The Robot: Eos-Explorer

Chenran Ye
UFID: 17618334

EEL 4665/5666
Intelligent Machines Design Laboratory
Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

TAs: Ryan Stevens
Tim Martin

Table of Contents

Abstract

The Eos-Explorer: an autonomous navigation robot that is designed to report environment condition of certain location back by real time video captured.

The Eos-Explorer has the capability to locate itself and move over to the desired destination that has been sent to it from the base station which would be a PC in this case. At the meantime, it will avoid the obstacles on the route. By using sonar sensors and camera which is used to recognize characters set at the destination location, the Eos-Explorer will find the right position and move towards to it. Once finishing the navigation, the LCD will display the message that saying ‘Reach the End’. Project involves communication between the robot and laptop by implementing X-Bee module. IP camera is used to achieve wireless communication.

Exclusive Summery

Eos-Explorer is autonomous smart navigation robots that is able to location itself and find the desired destination by self-routing and object detection and tracking.

Eos-Explorer begins its mission by receiving a message from the base station which considered as a command unit, saying what the destination location it is based on latitude and longitude. It turns its servos to drive to the location while doing obstacle avoidance on the way.

In this case, sonar sensors are used for obstacle avoidance, and a GPS receiver is used to get location information. However according to the accuracy of the GPS receiver, it can only help the robot get closer to the destination. Therefore, image processing is implemented in order to help the robot eventually reach the final destination.

Two useful sources have been used for image processing part of this project. One is AForge.Net, which I first used for object tracking; the other one is a powerful library called OpenCV. These are all open source software that does really help to achieve the goal.

Introduction

-Background

With the rapid development of the whole world, the environment problems become more and more serious. More and more disasters took place around us. While it happened, we need to take actions as soon as possible, however in some kinds of situation, it is hard to send people to where it is suffering bad living environment. Therefore it is necessary to come up with a model of robot that can help with this difficulty. Although the nature environment is too complicated to deal with for now, the navigation and obstacle avoidance function can be achieved in my robot. Also, it has the possibility to be improved for future use.

-Scope and Objective

The Objective is to make the GPS module work properly with the object avoidance function. If time allowed touch screen may also be implemented as an interface other than base station, otherwise it will work as graphic LCD for display purpose.

Integrated System

-Base Station/Command Unit

There are two modes available on the base station that can be select for the robot. When it is on fully-controlled mode, the robot will work exactly follow the commands received from the base station. Under the autonomous mode, the base station is used to send out destination information then receive and process the video received from the robot.

-MCU

The MCU considered as the brain of the robot, is used to process all signals from other components and make decision what to do next by the robot. In this case, the Arduino Mega 2560 is chosen for the robot.

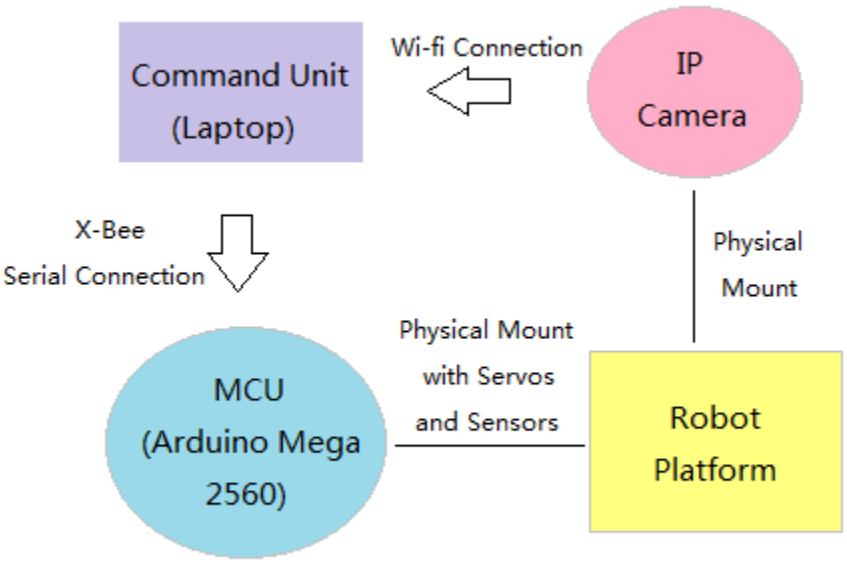
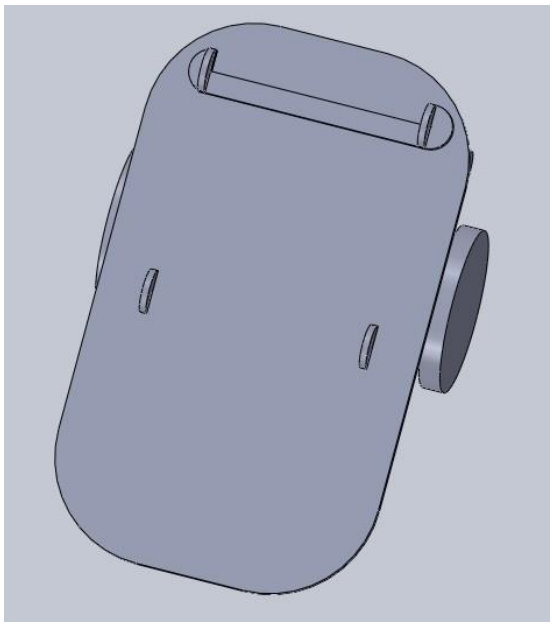


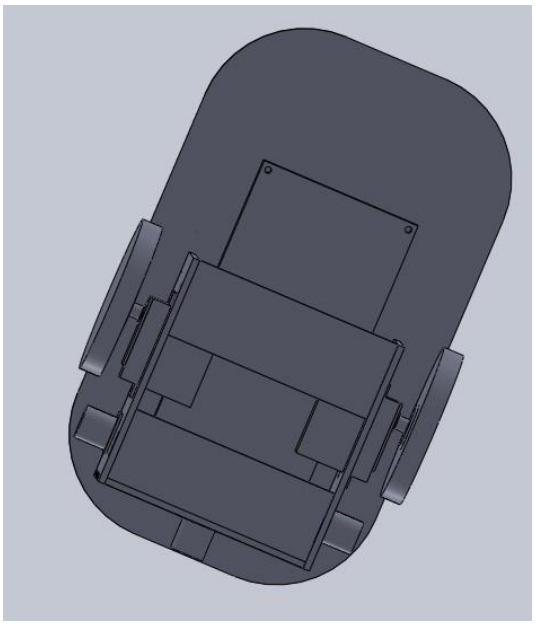
Figure 1 Eos-Explorer Integration Block Diagram

Mobile Platform

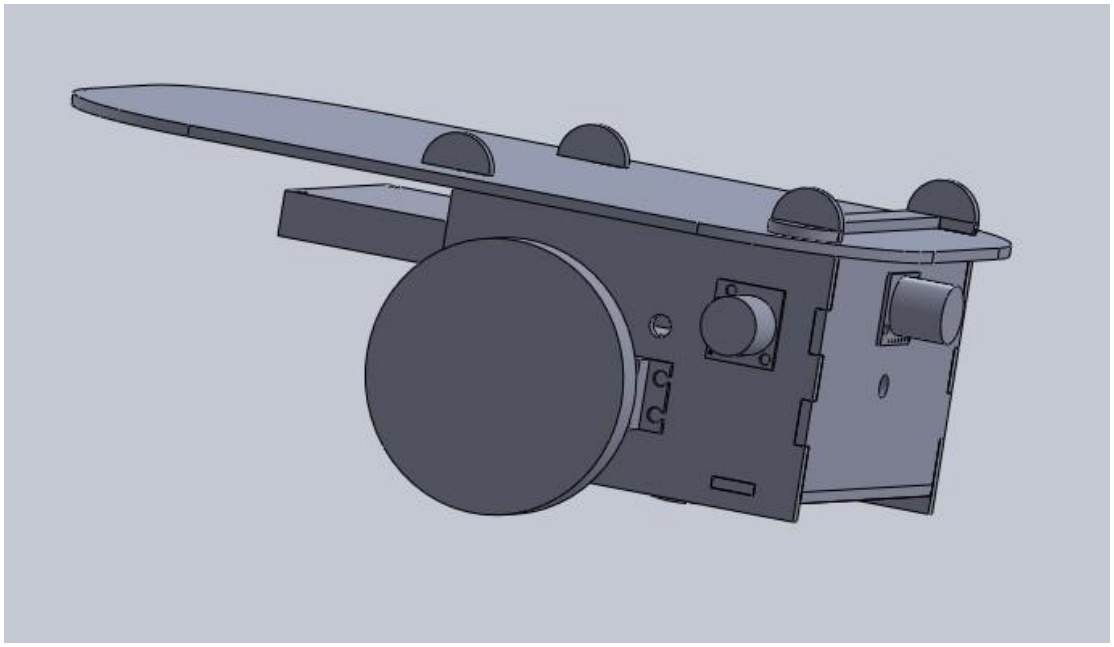
The mobile platform will be constructed by using the wood material provided in the lab. It contains two motors in the front used for moving and adjusting direction with the help of universal wheel that can turn to all directions.



(a) Top View



(b) Bottom View



(c) Side View

Figure 2 Mobile Platform assembled using SolidWorks

Actuations

The motion of this robot is driven by two servo motors and a universal wheel. The two servos in the front is used to move forward, as well as adjust the direction of moving.

Sensors

-GPS receiver

The GPS module is a 20 channel EM-406A SiRF III Receiver with antenna.

Figure 3 EM-406A GPS Receiver

This is a module running at 5VDC input power that transmitting the output navigation and measurement data to the robot. It can be tested by connecting to PC using GPS Evaluation board from Sparkfun.com. A sample terminal program provided by kronosrobotics.com can be used to see the raw NMEA data that the GPS unit is transmitting.

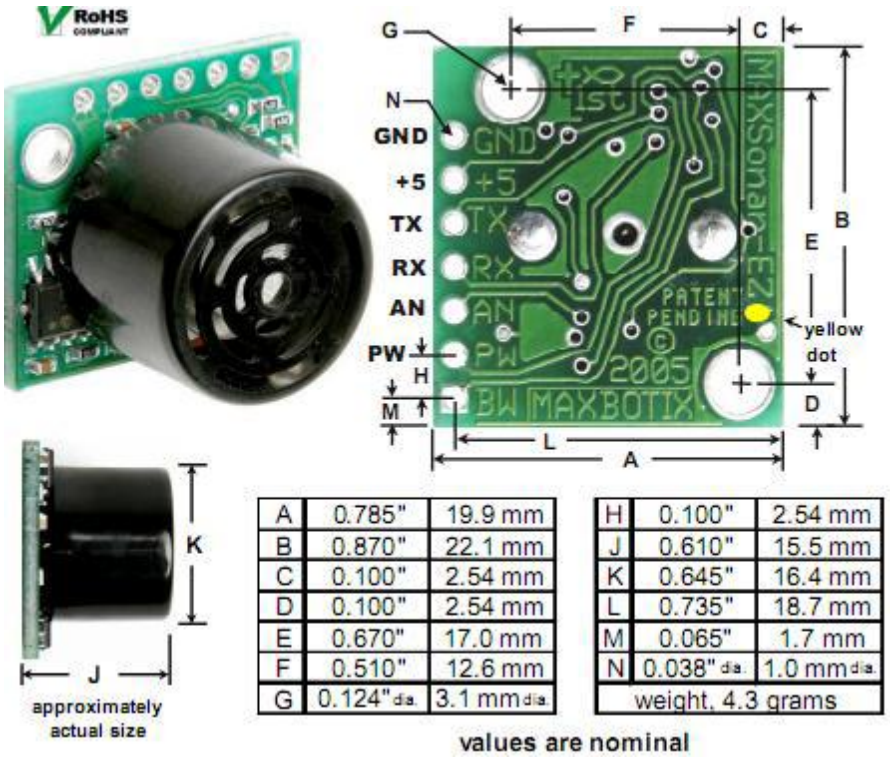
Here are messages that we will take a look at the NMEA protocol:

Message	Description
GGA	Time, Position, Fix Type
GSA	GPS receiver operating mode, Satellite used in the position solution, DOP values
GSV	The number of GPS satellites in view, Satellite ID numbers, Elevation, Azimuth, SNR values.
RMC	Time, Date, Position, Course, Speed

These messages can be processed to use for identify the current location of the robot, and then compare with the desired location to give navigation to the Eos-Explorer.

However under the limitation of the accuracy, this GPS module can only help the robot get closer to the destination around 5m to 10m.

-Sonar sensor



A	0.785"	19.9 mm
B	0.870"	22.1 mm
C	0.100"	2.54 mm
D	0.100"	2.54 mm
E	0.670"	17.0 mm
F	0.510"	12.6 mm
G	0.124" dia.	3.1 mm dia.

H	0.100"	2.54 mm
J	0.610"	15.5 mm
K	0.645"	16.4 mm
L	0.735"	18.7 mm
M	0.065"	1.7 mm
N	0.038" dia.	1.0 mm dia.
weight, 4.3 grams		

values are nominal

Figure 4 Sonar Sensor Dimension

Three Ultrasonic Range Finder sonar sensors (Maxbotix LV-EZ0) are attached to the front as well as left and right side of the Eos-Explorer. PMW output that is being sent by the Maxbotix device can be read and used to calculate the distance with a scale factor of 147 uS per Inch. When the distance detected by front sensor is less than 20cm, if the distance to the left is less than 20cm then turn right; if the distance to the right is less than 20cm, turn left; if both distance to the right and left are more than 20cm, compare these two distances, turn right when has longer distance on the right, turn left when left side has more space.

-IP Camera

Figure 5 Foscam Wireless IP Camera

This Genuine Foscam Fi8908w Wireless IP Camera will be used to capture images/video on the way that Eos-Explorer approaching to the destination. Also this is used to detect the destination sign and help the robot get as closer as possible to the destination.

The destination sign will be an object in certain color and shape. It will be preset in the PC. When the images captured by the camera collected by openCV using certain format of IP address, the function of detect and track moving object can be provided.

-Communication

Two X-Bee S1 modules and Explorer kit for both PC and MCU are implemented for communication purpose.

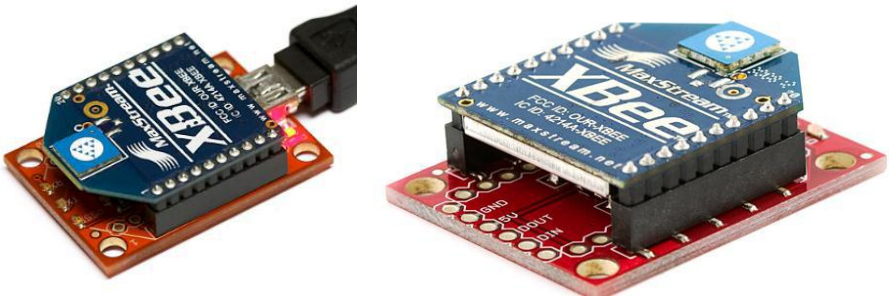


Figure 6 X-Bee Modules with Explorer

Behaviors

The full function can be summarized as 3 steps:

-Locate and Route

Eos-Explorer gets current location by using GPS module and moves closer to the destination location sent by base station (PC).

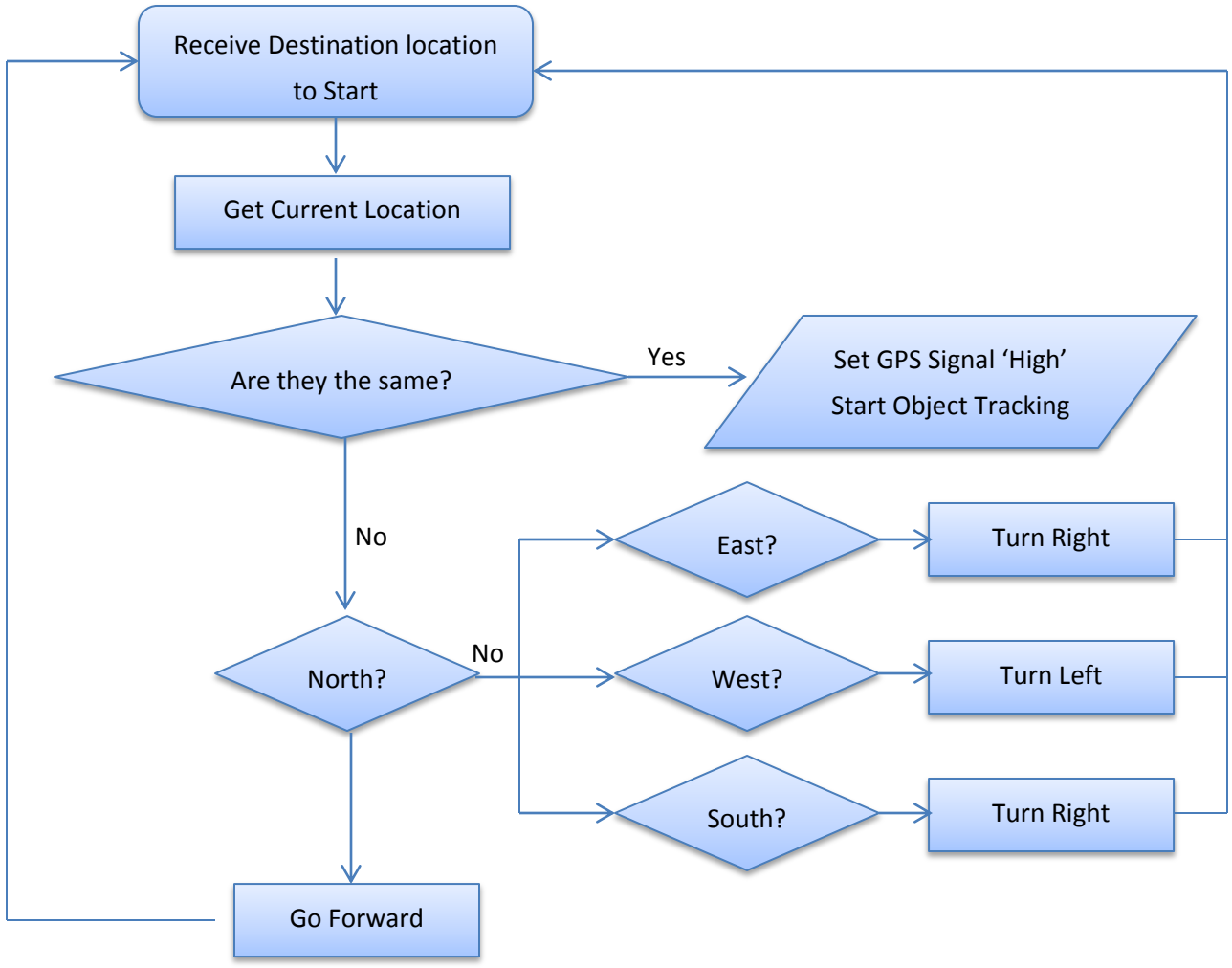


Figure 7 GPS Navigation Flow Chart

-Obstacle avoidance

Stop and make a turn when detect objects in the range of preset distance.

-Object Detect and Track

By using camera find the object that represent final destination and send command to the Eos-Explorer in order to keep the object remain in the middle of the camera view.

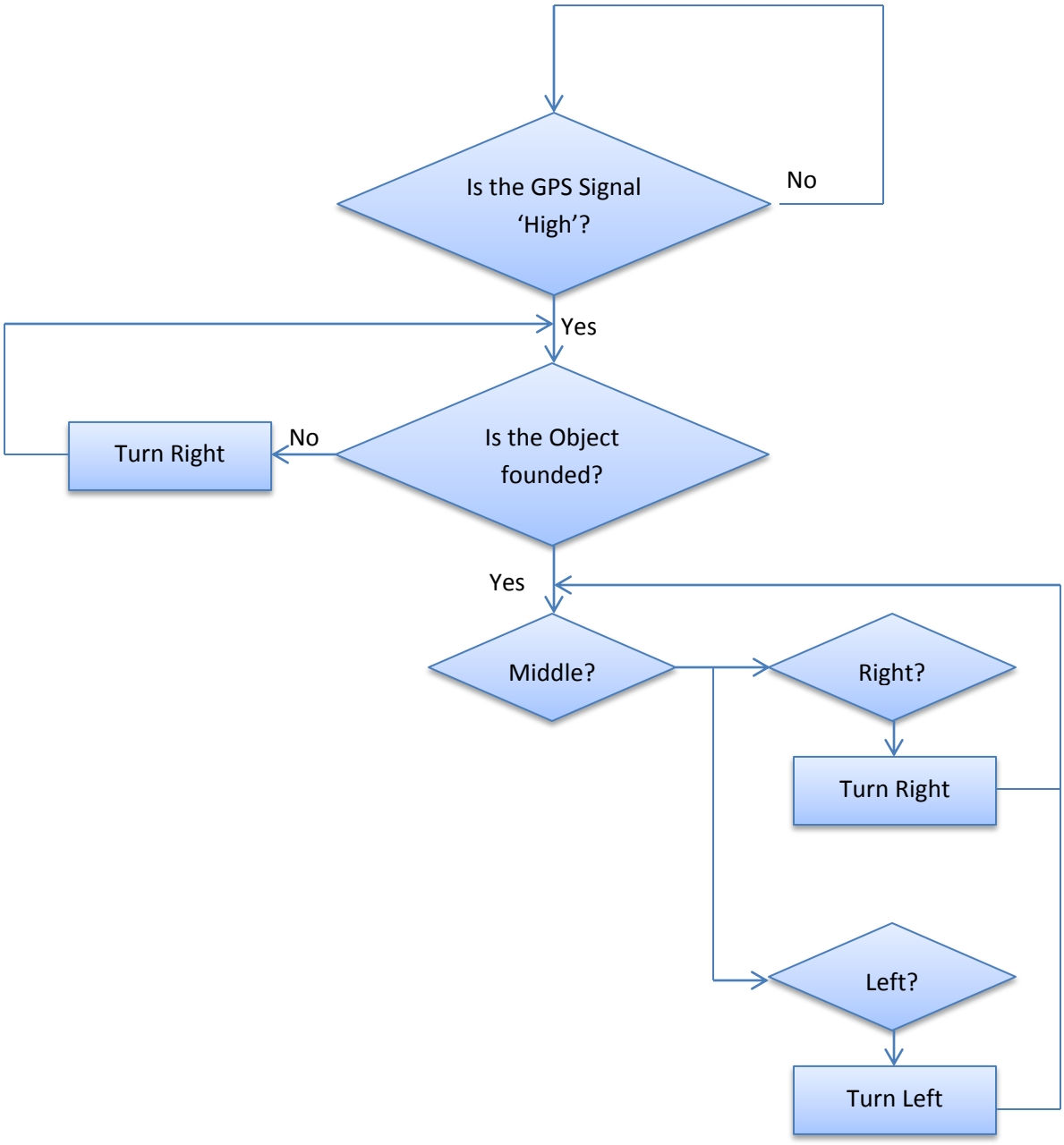


Figure 8 IP Camera Flow Chart

Experimental Layout and Conclusion

Figure 9 Platform of Eos-Explorer

-Work Accomplished

Eos-Explorer has amount of work that was accomplished:

- Mobile platform design, construction and assembly
- Integration of sensors, communications between computer and microprocessor
- Identify location behavior
- Object Avoidance behavior
- Computer vision

-Limitations of Work

Since the cheap GPS module has be used in the project, sometimes it is hard to get satellite signal inside the building. It needs long time to warm up in order to get location data in. Also the accuracy was not good enough therefore it may not be able to drive the robot to where can really see the destination sign.

For the object detect and tracking section, although it would work fine for the tracking part whenever the object has been detect, sometime it will have trouble to detect the sign because of the angle from where the robot locate to the sign. Considered of this I switch to color detect for the demo. Better algorithm may be implemented later for better results.

Documentations

-Maxbotix LV EZ0 datasheet:
http://www.maxbotix.com/uploads/LV-MaxSonar-EZ0-Datasheet.pdf
-Arduino Mega 2560:
http://www.arduino.cc/en/Main/ArduinoBoardMega2560
-Opencv:
http://www.opencv.org.cn/index.php/
http://code.google.com/p/cvblob/
-Aforge.NET:
http://www.aforgenet.com/

Appendix

A. Mobile Platform Main Code

```
#include <Servo.h>
#include <TinyGPS.h>
#include <LiquidCrystal.h>

//Digital pin 7 for reading in the pulse width from the MaxSonar device.
//This variable is a constant because the pin will not change throughout execution of this code.
const int pwPinFront = 7;
const int pwPinLeft = 6;
const int pwPinRight = 5;
//variables needed to store values
long pulse, inches, cm;
long pulseLeft, inchesLeft, cmLeft;
long pulseRight, inchesRight, cmRight;
int val = 0;
Servo Left;
Servo Right;

LiquidCrystal lcd(32, 30, 22, 24, 26, 28);
TinyGPS gps;
void getgps(TinyGPS &gps);

void setup()
{
  //This opens up a serial connection to shoot the results back to the PC console
  Serial.begin(9600);
```

```
   Left.attach(8);
   Right.attach(9);
   Serial1.begin(4800);
   lcd.begin(16, 2);
   lcd.print("Waiting for GPS");
   Serial2.begin(9600);
   Serial.println("...Xbee Test...");
}

void loop()
{
   pinMode(pwPinFront, INPUT);
   //Used to read in the pulse that is being sent by the MaxSonar device.
   //Pulse Width representation with a scale factor of 147 uS per Inch.
   pinMode(pwPinLeft, INPUT);
   pinMode(pwPinRight, INPUT);

   pulse = pulseIn(pwPinFront, HIGH);
   pulseLeft = pulseIn(pwPinLeft, HIGH);
   pulseRight = pulseIn(pwPinRight, HIGH);
   //147uS per inch
   inches = pulse/147;
   inchesLeft = pulseLeft/147;
   inchesRight = pulseRight/147;
   //change inches to centimetres
   cm = inches * 2.54;
   cmLeft = inchesLeft * 2.54;
   cmRight = inchesRight * 2.54;

   if (cm < 20)
   {//hold();
      //if (cmLeft > cmRight)
      //    {leftTurn();}
      //else
      //    {rightTurn();}
      if (val == 77)
      {hold();}
      else
      {rightTurn();}
   }
   else
   {
      if (val == 76)
      {leftTurn();}
```

```
      else
      {
        if (val == 82)
        {rightTurn();}
        else
        {forward();}
      }

  }

  while(Serial1.available())          // While there is data on the RX pin...
  {
       int c = Serial1.read();        // load the data into a variable...
       if(gps.encode(c))              // if there is a new valid sentence...
       {
          getgps(gps);                // then grab the data.
       }
  }

  while(Serial2.available())
  {
    val = Serial2.read();
    Serial.print("I received: ");
    Serial.println(val);
  }

  Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(500);
}

void forward()
{
  Left.write(180);
  Right.write(0);
}

void backward()
{
  Left.write(0);
  Right.write(180);
```

```
}

void leftTurn()
{
   Left.write(0);
   Right.write(0);
}

void rightTurn()
{
   Left.write(180);
   Right.write(180);
}

void hold()
{
   Left.write(90);
   Right.write(95);
}

void getgps(TinyGPS &gps)
{
   // Define the variables that will be used
   float latitude, longitude;
   // Then call this function
   gps.f_get_position(&latitude, &longitude);
   // You can now print variables latitude and longitude
   lcd.setCursor(0,0);
   lcd.print("Lat :    ");
   lcd.print(latitude,5);
   lcd.setCursor(0,1);
   lcd.print("Long: ");
   lcd.print(longitude,5);

   // Here you can print statistics on the sentences.
   unsigned long chars;
   unsigned short sentences, failed_checksum;
   gps.stats(&chars, &sentences, &failed_checksum);

}
```

B. Main Code of AforgePlatform for object tracking

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.IO;
using System.Windows.Forms;
using AForge.Video.DirectShow;
using AForge.Video;
using AForge.Vision.Motion;
using AForge.Imaging;
using System.Threading;
using ImageProcessor;
using Tracker;
using UIGraphic;
using Statistic;
namespace AForgePlatform
{
public partial class Form1 : Form
{
//vars
private FilterInfoCollection videoCaptureDevices;
private VideoCaptureDevice videoCaptureDeviceSelected;
Bitmap im;
public Form1()
{ InitializeComponent();
}
private void Form1_Load(object sender, EventArgs e)
{
_path = "..\\..\\";
//Load images
_isTracking = false;
DeviceSelection.Checked = false;
FileSelection.Checked = true;
videoCaptureDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);
foreach (FilterInfo vidDevice in videoCaptureDevices)
{
```

```
LocalDevice.Items.Add(vidDevice.Name);
}
LocalDevice.SelectedIndex = 0;
LocalFile.Items.Add("Image Sequence");
LocalFile.SelectedIndex = 0;
}
}
void videoCaptureDeviceSelected_NewFrame(object sender, NewFrameEventArgs eventArgs)
{
Bitmap image = (Bitmap)eventArgs.Frame.Clone();
picResult.Image = image;
im = image;
}
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
}
//create the target selector
private void CreateSelector()
{
if (_selector == null)
{
_selector = new Selector(picPreview.Width, picPreview.Height);
_selector.TargetROISelected = new Selector.TargetSelected(this.CreateTargetModel);
//wire preview window events to selector's events
picPreview.MouseDown += new MouseEventHandler(_selector.OnMouseDown);
picPreview.MouseMove += new MouseEventHandler(_selector.OnMouseMove);
picPreview.MouseUp += new MouseEventHandler(_selector.OnMouseUp);
}
}
//create the model when the target is slected
private void CreateTargetModel()
{
//create the model
Tracker.CreateTargetModel((Bitmap)picPreview.Image, Bin1, Bin2, Bin3, _selector.TargetWindow,
_selector.SearchWindow);
}
private int Bin1
{
get { return Convert.ToInt32(textBox1.Text); }
}
private int Bin2
```

```
{
get { return Convert.ToInt32(textBox2.Text); }
}
private int Bin3
{
get { return Convert.ToInt32(textBox3.Text); }
}
//creates the histograms
private Histogram CreateHistogram(Bitmap img)
{
return _imgProc.Create1DHistogram(img, Bin1, Bin2, Bin3);
}
private void DisplayInformation(Dictionary<string, string> information)
{
lblInfo.Text = "";
foreach (KeyValuePair<string, string> kv in information)
{
lblInfo.Text += kv.Key + ": " + kv.Value + Environment.NewLine;
}
lblInfo.Refresh();
}
//Track the object
private void TrackObject(Bitmap bmp)
{
//run track on the tracker to find centroid
Window roi;
Window searchRoi;
Dictionary<string, string> information;
Tracker.Track(bmp, out roi, out searchRoi, out information);
Bitmap b = Tracker.ProcessedImage;
//Draw the centroid if valid
if (roi.CentreX > 0 && roi.CentreY > 0)
{
//Display Info
DisplayInformation(information);
}
picPreview.Image = b;
picPreview.Refresh();}
private ITracker Tracker
{
get
```

```
{
if (_tracker == null)
CreateTracker();
return _tracker;
}
}
private void CreateTracker()
{
if (TrackerCentroid.Checked)
_tracker = new TrackerFactory(TrackerType.CENTROID).Tracker;
else if (TrackerMS.Checked)
_tracker = new TrackerFactory(TrackerType.MEANSHIFT).Tracker;
else if (TrackerMSK.Checked)
_tracker = new TrackerFactory(TrackerType.MEANSHIFTKERNEL).Tracker;
else if (TrackerMSKBG.Checked)
_tracker = new TrackerFactory(TrackerType.MEANSHIFTKERNELBG).Tracker;
}
#region "CONTROL EVENTS"
private void cyLoadSequence_Click(object sender, EventArgs e)
{
//makes selected camera the final camera and starts capture
if (DeviceSelection.Checked == true)
{
videoCaptureDeviceSelected = new
VideoCaptureDevice(videoCaptureDevices[LocalDevice.SelectedIndex].MonikerString);
videoCaptureDeviceSelected.NewFrame += new
NewFrameEventHandler(videoCaptureDeviceSelected_NewFrame);
videoCaptureDeviceSelected.Start();
picPreview.Image = im;
}
else if (FileSelection.Checked == true)
{
if (videoCaptureDeviceSelected != null)
{
videoCaptureDeviceSelected.Stop();
}
if (LocalFile.SelectedIndex == 0)
{
folderBrowserDialog1.SelectedPath = "C:\\Aforge\\AviSequence";
if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
{
```

```
Cursor.Current = Cursors.WaitCursor;
_path = folderBrowserDialog1.SelectedPath;
picPreview.Image = null;
picResult.Image = null;
//picPreview.Padding = new Padding(10, 10, 10, 10);
picPreview.Refresh();
picResult.Refresh();
_histTarget = null;
_histCandidate = null;
lblInfo.Text = "";
_count = 0;
string imagePath = "";
Bitmap b = null;
string[] files = Directory.GetFiles(_path, "*.jpg");
if (files.Length > 0)
{
imagePath = files[0];
b = (Bitmap)System.Drawing.Image.FromFile(imagePath);
}
else
{
files = Directory.GetFiles(_path, "*.bmp");
imagePath = files[0];
b = (Bitmap)System.Drawing.Image.FromFile(imagePath);
_imageFileExtension = ".bmp";
}
picPreview.Image = b;
picResult.Image = b;
FileInfo fi = new FileInfo(imagePath);
_imageFileExtension = fi.Extension;
picPreview.Refresh();
picResult.Refresh();
_trackSequenceCount = files.Length;
}
}
}
CreateSelector();
Cursor.Current = Cursors.Default;
}
//Tracking
private void cyTracking_Click(object sender, EventArgs e)
```

```
{
if (cyTracking.Text == "Start Tracking")
{
_isTracking = true;
cyTracking.Text = "Stop Tracking";
timer1.Enabled = true;
timer1.Start();
}
else if (cyTracking.Text == "Stop Tracking")
{
timer1.Stop();
timer1.Enabled = false;
cyTracking.Text = "Start Tracking";
}
}
#endregion
string _imageFileExtension = ".jpg";
int _trackSequenceCount = 0;
int _count = 0;
string _path = "";
Processor _imgProc = new Processor();
Histogram _histTarget = null;
Histogram _histCandidate = null;
Selector _selector = null;
ITracker _tracker = null;
bool _isTracking = false;
private void timer1_Tick_1(object sender, EventArgs e)
{
if (FileSelection.Checked)
{
if (_count < _trackSequenceCount)
{
//set the next image
Bitmap bmp = (Bitmap)System.Drawing.Image.FromFile(_path + "\\" + _count +
_imageFileExtension);
TrackObject(bmp); //do tracking
_count++; //move to next image
}
else
{
cyTracking.Text = "Start Tracking";
```

```
_count = 0;
timer1.Enabled = false;
timer1.Stop();
}
}
else
if (DeviceSelection.Checked)
{
TrackObject(im);
}
}
}
}
```

For more detailed code, please check the website at
https://sites.google.com/site/eosexplorerye/