ECE495G Lab Instruction Manual

# Bob Baker LJ Huseinovic Kirk Iler

# Table of Contents

[Overview 3](#_Toc269214803)

[Hardware 3](#_Toc269214804)

[Software 3](#_Toc269214805)

[Software 4](#_Toc269214806)

[Software Setup 4](#_Toc269214807)

[Installation of Aria 4](#_Toc269214808)

[Creating and Configuring Project Files in Visual Studio 2008 4](#_Toc269214809)

[Configuring Visual Studio 2008 5](#_Toc269214810)

[Installation of Mobile Sim 7](#_Toc269214811)

[Installation of OpenCV 7](#_Toc269214812)

[Initializing 10](#_Toc269214813)

[Sample Code 10](#_Toc269214814)

[Sonar 12](#_Toc269214815)

[Laser 13](#_Toc269214816)

[Camera 16](#_Toc269214817)

[Appendix A 19](#_Toc269214818)

[ARIA Commands 19](#_Toc269214819)

[OpenCV Commands 20](#_Toc269214820)

[Appendix B 22](#_Toc269214821)

[Device Driver for RadioShack USB to Serial Cable 22](#_Toc269214822)

# Overview

## C:\Documents and Settings\Kirk\Desktop\Manual Images\Courtesy of MobileRobotics\NavigationSoftware_Image3.jpgHardware

The robot to be utilized for the course is a MobileRobots research robot called Pioneer 3-DX. (P3-DX) The P3-DX consists of an onboard microcontroller interfaced with a sonar, a laser, and a drive system. The robot keeps track of its coordinates and orientation based on how the wheels have been rotated, and newer models utilize an internal gyroscope to provide accurate correction. The robot and software also allow for interfacing with a Pan Tilt Zoom camera mounted on the robot.

Image taken from MobileRobots



1. Power switch for the robot
2. Main power charge (battery

charger port)

1. Serial Port to connect robot to

the computer via Serial to USB

cable

## Software

When connecting to and using the P3-DX robots, we will use the provided MobileRobots code library, called Aria. This library contains all the software needed to interface your PC with the robot over a serial connection. When creating a program using the robot, we will write our own code in C++ which uses the Aria commands to connect to and direct the robot.

The development environment we will use in this course is Microsoft Visual Studio 2008, which we will run on Windows XP or Windows 7. Microsoft Visual Studio is used to write, compile, build, and debug applications written in C++.

It is possible to do everything in this manual on the Linux operating system, but we have chosen to use Windows to make things more standardized.

# Software Setup

## Installation of Aria

Aria can be downloaded from the Mobile Robots website:

http://robots.mobilerobots.com/wiki/ARIA

Download the Windows executable and run it.

Click Next, Agree to the Terms and Conditions, Click Next a couple more times.

Install the Contents into the following destination folder for simplicity.

C:\Program Files\MobileRobots\Aria

Useful Information

The reference manual “Aria-Reference” is provided in the folder C:\Program Files\MobileRobots\Aria

Sample codes are provided in C:\Program Files\MobileRobots\Aria\examples

Source files for various integrated commands are provided in C:\Program Files\MobileRobots\Aria\src

## Creating and Configuring Project Files in Visual Studio 2008

Once Aria is installed, we need to use Microsoft Visual Studio 2008 to create a project and then configure it to recognize the Aria library.

1. In Visual Studio 2008, create a new project file by going to: File -> New -> Project

Choose: Visual C++ -> General -> Empty Project

2. Bring up the Solution Explorer by going to: View -> Solution Explorer

3. In the Solution Explorer box on the left, right click the ‘Source Files’ folder.

Choose: Add -> New Item -> Visual C++ -> Code -> C++ File (.cpp) -> Add

4. Put sample robot code into this newly created C++ File for testing.

C++ files can be added to the project by right clicking on “source files” and then adding “existing item”. These files can be removed by right clicking on the file itself and then selecting “exclude from project”. It is recommended to keep all active code files in one folder such that they are always added from the same place. Code backups should be stored elsewhere and never added to the solution to be sure that they are not accidentally overwritten. One must always open the solution file to build any project. Opening the C++ file only allows modification of code.

## *Configuring Visual Studio 2008*

For the first time using Visual Studio 2008, do the following.

Go to: Tools -> Options -> Projects and Solutions -> VC++ Directories

Select the Platform as Win32.

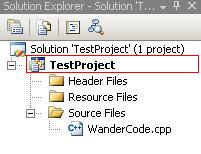
Select Executable Files, Add C:\Program Files\MobileRobots\Aria\bin

Select Include Files, Add C:\Program Files\MobileRobots\Aria\include

Select Library Files, Add C:\Program Files\MobileRobots\Aria\lib

Select Source Files, Add C:\Program Files\MobileRobots\Aria\src

Configuring Any Project



The Visual Studio 2008 ‘Project’ is outlined in the box. Above it is the ‘Solution’ At the bottom is the source code.

It is recommended to use only one Visual Studio 2008 Project for this course, otherwise each project must be reconfigured appropriately. The following must be done each time you create a new project.

Right Click the Project in the Solution Explorer, select Properties -> Configuration Properties

1. Go to: C/C++ -> General (NOT Configuration Properties -> General)

Select Additional Include Directories, click “…”, and add the following:

C:\Program Files\MobileRobots\Aria\include

2. Go to: C/C++ -> Code Generation

Change Runtime Library to “Multi-threaded DLL (/MD)”

(This is NOT “Multi-threaded Debug DLL (/MDd)”)

3. Go to: Linker -> Input

In Additional Dependencies add the following: AriaVC9.lib

Building the Program

In the Solution Explorer right click the Project and ‘Build’. A project must be rebuilt each time a change is implemented to update the executable. Sometimes minor changes to a project are not noticed by the ‘Build’ command, and one must ‘Rebuild’ to force an update. Building a project will create an executable file in the following folder.

C:\Documents and Settings\user\My Documents\Visual Studio 2008\Projects\ProjectName\Debug

Before running this program, go to C:\Program Files\MobileRobots\Aria\bin and copy AriaVC9.dll into the folder with the executable.

Running the Program

It is recommended to run the executable in the Visual Studio 2008 Command Prompt (in the ‘Tools’ menu) in order to troubleshoot any errors.

In the command prompt type the following line by line.

cd \

cd Documents and Settings\user\My Documents\Visual Studio 2008\Projects\ProjectName\Debug

ProjectName

Instead of having to change to the right directory every time the project is executed, one may change the default directory to the project directory by performing the following.

Tools 🡪 External Tools 🡪 VS 2008 Command Prompt 🡪 Initial directory: $(TargetDir)

If there is a COM port error, you may need to manually assign which COM port to connect to the robot, laser, or other device. In this case the final command may be as follows.

ProjectName.exe -rp COM1 –lp COM2

The command –rp is for the robot port, and the command –lp is for the laser port. As shown later, these default arguments can be added to the program so that they do not need typed into the command prompt each time. Typically the robot may be plugged into any of COM1-COM12, so one must either figure out how the PC USB ports are assigned or simply perform trial and error. The port assignment can be seen under Device Manager 🡪 Ports. To get to Device Manager go to Control Panel and then System. In Windows XP and older, Device Manger is found the same way, but it is under the Hardware tab in System.

In case the USB to Serial converter cable is not recognized, see Appendix B for driver information.

## Installation of MobileSim

MobileSim is a simulation software designed for testing code or robot behavior without actually connecting to the robot.

MobileSim is also available from the MobileRobots website:

* http://robots.mobilerobots.com/wiki/MobileSim
* Download MobileSim 0.5.0 – Windows
* Run the installer. Click Next. Agree to the Terms and Conditions.
* Install the Contents into the following destination folder for simplicity.
  + C:\Program Files\MobileRobots\MobileSim

## Installation of OpenCV

OpenCV is a free, open-source online C++ library that provides a large array of useful image processing and camera tools. We will be using OpenCV to load video, perform imaging tasks, and display to the computer’s screen.

* Download OpenCV-2.1.0-win32-vs2008.exe from (NOT the main file but a subfile)

<http://sourceforge.net/projects/opencvlibrary/files/>

* Run the executable and install OpenCV in C:\OpenCV2.1
* When asked, you may choose “Do not add OpenCV to the system path” or otherwise

First Time Configuring OpenCV in Visual Studio C++:

\*Note: Some files may not be necessary, but we aren’t sure at this point.

Go to: Tools -> Options -> Projects and Solutions -> VC++ Directories

Select the Platform as Win32.

Select Executable Files, Add C:\OpenCV2.1\bin

Select Include Files, Add C:\OpenCV2.1\include and

C:\OpenCV2.1\include\opencv

Select Library Files, Add C:\OpenCV2.1\lib

Select Source Files, Add C:\OpenCV2.1\src

Each Time a New Project is configured in Visual Studio C++:

Right click on the project, click properties, go to Configuration Properties -> C/C++ -> General

In additional include directories, click on ‘…’ and add the following.

C:\OpenCV2.1\src\ml

C:\OpenCV2.1\src\highgui

C:\OpenCV2.1\src\cxcore

C:\OpenCV2.1\src\cvaux

C:\OpenCV2.1\src\cv

C:\OpenCV2.1\include\opencv

Still in Project Properties, go to Configuration Properties -> Linker -> Input

In additional dependencies add the following libraries separated by spaces.

cv210.lib cvaux210.lib cxcore210.lib highgui210.lib ml210.lib

The following is a simple program that can be used to test OpenCV.

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

{

IplImage\* img = cvLoadImage( argv[1] );

cvNamedWindow( “Example1”, CV\_WINDOW\_AUTOSIZE );

cvShowImage( “Example1”, img );

cvWaitKey(0);

cvReleaseImage( &img );

cvDestroyWindow( “Example1” );

}

This program loads and displays an image from the directory the program is in.

Build this project in Visual Studio 2008, and copy the following files from C:\OpenCV2.1\bin into the folder with the executable.

highgui210.dll ml210.dll cv210.dll cvaux210.dll cxcore.210.dll

To execute the program from the command prompt while in the correct folder, type the following:

Project “picture.jpg”

\*Note that in certain programs, one may need to manually add or exclude libraries to get the code to work properly.

In the code one may have to add a line to include winsock2.h

#include "winsock2.h"

In the project one can exclude a library causing issues such as atlthunk.lib by the following procedure

Right click the Project in the Solution Explorer, select Properties -> Configuration Properties

Go to: Linker -> Input

In ‘Ignore Specfic Library’ add atlthunk.lib

# Initializing

## Sample Code

The following code is a standard way of initializing the robot, sonar, laser and camera in Aria.

#include "Aria.h"

#include <stdio.h>

#include <cv.h>

#include <highgui.h>

#include <videoInput.h>

using namespace std;

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

{

Aria::init();

ArArgumentParser argParser(&argc, argv);

ArRobot robot;

ArRobotConnector robotConnector(&argParser, &robot);

ArLaserConnector laserConnector(&argParser, &robot, &robotConnector);

argParser.addDefaultArgument("-rp COM4"); //The COM port number will need

changed for each computer

///////////////////////////////////////////////////

////////LASER INITIALIZATIONS LMS200 ROBOTS ONLY//////

//argParser.addDefaultArgument("-lp COM5");

//The COM port number will need changed for each computer

//argParser.addDefaultArgument("-connectLaser");

///////////////////////////////////////////////////

///////////////////////////////////////////////////

////////LASER INITIALIZATIONS LMS100 ROBOTS ONLY//////

argParser.addDefaultArgument("-lt lms1xx");

argParser.addDefaultArgument("-lpt tcp");

argParser.addDefaultArgument("-lp 192.168.0.1");

argParser.addDefaultArgument("-connectLaser");

///////////////////////////////////////////////////

if(!robotConnector.connectRobot()) //Connect to robot and troubleshoot if

error such as wrong COM port.

{

ArLog::log(ArLog::Terse, "Could not connect to the robot.");

if(argParser.checkHelpAndWarnUnparsed())

{

// -help not given, just exit.

Aria::logOptions();

Aria::exit(1);

}

}

robot.runAsync(true);

ArKeyHandler keyHandler; //Keyhandler added to allow the escape

key to break the program

Aria::setKeyHandler(&keyHandler);

robot.attachKeyHandler(&keyHandler);

//////////////////////SONAR///////////////////////////////////////////

ArSonarDevice sonar;

robot.addRangeDevice(&sonar); //Sonar device added

//////////////////////////////////////////////////////////////////////

/////////////////////LASER/////////////////////////////////////////////

while (!laserConnector.connectLasers()) //Wait for LASER to connect

indefinitely

{

ArUtil::sleep(100);

}

ArUtil::sleep(2000); //It is best to sleep here to make sure laser

gives the first laser reading correctly.

puts("Laser is Connected!\n"); //Retrieving the pointer to the laser,

ArLaser\* laser = (ArLaser\*)(robot.getRangeDeviceList())->front(); //note that the sonar will probably be the first object

in the list if the laser does not connect.

/////////////////////////////////////////////////////////////////////////

////////////////////CAMERA INITIALIZATIONS///////////////////////////////

/////////////////////////////////////////////////////////////////////////

ArSerialConnection con;

ArVCC4 camera(&robot);

camera.init();

con.setPort("COM5"); //set the COM port which the serial cable of the

camera is plugged into

camera.setDeviceConnection(&con);

videoInput VI;

VI.setVerbose(false);//turn off console messages.

int numDevices = VI.listDevices();//Prints out a list of available

devices and returns num of devices found

int device1 = 0; //this could be any deviceID that shows up in listDevices

VI.setupDevice(device1, 640,480,VI\_S\_VIDEO); //setup the first device with

the default settings

VI.setFormat(device1, VI\_NTSC\_M);

//As requested width and height can not always be accomodated make sure to

check the size once the device is setup

int width = VI.getWidth(device1);//480

int height = VI.getHeight(device1);//640

int size = VI.getSize(device1);

printf("Image Dimensions = %i x %i \n",height,width);

IplImage\* src = cvCreateImage(cvSize(width, height), 8, 3);

//Create Image to hold/display camera frame

unsigned char \* img = new unsigned char[VI.getSize(device1)];

//Used to hold image data from camera

cvNamedWindow("Display", 1);

cvWaitKey(1000); //Similar to "sleep" in the OpenCV library

END INITIALIZATIONS

////////////////////////////////////////////////////////////////////////////

robot.comInt(ArCommands::SOUNDTOG, 1); //Sound (1 = on, default)

robot.comInt(ArCommands::SONAR, 1); //Sonar (1 = on, default) (Turn

off to remove noise if not using.)

robot.enableMotors(); //Allow robot to move.

robot.setAbsoluteMaxRotVel(20); //sets maximum rotational velocity to

xx deg/s

robot.setAbsoluteMaxTransVel(250); //sets maximum translational velocity

to xxx mm/s

puts("This program displays laser, sonar, and camera data. \nPressing

Escape will end the program.\n");

double sonarDist;

int laserDist;

double laserAng;

double sonarAng;

while(1)

{

////////////CAMERA////////////////////////

VI.getPixels(device1, img, false); //returns pixels as a BGR/RGB

(depending on device) unsigned char array

src->imageData = (char \*)img;

cvFlip(src, src, 0);

cvShowImage("Display", src);

cvWaitKey(500);

///////////////////////////////////////////

laserDist = laser->currentReadingPolar(-50,50,&laserAng);

//get distance and angle of nearest object between -50 and 50 degrees

sonarDist = sonar.currentReadingPolar(-50,50,&sonarAng);

printf("sonarDist = %6.1f at %3.1f, laserDist = %6.1d at

%3.1f\n", sonarDist, sonarAng, laserDist, laserAng);

}

cvDestroyWindow("Display");

return 0;

}

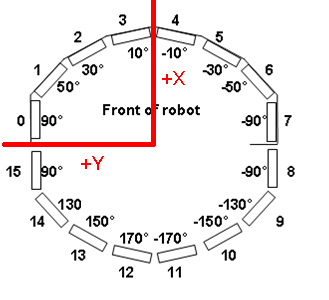
## Sonar

*Mounting the Device*

Unlike the laser, the sonar is a physical part of the robot and therefore does not require a separate USB port to be mounted onto. In order to use the sonar, the program that is using the device must connect to it by a function separate from connecting to the robot. The device also has to be added to the robot.

*Structure*

The sonar device on the P3-DX has 16 discs on our robots with the LMS200 and 8 discs on our robots with the LMS100 with which to collect data. Shown in the figure below is the numbering and angle of each disc. The 8 disc sonar only has discs 0 through 7. The robot considers its own radius to be 250 millimeters using the robot.getRadius command. The actual distance from the center of the robot to a sonar disk is about 180-200 mm. When the robot receives distance to an object using the sonar, it calculates a distance from the center of the robot rather than the actual distance to the sonar disc. Reading data from the sonar is demonstrated in the sample initializations. The individual sonar angles are given below. The robot’s internal coordinate system is also labeled.



Front



Images taken from MobileRobots

## Laser

## *Mounting the Device*

*\*Note – The LMS200 laser has its own power switch located on the back.*

LMS200 Laser:

The LMS200 connects through its own COM port just like the robot. The following is an example of what to type into the command window to run a program. The parameters can also be added within the program as shown in the sample code provided.

example.exe -rp COM4 -lp COM5

Above, -rp and -lp represent the robot port and laser port respectively. The number of the COM port will vary. To determine which COM port is being used, go to Device Manager and check under Ports. Typically, when a device is plugged in, the manager will refresh and the device will show up with its COM port also displayed. If for some reason the COM port is not given, a trial and error of COM ports 1 – 12 might be helpful.

LMS100 laser:

The LMS100 (on the newer robots) connects through the Ethernet port. This will require the Network connections to be modified. This can be done by going to Network Connections and then opening the properties of the Local Area connection. In these properties under the Networking tab, there should be an Internet Protocol Version 4. Under the properties for IPv4, type in the following information:

IP Address: 192.168.0.2

Default Gateway: 192.168.0.1

LMS 100

In older versions of Windows, there may only be a selection called Internet Protocol. This is the same thing and the IP Address and Default Gateway are the same.

Images taken from Sick Laser Manual

Leave the Subnet Mask alone as it will fill itself. Once these settings have been made, the program can be executed and the laser connected to in the following way:

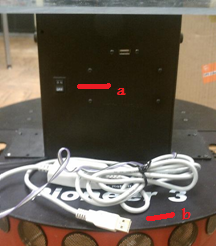
example.exe –rp COM4 -laserType lms1xx -lpt tcp -lp 192.168.0.1

Here, -lpt is laserPortType and –lp is, as mentioned before, the laserPort.

To avoid having to type all of this every time the program is executed, the following shows how to put these parameters into the code directly.

argParser.addDefaultArgument("-lt lms1xx");  
  
argParser.addDefaultArgument("-lpt tcp");  
  
argParser.addDefaultArgument("-lp 192.168.0.1");  
  
argParser.addDefaultArgument("-connectLaser");

* 1. Power switch for the LMS200
  2. Serial to USB cable used to connect the

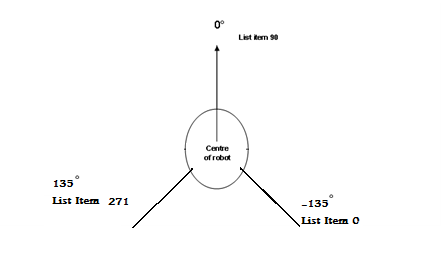
LMS200 to the computer

*Structure*

The LMS200 takes 181 readings over the range of -90 to 90 degrees.



The LMS100 takes 271 readings over the range of -135 to 135 degrees.



A manual for each type of laser will be provided.



## Camera

In order to be able to use the Canon VC-C50i camera PTZ functionality, the videoInput library needs to be installed. The videoInput library is another free C++ library that handles both reading the image data from the camera and controlling the

motors. The following explains how to install the library.

Canon VC-C50i

* Go to http://muonics.net/school/spring05/videoInput/
* Download the latest release “[New - videoInput0.1995 lib and src](http://muonics.net/school/spring05/videoInput/files/videoInput0.1995.zip)” (This will also be made available.)
* Place the file in C:\Program Files

In order to add this library to Visual Studio 2008:

Go to: Tools -> Options -> Projects and Solutions -> VC++ Directories

Select the Platform as Win32.

Select Include Files, Add

C:\ProgramFiles\videoInput0.1995\videoInput0.1995\compiledLib\compiledByVS2008\include

Select Library Files, Add

C:\ProgramFiles\videoInput0.1995\videoInput0.1995\compiledLib\compiledByVS2008

In Project Properties, go to Configuration Properties -> Linker -> Input

Add: videoInput.lib

Install the driver for the audio/video grabber cable:

<http://www.manhattan-products.com/en-US/support/downloads/product/6602-hi-speed-usb-2-0-audio-video-grabber>

Run the Setup from the main folder.



1. Power cable for the camera
2. Camera’s video cable –enables the

Computer to show images/video from the camera

1. Camera’s motor cable –enables

the computer to send commands to move the motors (connects through serial to USB cable)

1.  S-Video to USB converter (red, white, and yellow are not used)

\*Note – The camera requires a 12V power source in order to run. The robots will have a power cord for the camera connected to the internal power source. In the case that the camera is to be used without the robot, there is a separate, external power source which can be used.

Appendix A

*ARIA Commands*

Moving the robot:

* setVel(double  velocity) //sets robot velocity in (mm/sec)
* setRotVel(double  velocity) //sets robot rotational velocity (deg/sec)
* move(double  distance) //moves robot by given distance (mm)
* IsMoveDone() //returns whether the given move command has finished
* setHeading(double  heading) //turns heading from original postion (deg)
* setDeltaHeading(double deltaHeading) //turns heading from current position (deg)
* IsHeadingDone() //returns whether the given turn command has finished
* getX // returns X position of robot based on original position
* getY // returns Y position of robot based on original position
* getTh() // returns orientation of robot based on original position
* moveTo(ArPose(x,y,Th)) //moveTo changes the stored pose, ArPose gives a new pose

Using the joystick:

* setSpeeds(int x, int y, int z=0) //sets the max that X or Y will return
* getButton //gets the button
* getAdjusted // Gets the adjusted reading, as integers, based on the setSpeed

Using the laser:

* getDegDiff(); //gets degrees to turn at which laser will take a new reading
* getDistDiff(); //gets distance at which laser will take a new reading
* setDegDiff(double degDiff); //sets degrees for laser to turn for a new reading (deg)
* setDistDiff(double distDiff); //sets distance for laser to take a new reading (deg)
* currentReadingPolar(Angle1, Angle2, &readingAngle); //between angle1 and angle2, get minimum distance reading and its angle

Using the sonar:

* getNumSonar() //returns the number of sonar sensors
* getSonarReading(i); //get individual sonar data (0 to 15)
* getRange() //distance of the reading
* getSensorTh() //angle of reading
* getSonarRange(i) //physical distance from sonar disc to where the sonar

bounced back

* getRobotRadius()
* currentReadingPolar(Angle1, Angle2, &readingAngle); //between angle1 and angle2, get minimum distance reading and its angle

Others:

* ArUtil::sleep(int milliseconds) //prevents robot from receiving any commands for that time

*OpenCV Commands*

The following are a few useful openCV commands with a brief explanation. The page numbers refer to the pdf “opencv-0\_9\_5”. It is recommended to use this document as it will provide a much more detailed explanation of the function. For those commands that do not have page numbers next to them, a different pdf “Learning OpenCV Computer Vision with the OpenCV library” will need to be searched for more information.

cvCreateImage(CvSize size, int depth, int channels)

* Creates header and allocates data pg(36)

cvNamedWindow(const char\* name, unsigned long flags )

* Creates a window (image placeholder) pg(273)

cvDestroyWindow(const char\* name )

* Destroys the named window pg(273)

cvShowImage(const char\* name, const CvArr\* image )

* Shows the image in the specified window pg(277)

cvReleaseImage(IplImage\*\* image )

* Releases image and header data pg(285)

cvCreateCameraCapture(0)

cvQueryFrame(CvCapture\* capture )

* Grabs and returns a frame from a camera or AVI pg(285)

cvReleaseCapture(CvCapture\*\* capture )

* Release the CvCapture structure pg(285)

cvFlip(const CvArr\* A, CvArr\* B=0, int flip\_mode=0)

* Flip a 2D array around horizontal, vertical or both axes pg(62)

cvSmooth(const CvArr\* src, CvArr\* dst, int smoothtype=CV\_GAUSSIAN,

int param1=3, int param2=0 )

* Smooths the image in one of several ways pg(166)

cvMorphologyEx(const CvArr\* A, CvArr\* C, CvArr\* temp, IplConvKernel\* B,

CvMorphOp op, int iterations )

* Performs advanced morphological transformations pg(165)

cvCvtColor(const CvArr\* src, CvArr\* dst, int code )

* Converts image from one color space to another pg(167)

cvThreshold(const CvArr\* src, CvArr\* dst, double threshold, double maxValue,

int thresholdType )

* Applies fixed level threshold to array elements pg(170)

cvFloodFill(CvArr\* img, CvPoint seed, double newVal, double lo=0, double up=0,

CvConnectedComp\* comp=0, int flags=4, CvArr\* mask=0 );

* Fills a connected component with a given color pg(175)

cvDrawLine(img, edges.front(), edges.front(), cvScalarAll(255));

cvDrawCircle(img, cen=centroid(&edges, &angle), 3, cvScalarAll(255));

cvCreateMat(int rows, int cols, int type )

* Creates a new matrix pg(41)

cvReleaseMat(CvMat\*\* mat )

* Deallocates the matrix pg(42)

cvWaitKey(int delay CV\_DEFAULT(0))

* Waits for a pressed key pg(283)

Appendix B

*Device Driver for RadioShack USB to Serial Cable*

The following guide was taken from <http://www.stenotech.com/pages/Installing-a-Radio-Shack-USB-to-Serial-Adapter.html>.

This is a fix for Vista users or a link to download the XP drivers. Vista will recognize that the cable is there but cannot install the drivers nor download them from the net.

For Windows 7, use the black USB to Serial converter cables, and let Windows 7 search automatically by itself for the driver.

The following instructions are for the white cable driver.  
  
1. Go here <http://www.floridaprobe.com/usb_install.htm> and towards the bottom of the page you will see this "Driver Installation files for Radio Shack users". Download the files.

2. Extract them (it will create a folder called "rshack") and copy the rshack folder to the C:\Windows\inf\ folder if in Vista/7, and to C:\WINDOWS\System32\drivers if in XP. (I am going to assume the C: drive at this point if you use another drive just change the C: to whatever it is) The following is an explanation for Vista, but it can be followed for XP also as long as the file path mentioned above is used.   
  
3. If you have your cable attached go to the control panel.

Select Hardware and Sound.  
Select Device Manager.  
In the other devices section you should see something like USB to Serial, Right click and select Update Driver Software.  
Select Browse my computer for driver software.  
In the dialog box type in the directory you put the files in earlier  
(C:\Windows\inf\rshack)  
Make sure you have a check mark in check subfolders box.  
Click next.  
At this point Vista should find the driver files for you.  
The driver is not digitally signed so you will get a dialog wanting you to confirm installation of the drivers.  
You should be done.

4. If you have not attached your cable.

Complete all steps up to and including step 3.  
Plug in your cable.  
Allow Vista to look for your device...  
When it can't find it select other options and then browse my computer for driver files.  
Point it to the C:\Windows\inf\rshack folder make sure the check subfolders box is selected.  
Click next.  
At this point Vista should find the driver files for you.  
The driver is not digitally signed so you will get a dialog wanting you to confirm installation of the drivers.