.

1. Minimum torque $T_{\min}$: the minimum torque should be able to rotate the wheels when the robot is on the ground. The weight of the robot G_0 can be estimated, and the friction coefficient between the wheels and the ground μ can be known through table of friction coefficient. The maximum speed $v_{\max}$ and maximum acceleration $a_{\max}$ can be certified by the experiment that the images of the cameras won't blur and laptop has enough time to deal with each image. According to Newton's Second Law $2\times\frac{T_{\min}}{r}-\mu\times G_0=\frac{G_0}{g}\times a_0$, r is the radius of the wheel
2. Maximum revolving speed: The maximum speed of the robot $v_{\max}$ can be certified by the experiment that the images of the cameras won't blur. Then the maximum revolving speed of the motors is $n=\frac{30\cdot v_{\max}}{\pi\cdot r}$.
3. Voltage: the voltage of motors is usually 12V or 24V. They are both acceptable
4. Power: the energy of motors equals the kinetic energy the robot gains within one second plus friction loss within one second.

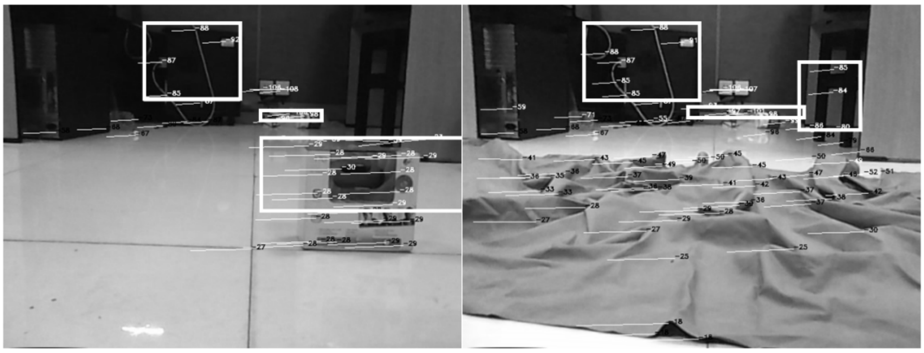
About the calculation of battery: the parameters we need to know about battery are voltage and current. The voltage and current of battery should be no bigger than the maximum voltage and current the motors can stand. Another important parameter of battery is that battery capacity as it determines how long the battery can work at a certain current.

Chapter 4 Findings and results

Obstacle Detection Evaluation

An obstacle detection algorithm needs to satisfy two criteria to be successful. First, it has to detect obstacles when they are present. Second, it has to avoid falsely labeling areas that are not obstacles as obstacles.

Our algorithm's performance under the first criteria was determined by showing it a set of obstacles with different characteristics at several different distances from the front of the robot.



Obstacles and free space. (White boxes shown surrounding obstacles)

Observed indoors, under fluorescent lighting:

1. Two different kinds of plastic bottle with wrapper: these obstacles are partially transparent

2. Computer case from the back, showing I/O ports and ventilation grating: a large, rectangular obstacle with a large amount of surface detail

a) Computer case from the side (One solid color): a large, rectangular obstacle with very little surface detail

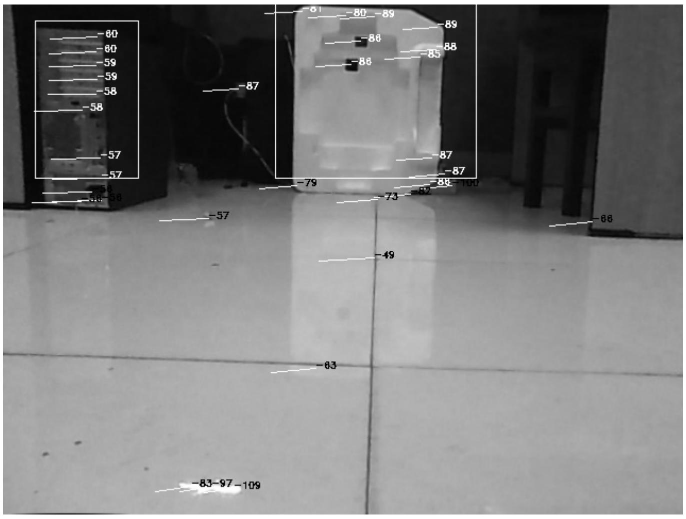
3. Styrofoam packing material: a large object of the same color as the floor

4. Hot water jug: a non-rectangular obstacle of one color with minimal surface detail

5. A chair: an obstacle with more complex geometry (thin legs)

6. Webcam packaging: a small cardboard box with medium surface detail

These obstacles were observed at distances of 2 feet, 4 feet, 6 feet and 8 feet from the front of the robot. The algorithm successfully detected all obstacles except for the solid-color side of the computer case. In that case, the algorithm could not find any features to track. The point clusters representing the boundaries of some of the smaller obstacles, such as the water bottles, merged with the point clouds of nearby obstacles when viewed at a distance of 8 feet.



Of particular note is the algorithm’s success in detecting the Styrofoam packing material. Its color (white) was only slightly different from that of the floor in the lab (off-white).

The algorithm’s performance under the second criteria was determined by showing it a set of non-obstacles that could be falsely labeled as obstacles, again at several different distances. Some examples of these non-obstacles are a piece of paper lying flat on the ground and a reflective floor.

Observed indoors, under fluorescent lighting:

1. An empty shopping bag

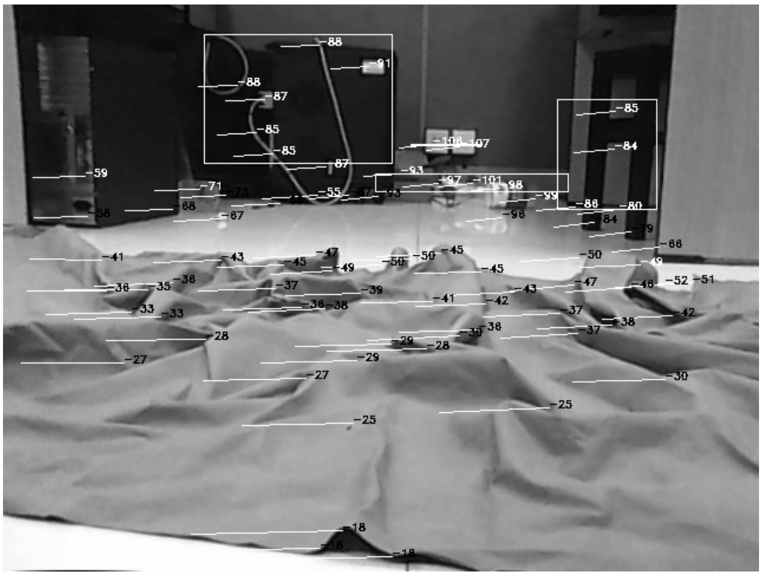
2. An empty shopping bag made of thick paper

3. A large sheet of cloth

4. A piece of paper with dark black lines drawn on it

5. An area of floor in which the reflections of objects where visible

The non-obstacles were viewed in various positions between 1 and 6 feet from the robot, at various positions in the left, right and center of the robots view. The algorithm correctly identified all of these non-obstacles, using the same settings that were used to detect the obstacles in the previous tests.



Of particular note is the algorithm's success in recognizing the large cloth sheet as part of the floor, even when it was bunched up so that parts of it formed small ridges.

The speed of the algorithm was also analyzed. Mobile robots require real-time detection of obstacles, which in turn requires algorithms that run at reasonable speeds. First, the speed at which our algorithm runs is bounded by the available processor resources. The computer we used to control the robot for our tests had a 1.66 Gigahertz processor and 2.0 Gigabytes of RAM. With these resources, the algorithm runs in approximately 0.3 seconds. It takes less time when presented with a scene with a small amount of features.

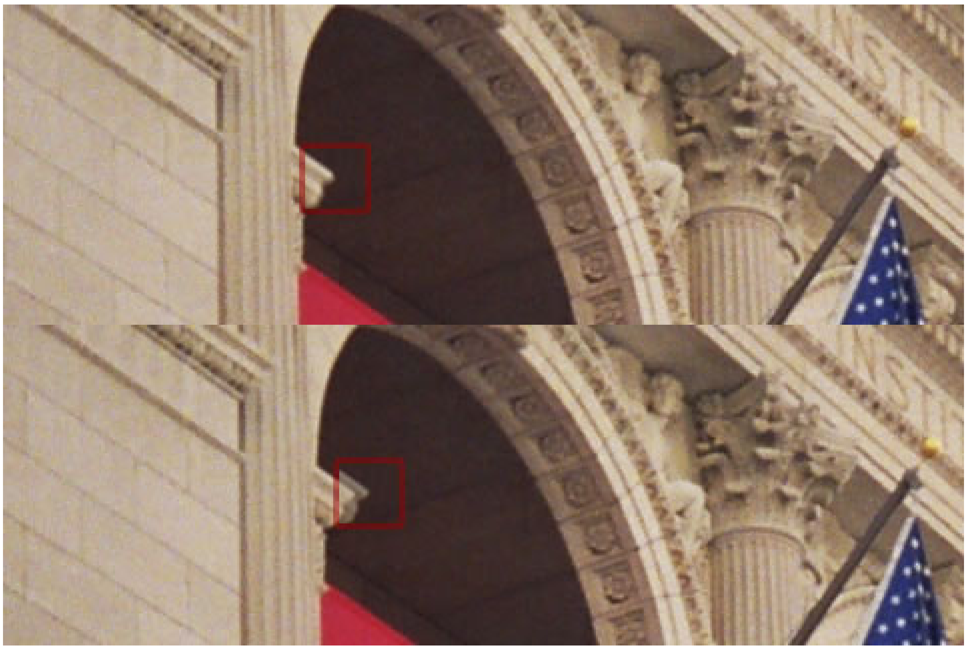
Testing with the robot has shown this to be sufficient to detect and avoid obstacles.

Algorithms Used

Two algorithms do the majority of the work in the computer vision portion of this project. First, it uses the Shi-Tomasi corner detection algorithm to find features in one of the two images. Second, it uses the sparse version of the

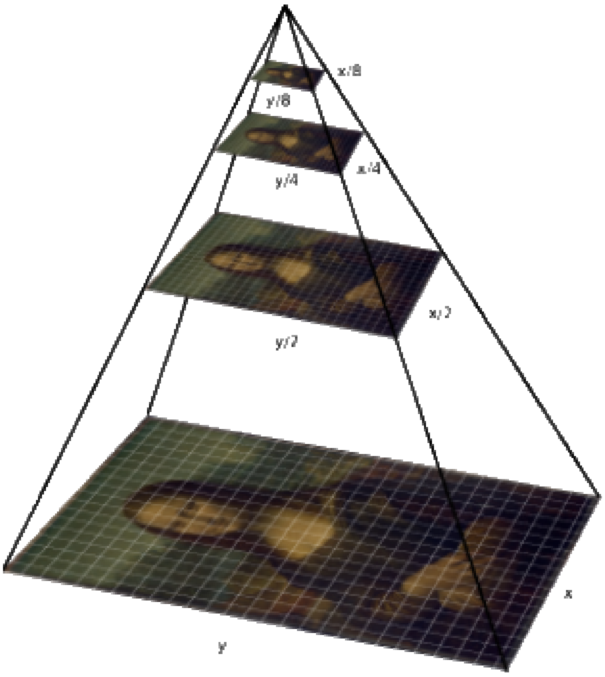
Lucas-Kanade optical flow algorithm to find the location of these features in the second image.

The Shi-Tomasi corner detector works by finding areas within an image that, if moved, produce a large variation. It's called a corner detector because these areas are often corners. [6]



An example of a “good feature to track”

The Lucas-Kanade algorithm takes as input two images and features found in the first of these images via the previously mentioned algorithm. It then attempts to match these features (technically the area around the location of these features in the first image) with the corresponding area in the second images. It is a pyramidal implementation of the algorithm, meaning that it runs repeatedly on the input images at different resolutions. For example, an image pyramid with n levels would have at the top an image of size (width = original width/(2^n), height = original height/(2^n)). The image below that one in the pyramid would have size (width = original width/(2^(n-1)), height = original height/(2^(n-1)), and so on.[7]



An image pyramid with three levels

The settings used by both of these algorithms were chosen with speed in mind. The program as a whole needed to run at a reasonable. The definition of reasonable speed depends on the speed the robot moves – three cycles per second was chosen based on experimentation. Window sizes and minimum allowable errors were chosen that would allow the algorithm to run at this speed without failing to correctly find and match features.

Control Section Results

1. The design of the hardware system

1.1 Introduction

We take the STC89C58RD+ microcontroller as the core of the system design. After the input of the laptop, the microcontroller can send signals to control the motors' states which determine the movement of the robot. In this design, the PWM modulation technology is adopted for the motor controlling, which adjust the speed precisely by calculating the duty ratio.

1.2Control Principle

The DC motor PWM speed regulation system is constructed by single chip, and the core part consists of minimum system, power source module (12V 5V), motor driving chip and DC motors. We use HB-25 motor controller as the main loop of PWM pulse to drive DC motors. Microcontroller outputs PWM signals, realizing the speed control of DC motors and achieving good both static and dynamic performance. As this system has high property and price ration and simple structure, it has high application value and deserves to be widely promoted.

Stereo vision gets environment information which is then sent to laptop to proceed. The OpenCV program in the laptop can detect whether there are obstacles around and the position of the obstacles. According to particular strategies of avoiding obstacles, the laptop makes the decision about how the robot moves, and then sends commands to microcontroller through serial communication. Figure 1-1 shows graph of the DC motors control system with STC89C58RD+ microcontroller as the core. Microcontroller receives commands and then changes the commands into PWM signals and sends the PWM signals out by P1.0, through HB-25 motor controller into 2 DC motors, thus control the motors to get or lose power, realizing the corresponding movements of the robot.

1.3Motor Speed Regulation Module

1.3.1 Solution Choice

Solution 1: use resistance network or digital potentiometer to adjust the partial voltage of motors to achieve the goal of regulating speed. However, resistance network can only realize step speed regulation and digital resistance element is relatively expensive. What’s more important, common motors have very small resistance but very big current; partial voltage will not only lower efficiency but also be hard to realize.

Solution 2: use electric relay to control motors to be open or close, and regulate the speed of motors by switchover. The advantage of this solution is that it has simple circuits and the disadvantage is that electric relay responds slowly, has relatively short span, are not highly reliable and its mechanical structures are easy to break.

Solution 3: use HB-25 motor controller to drive DC motors, as the peripheral circuits of HB-25 produced by Parallax are simple, and H bridge circuits have good drive capability and the HB-25 itself has heat dissipation equipment. So we use Solution 3.

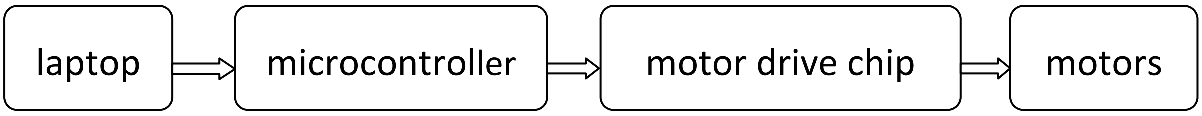
As unipolar duty cycles have lower percentages of AC in their voltage wave than bipolar duty, and unipolar duty's maximum current fluctuation is smaller than that of bipolar duty, thus we choose unipolar duty.

1.3.2 Approaches to Regulate Frequency Range of PWM

There are three ways to regulate frequency range: fix frequency and regulate range, fix range and adjust frequency and regulate both frequency and range. We choose the approach of fixing frequency and regulating range because the motors run relatively smoothly by this approach; and this approach makes it convenient to produce the PWM pulse with programming on microcontrollers.

1.4Systematic Analysis and Hardware Design

Laptop sends motion commands to microcontroller, and then the microcontroller sends out a PWM pulsecorresponding to the motion commands via p1.1 which will drive the motors by the HB-25,then the robot will realize the corresponding motion.The following flow chart shows the diagram of hardware design:



Systematic block diagram of hardware

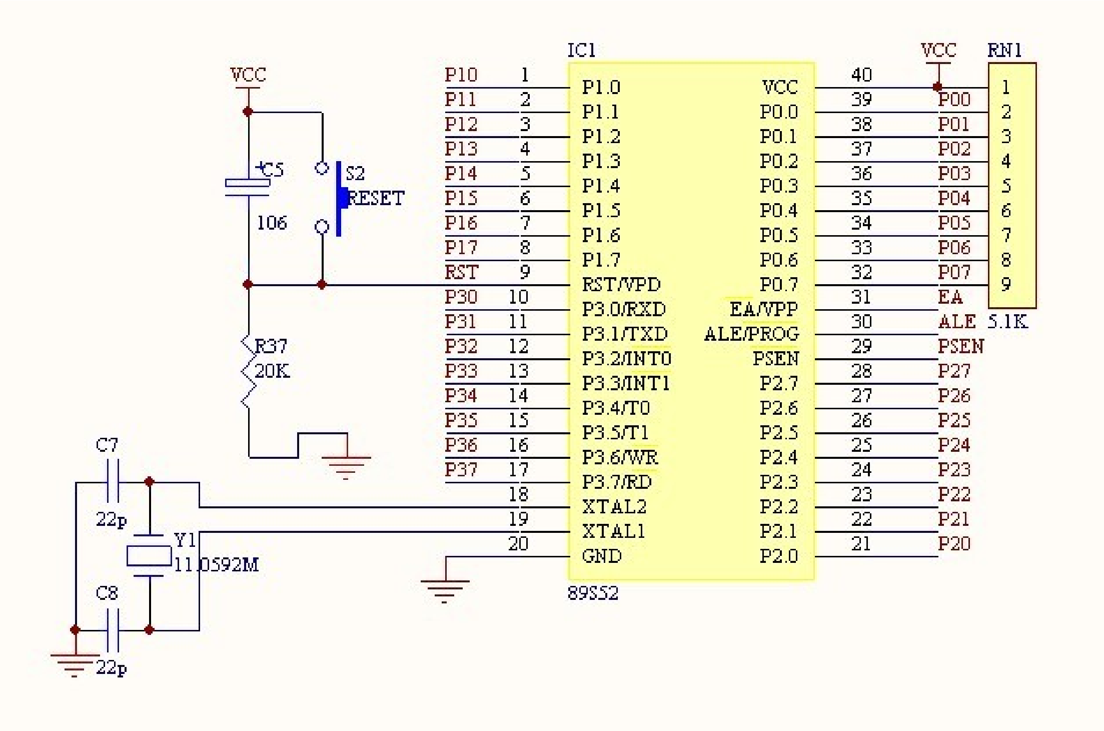
1.4.1 Design of Microcontroller's Minimum System

The microcontroller receives commands of movement from laptop, then change the commands into PWM pulse to control the revolving of two motors. To realize the above functions, we just need 3 I/O ports, 3Kb storage for programming, 1 timer and 1 serial cable.Therefore almost all microcontrollers can meet our requirements. With comprehensive consideration about price and company needs, we finally choose the STC89C58RD+.

The STC89C58RD+ 8 bits microcontroller is a strengthened type of 51 microcontroller developed by the American company STC based on standard core structure of 8051 microcontroller. This STC microcontroller has the following advantages:

- High encryption
- Super resistance to interference
- Three measures to decrease the radiation of microcontroller to the outside
    - ALE outputs are forbidden
    - If 6 clock or machine cycle is chosen, frequency of outside clock can decrease one half
    - The transmission gain of microcontroller’s clock oscillator can be set to 1/2 gain.
- Super low power consumption
- Can be programmed right in systems, programmer is not needed, can be remote upgraded
- Can be sent to STC-ISP, 10 thousand chips per person per day

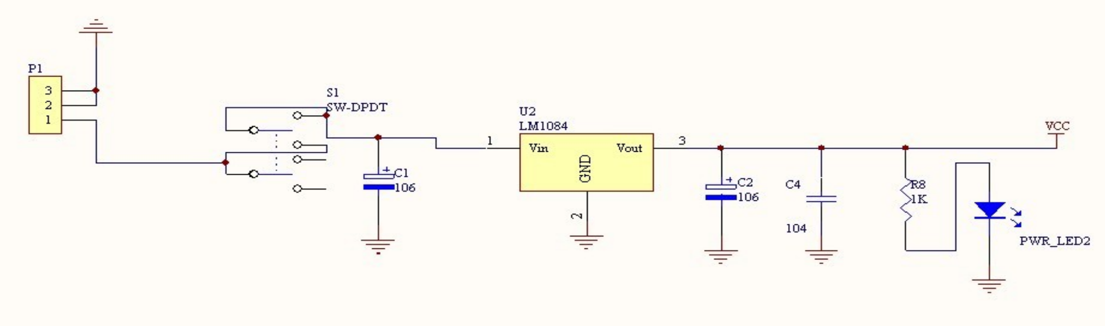
According to different requirements in different cases, this type of microcontroller provides different kinds of packaging. Actually minimum system sometimes needs to change microcontroller and thus we use the DIP-40 packaging, as is shown in the following figure.



Minimum system

In addition, it sometimes occurs that programming flies while the system is running, then people have to reset by hand. Therefore in this design we reset by hand.

1.4.2 Power Source Circuits Design

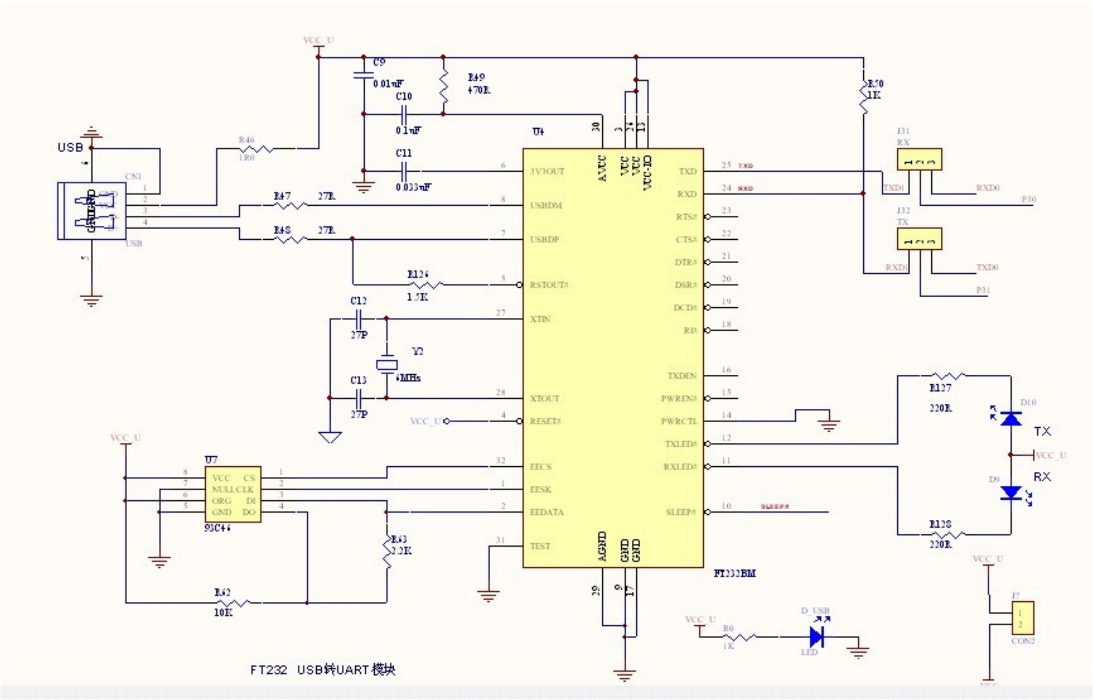


Power circuits

This power source circuits use LM1084-5 low voltage gap, linear regulated voltage integrated circuits. The input voltage is 6-9V, and the gap between input voltage and output voltage is lowered to 1.5V, and the output current is 5A. LM1084-5 specifically outputs 5V, operation temperature ranges from −40°C to 125°C, can limit current and protect itself when it is over heated, has simple hardware circuits, relatively fixed methods to use.

1.4.3 USB Serial Switch Circuits Design

Laptop communicates with microcontroller through serial communication. Usually laptop has no serial ports but only USB ports, so USB serial switch circuits are designed to realize the communication between laptop and microcontroller. The USB serial switch circuits are shown in the followingfigure.



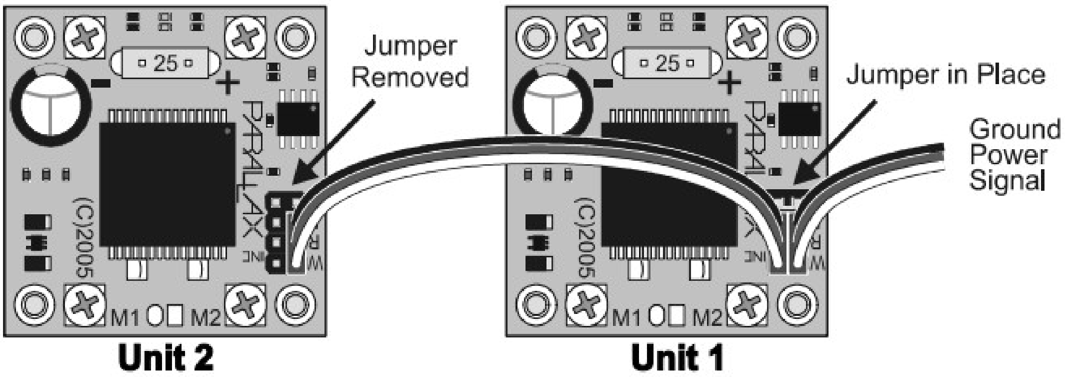
USB serial switch circuits

The FT232BM is a USB to serial UART interface. The USB to RS232 converter provided by FT232BM sets up a reliable connection between the USB port and the RS232 port. The “hot plug and play” of USB port makes peripheral equipment of RS232 easy to use.

1.4.3 DC Motor Drive Chip

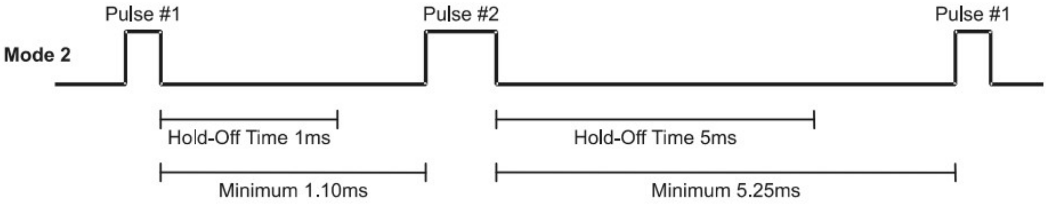
PARALLAX has produced DC motor drive chips——HB-25 that match the DC motors we have selected. The website of PARALLAX has provided the instruction of the HB-25. The below is just a simple introduction

Note that we need 2 HB-25 to drive 2 DC motors. We should wiring between the 2 HB-25 as shown in the following figure. So only then unit 1 will be connected to the single chip.



The connection between the 2 HB-25

Independentcontrol ofboth HB-25 units through a single I/O line is accomplished by sending two sequentialpulses with a pause of 1.1 ms between them. TheHB-25 looks for two sequential pulses and only responds to the second one. In reference to the diagram in the following FigureUnit 1 would respond to Pulse #1, and Unit 2 would respond to Pulse #2.Unit 2 looks for Pulse #1, pauses for a 1 ms hold-off time, then looks for Pulse #2. Therefore, the timing between Pulse #1 and Pulse #2 is important, and there should be a minimum of 1.1 ms between pulses. What is more, a single pulse is required to set motor speed.



Dual Mode Communication

2.Systematic Software Design

After powering on the single chip, the system enters the ready condition and the serial communication is initialized. Press the start button and the HB-25 is initialized. Then the single chip will wait for receiving the motion commands. If the single chip receives a motion command, the receive interrupt zone bit TI will be set to 1 and the single chip will switch to execute the serial interrupt program. In the serial interrupt program, the single chip send out a PWM pulse corresponding to the motion commands via p1.1 which will drive the motors by the HB-25,then the robot will realize the corresponding motion.

2.1 Strategy for Obstacle Avoidance

As shown in figuresA and B,there is only an obstacle in front of the robot. In figure A,if the obstacle is in the left side,the robot will keep turning right until there are no obstacles in front. Then the robot will go forward. In figure B,if the obstacle is in the right side,the robot will keep turning left until there are no obstacles in front. Then the robot will go forward.As shown in figure

C,there are 2 obstacles in front of the robot. The robot first detects the obstacle 1 and turns right for a certain angle, then it will detect if there are obstacles in front again. If no, the robot will go forward. If yes, no matter the obstacle is in the left side or in the right side, the robot will continue turning right until there are no obstacles in front. Then the robot will go forward.

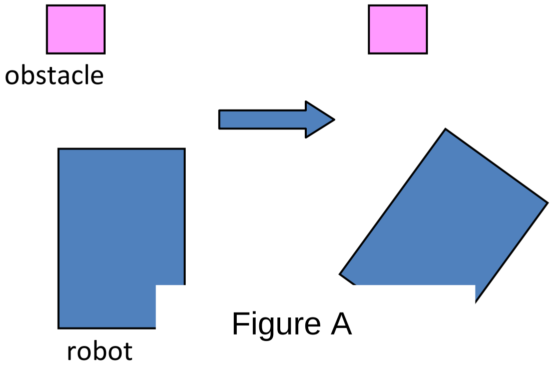


Figure A

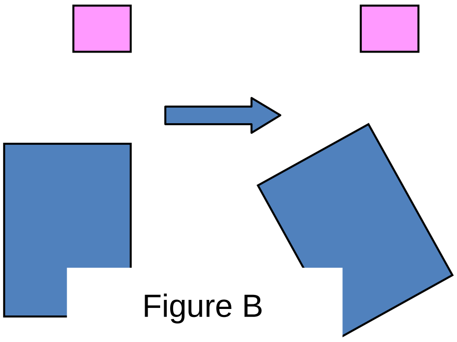


Figure B

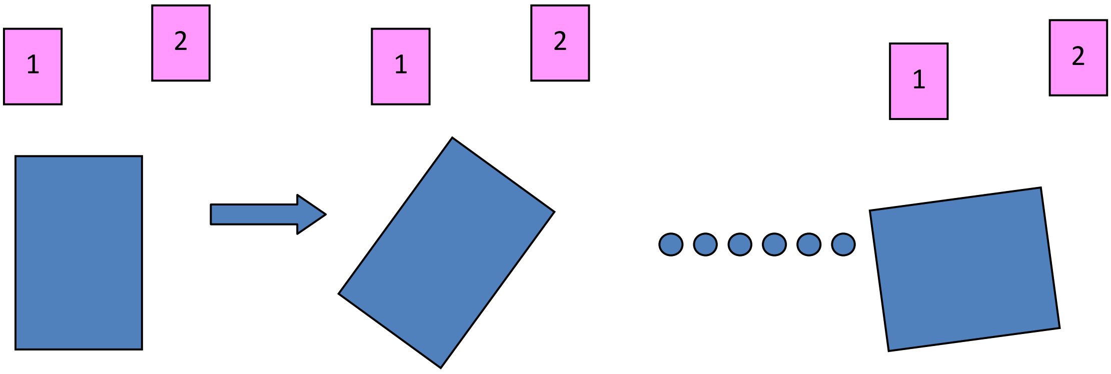
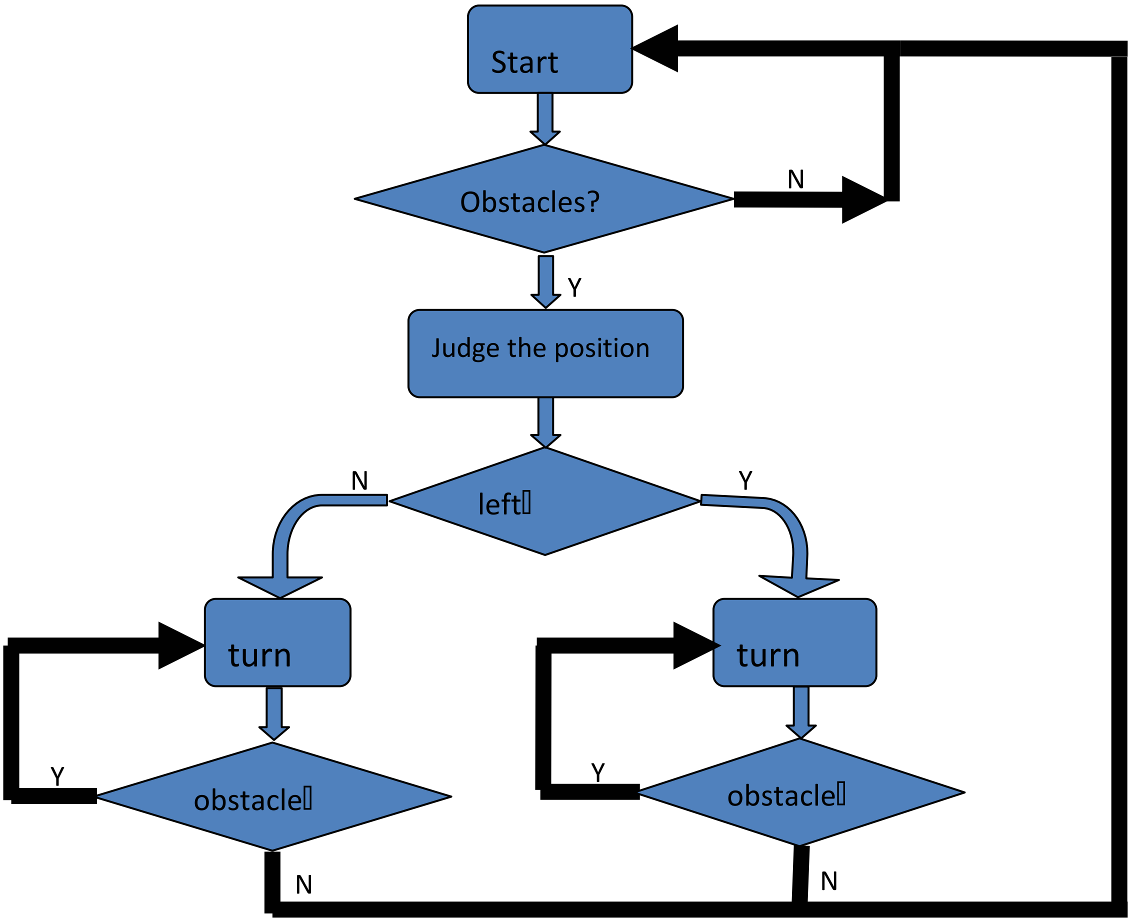
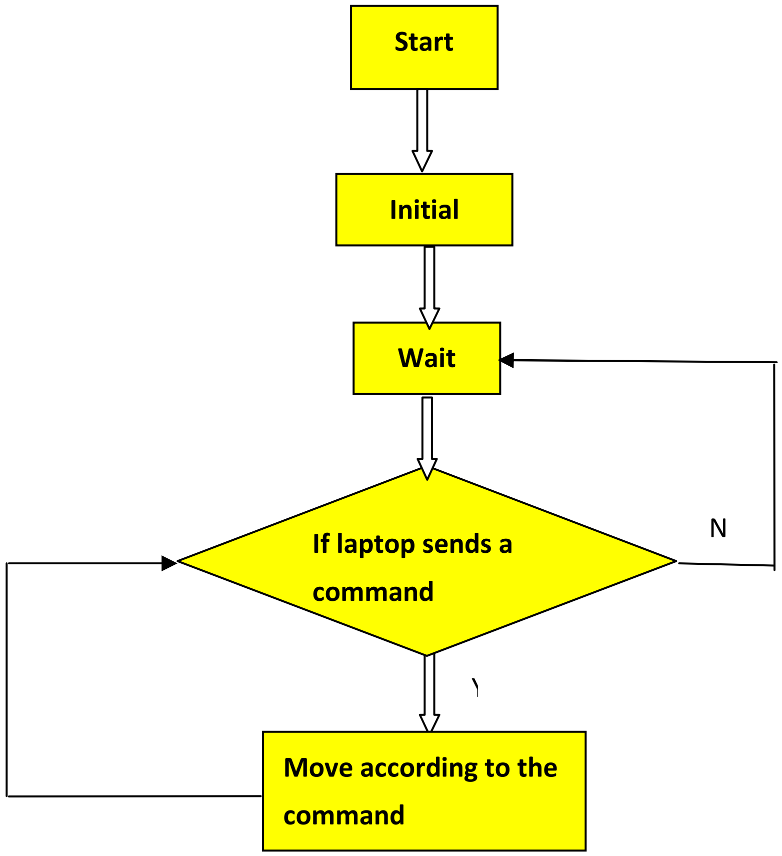


Figure C

The above is the basic obstacle avoidance rule of the robot. If the robot is under other conditions, it will avoid the obstacles according to the rule. The flow chart of obstacle avoidance is shown in the following figure.



2.2Flow Chart of Main Program



Source Program：

See appendix for source code

Mechanical Section Results

Selection of motors & battery:

- Minimum torque $T_{\min}$: $2\times\frac{T_{\min}}{r}-\mu\times G_0=\frac{G_0}{g}\times a_0$, $m_0=8kg$, so

  $G_0=8\times 9.8=88.4N$; $a_0=1m/s^2$, $r=0.075m$, all the above three parameters are estimated; $\mu=0.2$, so according to the equation,

  $T_{\min}=963N\cdot mm$

- The estimated maximum speed that can keep images from blurring is $v_{\max}=0.5m/s$, so the maximum revolving speed is $n=\frac{30\cdot v_{\max}}{\pi\cdot r}=\frac{30\times 0.5}{3.14\times 0.075}=63.66r/s$. Actually the maximum revolving speed of the motors can be higher, but in order to let the robot work well, the revolving speed should be no bigger than $63.66r/s$.

- The kinetic energy change is $\Delta k=\frac{1}{2}mv_0{}^2-0=\frac{1}{2}\times 8\times 1\times 1=4W$, v_0 is the speed the robot gains within 1 second at an acceleration of $1m/s^2$. The friction loss is $Q=G_0\cdot\mu\cdot S=88.4\times 0.2\times 0.5=8.84W$, suppose the efficiency $\eta=0.8$, then the power of one motor is $P=\frac{\Delta k+Q}{2\eta}=8.025W$

- The voltage of batter depends on the voltage of motors, and when the motors are selected, the battery’s current should correspond to the current of motors. It is best that our batter can work successively for more than two hours.

Laptop holder results

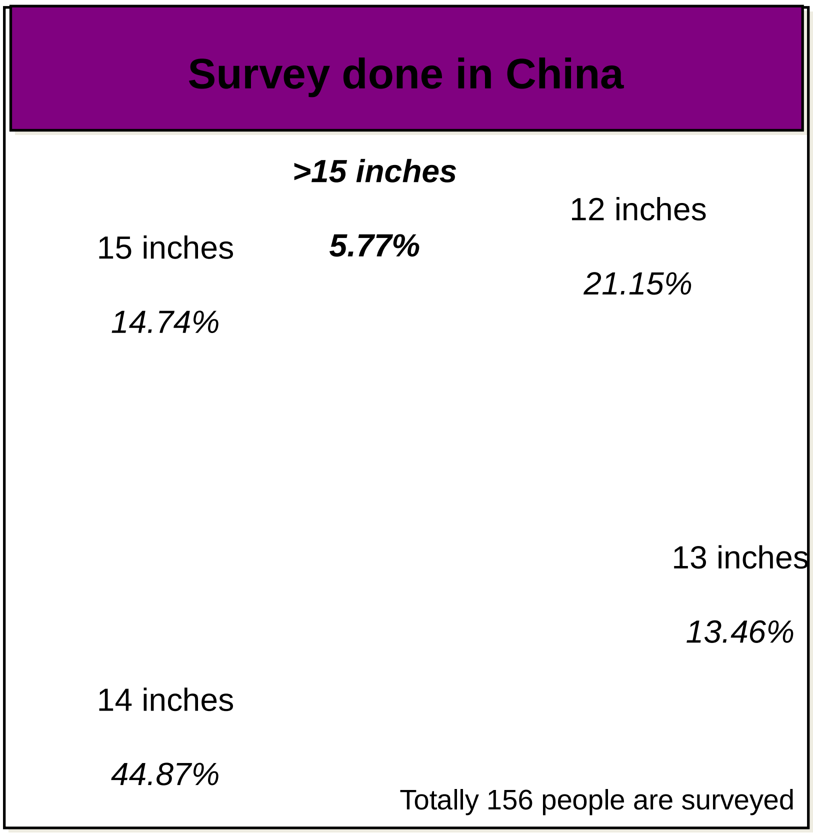
Function requirements:

- hold laptops of different sizes (different users may have laptops of different sizes)
- shouldn’t harm the laptops

Here are the results of my design process

1. after we think the design works, I ensure the exact dimensions of the laptop holder

Figure 2: Popularity of different laptop sizes

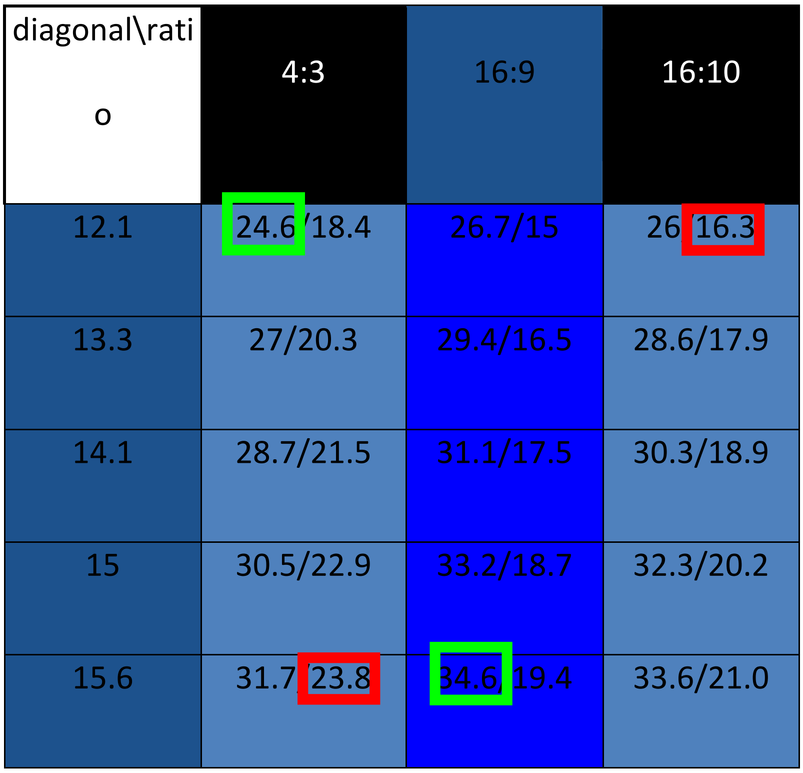


One survey about Chinese people’s preferences to the sizes of laptop computers [6]

2. The above survey shows that only less than 6% people prefer laptops that are bigger than 15 inches. Therefore it is reasonable that we only allow our laptop holder to hold laptops smaller than 15.6 inches. Depending on

different ratios of length and width, the same diagonal dimension may have different sizes of laptops, and the following is the table in which I calculate the length and width of laptops.

Figure 3: Sizes of laptops by ratio and diagonal length



diagonal\ratio	4:3	16:9	16:10
12.1	24.6/18.4	26.7/15	26/16.3
13.3	27/20.3	29.4/16.5	28.6/17.9
14.1	28.7/21.5	31.1/17.5	30.3/18.9
15	30.5/22.9	33.2/18.7	32.3/20.2
15.6	31.7/23.8	34.6/19.4	33.6/21.0

Diagonal Units: Inches

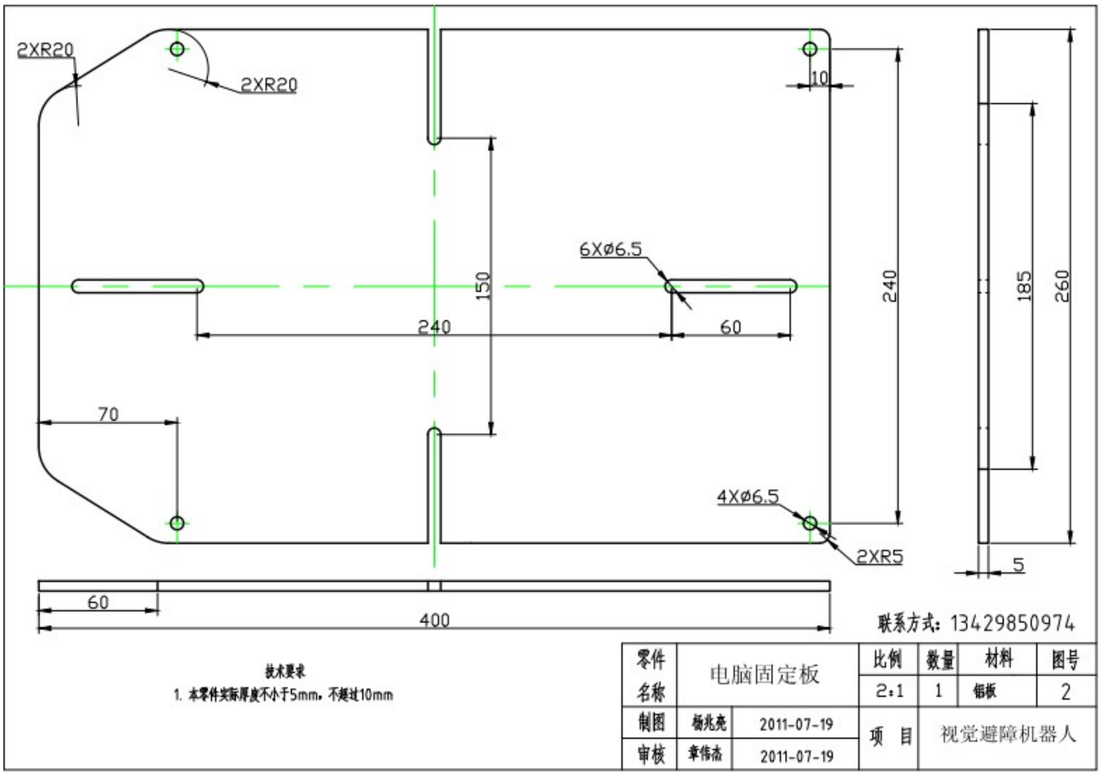
Length/Width Units: Centimeters

Matrix about width and length of different sizes of laptop computers at different ratios

3. So the length of grooves in width is 5cm, ranging from 7.5cm to 12.5cm from the center; the length of grooves in length is 6cm, ranging from 12cm to 18cm; In addition, as our time is limited, we have to use the original

chassis of Depush with motors. Thus the width of laptop holder is determined by the chassis, namely 260mm. The grooves are just 5mm from the edge of laptop holder, so I make the grooves all to the edges, meaning the grooves range from 7.5cm to 13cm from the center. The width of the grooves is 6.5mm because the holes of the chassis are 6mm and a wider groove width makes it easier from screws to get in. The length of laptop holder is 400mm, leaving some distance from the grooves to the edges.

4. According the above dimensional determination, we have the following laptop holder design:



联系方式：13429850974

技术要求
1. 本零件实际厚度不小于5mm，不超过10mm

零件名称	电脑固定板		比例	数量	材料	图号
			2:1	1	铝板	2
制图	杨兆亮	2011-07-19	项目	视觉避障机器人		
审核	章伟杰	2011-07-19				

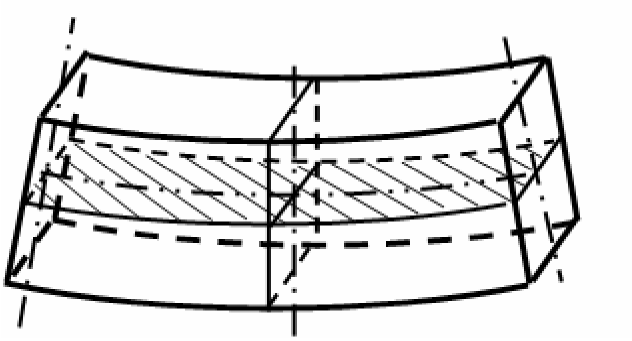
Two dimensional drawing of laptop holder [1][2][3]

In addition, the chamfers in the left side are made just to fit the chassis because the chassis has such chamfers too.

5. some calculations to justify that the laptop holder is strong enough to hold laptops:

In material dynamics classes, we learn to need to check bending normal stress & bending tangential stress to make sure it is strong enough. By the way, the thickness of the laptop holder is 5mm.

Equations used to calculate bending normal stress:



Schematic diagram of bent beam. [4]

$$\sigma = \frac{My}{I_Z} \quad [4]$$

$$\sigma_{\max} = \left(\frac{M}{W_z}\right)_{\max} \leq [\sigma] \quad [4]$$

$$[\sigma] = 140MPa[4]$$

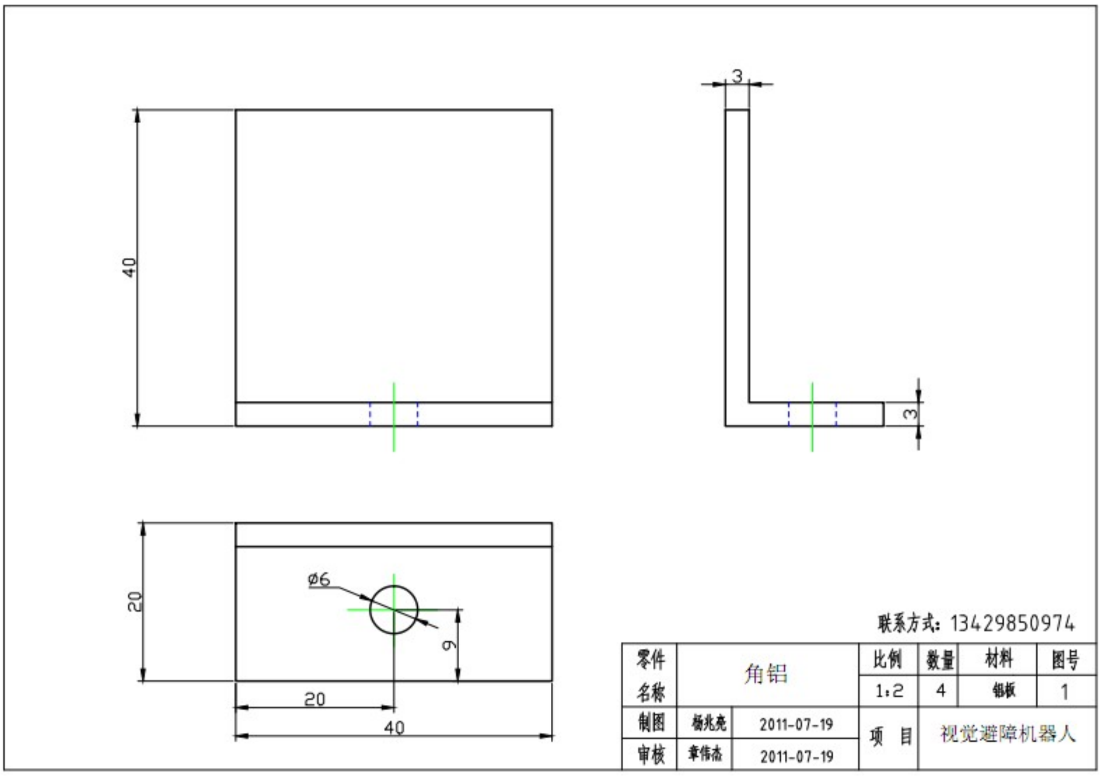
Equation for bent beam. [4]

Actually, detailed and exact calculation is not only difficult but also not necessary. So we use order of Magnitude Estimation to estimate.

- The magnitude of M is 10^{-3}
- The magnitude of W_z is 10^{-9}
- According to the definition of $M$and W_z and the weight of a laptop is usually less than 25.4N. It is quite reasonable that$\sigma_{\max} \leq 140 \cdot [\sigma]$.

It is the same with bending tangential stress that it is also within an allowed amount. In conclusion, theoretically the laptop holder is strong enough to hold the laptops.

6. Angle aluminum is used to be fixed onto the laptop holder to prevent laptops moving around. And chemical foam is glued to angle aluminum to protect the laptop. The chemical foam is made by us. The height of the aluminum is 40mm which is the thickness of most laptops. The hole of angle aluminum is also 6.5mm. the two dimensional drawing of angle aluminum is as follows:



Two dimensional drawing of aluminum alloy [1][2]

7. Then we should calculate to justify the friction provided by angle aluminum and laptop holder is enough to fix the laptop at a certain acceleration of the robot:

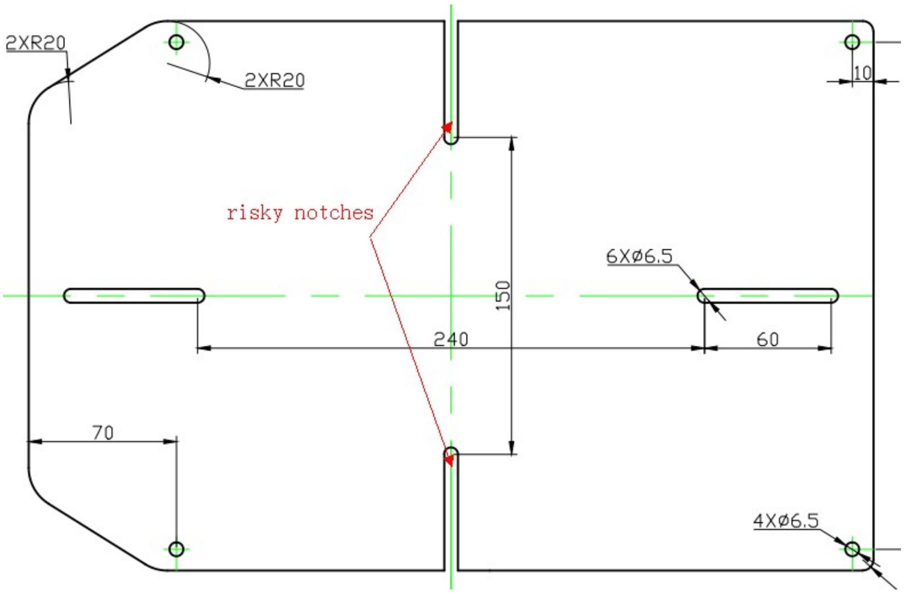
The motors are also provided by Depush and the parameters are known through the instructions provided by the supplier. The revolving speed of the motors is 150r/m, @12V, 1.5A, no load; the diameter of the wheels is 154mm, so the maximum speed of robot is 1.2m/s. The maximum acceleration of the robot is estimated to be 1m/s^2 and the friction coefficient between aluminum and aluminum is 1.05 to 1.35, or just use the average $\mu = 1.2$. Usually the mass of laptops is about 2.5kg, when acceleration $a_{\max} = 1 m/s^2$. The friction should be $f = m \cdot a_{\max} = 2.5 \times 1 = 2.5N$. Thus the tension of screws or the pressure of laptop holder is $F = \frac{f}{\mu} = \frac{2.5}{1.2} = 2.08N$. The diameter of screws here is 6mm. Thus the normal stress of screws is $\sigma = \frac{4 \times 1.3 \cdot F}{\pi \cdot d^2} = 0.0957 Mpa \le [\sigma]$. Obviously, screws can stand such little normal stress, or in other words, angle aluminum can provide enough friction to prevent the laptop from moving around.

8. About the material of the of the laptop holder, two kinds of material are considered: one is aluminum alloy and the other organic glass. The following matrix shows how I make the decision to use aluminum alloy:

Aspect & weight\material		**Aluminum Alloy**	**Organic glass**
Cost	0.1	4	6
Machinability	0.1	8	6
Strength	0.5	8	0
Density	0.3	3	7
Total		6.1	3.3

Table 6: comparison of laptop holder materials [3]

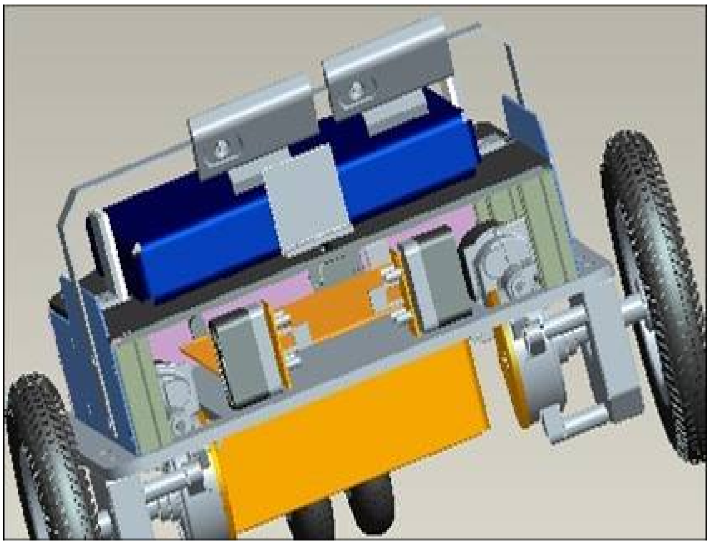
In the above matrix analysis, I consider four aspects, namely cost, machinability, strength and density. Strength is the key factor that determines which material we will use, thus I give strength weighting factor of 0.5. Organic glass's density is only the 43% of that of aluminum alloy, and less density is good for the robot, so density here is also considered and it has a weighting factor of 0.3. As a mechanical element, machinability should also be considered. In the end, cost should never be ignored. The scale is from 0 to 10 and more marks mean better performance. You may notice that organic glass only get 0 in strength and that is due to the fact that organic glass is easy to break along notches, which is called notch sensitivity. The following image shows the risky notches of our laptop holder:



Risky notches of laptop holder [2] [3]

Obviously, aluminum alloy get 6.1 marks while organic glass get 3.3 marks, so we choose aluminum alloy.

Results of camera holder



Blue view of camera holder on the robot

Function requirements

As a component to detect the obstacles and the ground, we have to fasten two cameras which can imitate human eyes on the robot car. To solve this problem, we need to consider the basic mechanical requirements and its compatibility with the computer vision system and control system.

As for the mechanical properties, we measure the weight of the cameras and discover that it is light enough to be attached to aluminum panel without much vibration. A detailed analysis of the stability will be presented later.

As for the integration function with the other two systems, there is not much relationship with the control system. However, how and where the cameras are placed will have great influence on the detection of the obstacles and the stereo vision system. To find out a right place, we need a through calculation of the height and the distance of two cameras.

In all, the requirements are as follows,

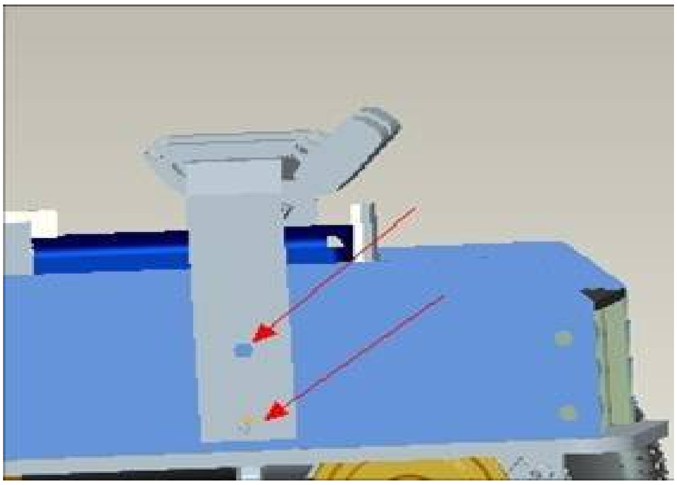
1.Stable

2.Cameras are at same height

3.Largefield of View

Stability analysis

In the process of movement of the car, we demand that the cameras should be fixed well with no vibration. For this reason, we tested the maximum acceleration of the car and calculate the possible force that could move the camera holder and the cameras respectively.



How camera holder is fixed onto chassis

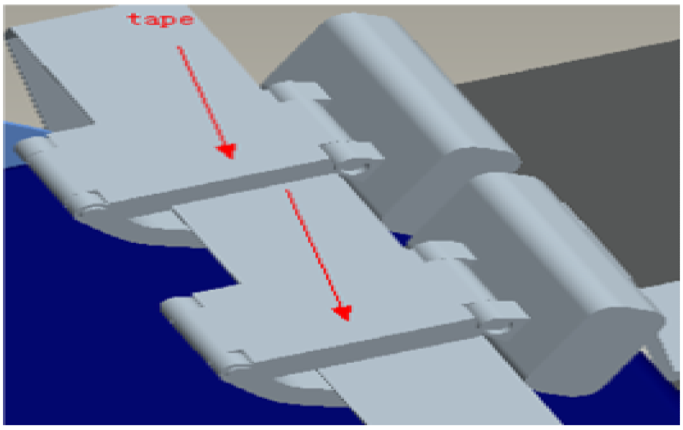
To fix the camera holder on the car, we use two screws of 5mm every side to avoid the back and forth movement. We control the width of the board as 30mm in case that it is too heavy for the screws to bear or too wide for the cameras to be put on. For the hole on the panel, we keep it symmetric with the other holes on the sideboard.

To avoid the up and down movement, the panel should not be too thick and high. The thickness determine for the aluminum board is 1.5mm which makes it common to press out. The ideal height will be determined later with the stereo vision system. For better manufacturing characteristics, we give two stamping bends to the panel each side to make the mechanical properties good enough in case that it is easily broken.

Apart from the camera holder part, it is of the same importance that the cameras should be attached to it stably. We have three methods in consideration, namely, clamps, screws and tape.

As the first idea, it will be very easy to adjust the position of the cameras if we use clamps. However, the problems come from the clamps as well. First, the clamps themselves will bring more weight and affect the stability of the camera holder. Second, how the clamps are fixed on the board will affect the functions of the cameras. Usually, we need screws then bring more errors into the position. Third, even if we can fix the clamps well, we have to fix the cameras again which makes the problem complex.

Then we think of fixing the cameras to the board directly without too many media components. After we observe a screw on the back of the camera, the idea to fix the cameras on the board with screws is very spontaneous. However, this solution might not work because of two problems. First, the screw is very small with a diameter of 2mm which makes it easily broken and very hard to drill holes on the board. Second, even if it is mechanically feasible, there is only one hole on the camera which requires us to drill another hole on the camera. Nevertheless, the plastic camera is so fragile that a small hole will bring too much stress on the seam. The easiest way is tape which is retractable and repeatable if there is any problem. The only worry is that the camera might be shaking when the car is moving because it is not strong enough. However, we have glue, seccotine and wide tape which will provide great force and stability.



How cameras are fixed onto camera holder

Camera Position

To help and simplify the CV calibration part, we require that the two cameras are basically at the same height from the ground. The errors of the obstacle in the two pictures will result in a contradictory and ambiguous consequence which will make trouble for the control part.

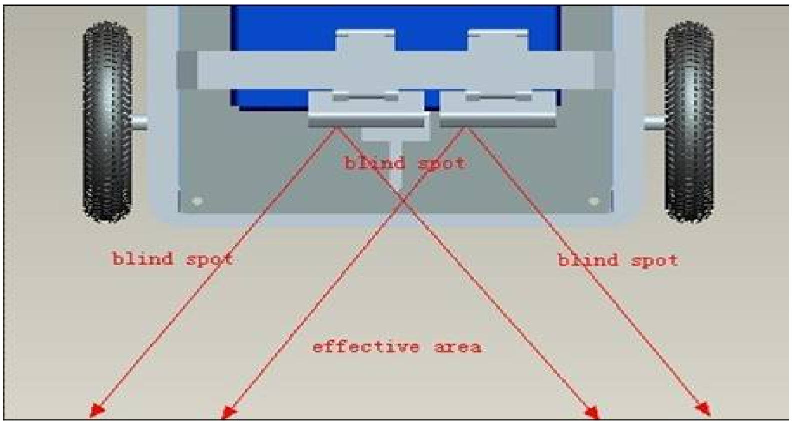
For two cameras we used, the errors in the picture come from two places:

1. The mechanical error from the cameras because they are intrinsically different;

To solve this problem, we choose two cameras from the same type at the same price to avoid possible large camera errors. We take two Logitech C170s to reduce the error but cannot eliminate it.

2. The manufacturing errors and assembly errors of the components; for another consideration, even if the two cameras are the same, the heights of their respective places are different from each other. These errors come from the manufacturing and assembly errors of the camera holder. Thus, we choose only a panel as the only one component to meet all needs which limits possible large errors.

Large field of View



Schematic diagram showing blind spot and effective area

We first put the holder in the back of the robot because the view will be larger than in the front. Apart from that, cameras in the front will be easily hurt by the obstacle or wall in the process of test at the beginning. However, if we put the cameras too back of the robot, too much redundant information will be received which will affect the decision making process. As a result, a compromise could be made that we put the cameras at the place 120mm distance from the front which is symmetric to the other two holes on the side board.

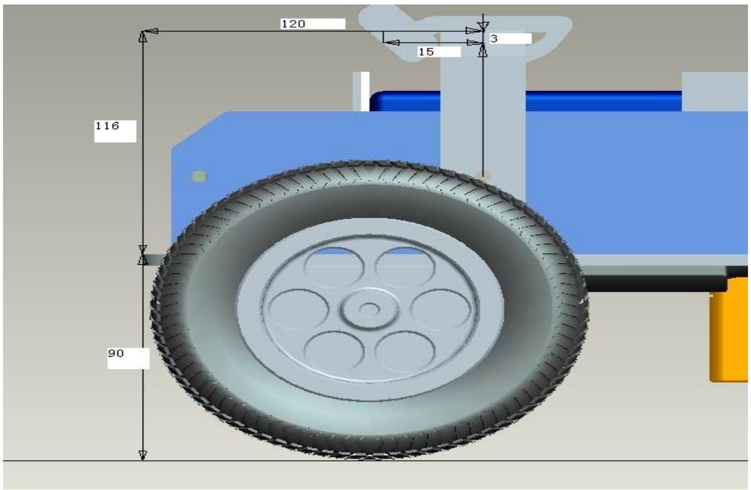
The cooperation of the cameras and the computer vision demand an ideal height where we can achieve as large view and small blind spot as possible.

We first need to measure the view angle of the camera and then calculate the height of the cameras with similar triangles.

The vertical distance from the bottom of the laptop to the ground is roughly 90mm.

The horizontal distance from the front of the robot to the camera holder is about 120mm.

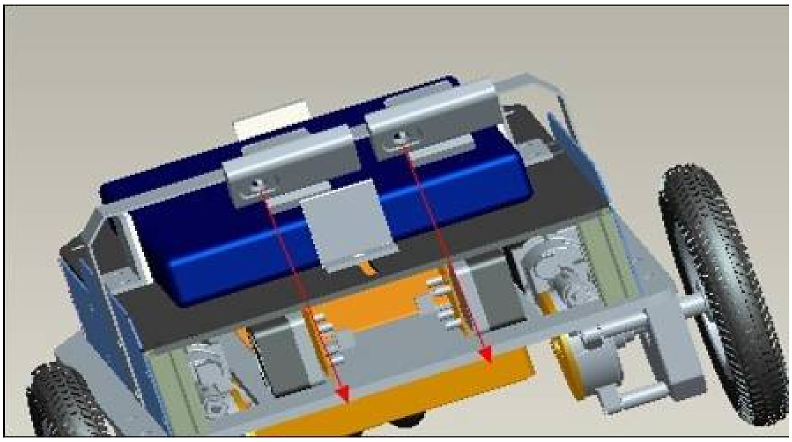
We have tested the cameras before, so the vertical and horizontal distance of the cameras could be measured as well. Compared to the camera holder, the vertical distance from the eye of the camera to holder board is 3mm. Horizontal 15mm. We finally determined the height of the camera holder as



The position of cameras on the robot

To limit the blind spot in the area of the robot itself, we need to adjust the distance between the two cameras and the depression angle many times until we can see in the picture that the laptop holder would not block the view and that we can see things far enough in two views.

When we manufacture all the components and assembly them on the car, we discover that even if we put the two cameras next to each other with no marginal place between each other, the position are rather good to take a view with nothing of the laptop holder and a view far enough for obstacle detection.



Keep the effective area away from the aluminum alloy

Chapter 5 Conclusion

The purpose of our project was to create an inexpensive and high-efficient obstacle avoidance robot for education and orientation use. In the conclusion of this report, the robot was able to display to us its turning strategy within a speed range.

The specific work we have finished are as follows:

- We successfully modified the robot with a few mechanical components to add two more functions to the main body, namely the laptop holder and the camera holder.

- We tested and optimized the AT89S52 development board designed and produced by Depush in a large scale, which was used to control the motors smoothly.

- We choose the cameras with great value at a relatively low cost, fixed them and adjusted them on the camera holder for good calibration of the computer vision system.

- After attaching the cameras, we successfully developed the software to detect the obstacles and send out signals to control the motors with OpenCV in Python.

- We establish the serial communication method between the upper laptop and the lower development board with USB port. The laptop will send out a signal of the motor condition to the development board. After receiving the signal

Chapter 6 Discussion

In order to optimize the movement of the robot, we have many considerations for improvement. However, most of these ideas will cost more money and time as well.

- For the choice of cameras, it is better to get CCD or industrial use ones to get clear and fast pictures. Even the ones we mentioned in the camera holder part will be better because of the special software. In the process of modification, we discover that the speed of the cameras is a problem because low frame rate will result in a slow picture output.

- For the computer vision part, it will be better if we have a more optimized program that could process the signals faster, which could let the robot respond to obstacles more quickly

- For the choice of motors and motor chips, we recommend that students should try to take advantage of a feedback loop. This would allow the robot to move more precisely.

- The computer vision part can provide more detailed information than left and right. With more precise motor control, it could choose specific paths instead of just turning away from obstacles.

- For the choice of development board, a faster and more efficient board will be more challenging. We suggest that students could learn to program for AVR, ARM and DSP on their own.

Chapter 7: Appendices

Control Section source code:

main.c

```
#include<BoeBot.h>
#include<uart.h>
#define HB_25 P1_1
voidFor_Ward(void);
voidLeft_Turn(void);
voidRight_Turn(void);
void Stop(void);
voidBack_Ward(void);
void HB25Initial(void);
void main(void)
{
uart_Init();//initialize the serial communication
HB25Initial();//initialize the HB-25 motor controller
while(1);
}
void UART_SER (void) interrupt 4 // program of serial interrupt service
```

```
{
unsigned char Temp;         //define a temporary variable


if(RI)                     //judge if the send interrupt emerges
   {
           RI=0;                    //clear the zonebit
           Temp=SBUF;               //read the buffer
          switch(Temp)
         {
     case 'L':
          Left_Turn();     //turn left for a certain angle
                    Stop();         //stop
                       SBUF=Temp;       //send the laptop an ‘L’
                    break;
     case 'R':Right_Turn();    // turn right for a certain angle
          Stop();          //stop
                       SBUF=Temp;        // send the laptop an ‘R’
          break;
     case 'F':For_Ward();break;   //go forward
          case 'B':Back_Ward();break;  //go back
          case 'S':Stop();break;       //stop
          }
       }
      if(TI)
    TI=0;

}
voidFor_Ward(void)  //go forward
{
   HB_25=1;
delay_nus(1700);
   HB_25=0;
delay_nus(1100);
   HB_25=1;
delay_nus(1685);
   HB_25=0;
delay_nus(5250);
}
voidLeft_Turn(void)   //turn left for a certain angle
{
   HB_25=1;
delay_nus(1300);
```

```
   HB_25=0;
delay_nus(1100);
   HB_25=1;
delay_nus(1685);
   HB_25=0;
delay_nms(400);      //400 decide the angle for turning left
}
voidRight_Turn(void)   //turn right for a certain angle
{
   HB_25=1;
delay_nus(1705);
   HB_25=0;
delay_nus(1100);
   HB_25=1;
delay_nus(1305);
   HB_25=0;
delay_nms(400);       //400 decide the angle for turning right
}
void Stop(void)    //stop
{
   HB_25=1;
```

```
delay_nus(1500);
   HB_25=0;
delay_nus(1100);
   HB_25=1;
delay_nus(1500);
   HB_25=0;
 }
voidBack_Ward(void)   // go back
{
   HB_25=1;
delay_nus(1300);
   HB_25=0;
delay_nus(1100);
   HB_25=1;
delay_nus(1305);
   HB_25=0;
  }
void HB25Initial(void)
{
     while ( !HB_25 );
     HB_25=0;
```

```
       delay_nus(5250);

       HB_25=1;

       delay_nus(1500);

    HB_25=0;

       delay_nus(1100);

       HB_25=1;

       delay_nus(1500);

       HB_25=0;

       delay_nus(5250);

}
```

BoeBot.h

```
voiddelay_nus(unsigned inti)  // delay I us,i>=12

{

i=i/10;

while(--i);

}


voiddelay_nms(unsigned int n)  // delay n ms

{

  n=n+1;

while(--n)
delay_nus(900);        //delay 1ms,and compensate
}
```

uart.h

```
/*--------------------------------------------------------------
                    8051 serial interrupt driver
--------------------------------------------------------------*/
#include <AT89X52.h>
#include <stdio.h>
#define XTAL 11059200
#define baudrate 9600

#define OLEN 8      //the size of the send buffer
unsigned char ostart;  // the starting index of the end buffer
unsigned char oend;      // the end index of the send buffer
charidataoutbuf[OLEN]; //the storage array of the send buffer

#define ILEN 8          // the size of the receive buffer
unsigned char istart;   // the starting index of the receive buffer
unsigned char iend;     // the end index of the receive buffer
charidatainbuf[ILEN]; // the storage array of the receive buffer
```

```
bitbdatasendfull;    //the mark of the full buffer

bitbdatasendactive;  //the mark of the effective sending

/*the program of serial interrupt service */

static void com_isr(void) interrupt 4 using 1        //the serial interrupt

{

  //------------ the interrupt for receiving datas -------------

char c;

if(RI)

 {

   c=SBUF;      //read the the character

   RI=0;     // clear the zonebit

if(istart+ILEN!=iend)

   {

inbuf[iend++&(ILEN-1)]=c;   //the buffer receives datas

   }

 }



 //------------the interrupt for sending datas-------------

if(TI)

 {

   TI=0;  //clear the zonebit
```

```
if(ostart!=oend)
  {
    SBUF=outbuf[ostart++&(OLEN-1)];// Transmit characters to the send
buffer
sendfull=0;        //set the mark of the full buffer
   }
else
   {
sendactive=0;        //set the sending is invalid
   }
  }
}

//PUTBUF: write characters to SBUF or the send buffer
voidputbuf(char c)
{
if(!sendfull) //send if the buffer is not full
  {
if(!sendactive)
    {
sendactive=1; //send a character directly
```

```
          SBUF=c;       //write to the SBUF start buffer
      }
else
      {
        ES=0;         //close the serial interrupt temporarily
outbuf[oend++&(OLEN-1)]=c;  //Transmit characters to the send buffer
if(((oend^ostart)&(OLEN-1))==0)
{ sendfull=1;}    //set the mark of the full buffer
         ES=1;        //open the serial interrupt
      }
    }
}
charputchar (char c)
{
if (c=='\n')            //increase a new row
    {
while(sendfull);   //wait until the send buffer is null
putbuf(0x0D);      //before LF send the new row CR
    }
while(sendfull);
putbuf(c);
```

```
return(c);
}
//getchar and gets function use _getkey
char _getkey(void)
{
char c;
while(iend==istart)      //judge if the starting index of the receive buffer equals the end index
 {;}
 ES=0;
 c=inbuf[istart++&(ILEN-1)];
 ES=1;
return(c);
}

/*the function for initializing the serial communication and the baud rate of UART */
voidcom_initialize(void)
{
istart=0;
iend=0;
```

```
ostart=0;

oend=0;

sendactive=0;

sendfull=0;


  TMOD |=0x20;   //set the mode of the timers

  SCON=0x50;     //set the mode of serial communication

  TH1=0xfd;      // the baud rate is 9600

  TL1=0xfd;

  TR1=1;         //start the timer

  ES=1;          //open the serial interrupt

}


voiduart_Init()

{

com_initialize();

  EA=1;          //CPU opens the total interrupt

}
```

Object Detection Source Code:

See attached files

Chapter 8: References

[1] YifangZhong, Changlin Wu, Zhengbao Tang, Mechanical and Machine Design() , 2 ed. Huazhong University of Science and Technology press, 2006

[2] Ming Chang, Descriptive Geometry and Engineering Graphics (), 3 ed. Huazhong University of Science and Technology press, 2004

[3] Shiquan Zhou, Fundamentals for Mechanical Manufacturing Process (), Huazhong University of Science and Technology press, 2005

[4] Jiao Ni, Guoqing Li, Qin Qian, Mechanical of Materials,(), Huazhong University of Science and Technology press, 2006

[5] Bradski, Gary , and Adrian Kaehler. Learning OpenCV: Computer Vision with the OpenCVLibrary.O'Reilly Media, 2008. Print.

[6] Jianbo Shi, Tomasi, C., "Good features to track," Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on , vol., no., pp.593-600, 21-23 Jun 1994

[7] Bouguet, Jean-Yves. "Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm." Print.