CS100R: Assignment 1

Issued: Wednesday 8/29/07 **Due: Friday 9/7/07 by 5PM**

1 Introduction

The purpose of the first assignment is to acquaint you with the basic software that is used in the class as well as to provide you the opportunity to learn about imaging fundamentals and to control the robots. In this assignment, we will assume that you have already read the Matlab Introduction that we provided you.

1.1 CS100R procedure for submitting assignments

In CS100R, you will never be submitting code. Instead, when you are done with an assignment you must demo your functions to a TA. You can demo it at any time before the deadline. However, during the last few hours before the deadline they will only check you off if you make an **appointment** with them in advance. The TAs may ask you to demonstrate that you understand any code printed in this lab, and all code that you have written.

1.2 CS100R procedure for writing code in Matlab

In CS100R, you will write Matlab procedures by editing a skeleton file that we will provide to you. (Such a file is often called a "stub".) For assignment 1, these files will be in the directory `M:\student_files\A1`. The first thing that you should do is to copy the contents of A1 to your `H:\` drive.

2 Thresholding

2.1 Experiment with the thresholding operation

Run `part1_gui`, which will allow you to see the output of the thresholding operation.

(a) Under 'Use Version' select 'Our Version'.

(b) To get an image, you can load one of our examples (Go to Image > Load Image From File and browse to `M:\student_files\A1`), or you may capture your own image (you can go to Image > Load Image from Camera).

(c) Good threshold values cause the wand, and only the wand, to be included in the output.

2.2 Implement thresholding

Double click on the `threshold_student.m` file. You will find the following code in the file:

```
function [ out_img ] = threshold_student(in_img, red_threshold, ...
green_threshold, blue_threshold)
 % This function creates an output image which is thresholded by the three
 % values in red_threshold, green_threshold, and blue_threshold.
```

Recall that this code tells Matlab to create a new function called `threshold_student`. As the comment in the code says, this function returns an array which you have named `out_img`. It takes in four arguments: an input image, and the threshold values for red, green and blue. (The "..." simply tells Matlab that you have a statement that spans multiple lines. If the entire function declaration were on one line, then there would be no need for the "...".) You can tell the `part1_gui` program to execute your code by selecting 'Your Version' from the 'Use Version' dropdown box.

2.3 Writing your code

Your goal is to return a binary image with a black background and a white wand. Use the values for `red_threshold`, `green_threshold`, and `blue_threshold` to distinguish between 'red' and the 'background'.

2.3.1 Useful Matlab functions

`A = zeros(x1,x2,x3...)` creates an array 'A' of zeroes with dimensions x1, x2, x3 etc. `x == y` returns 1 if x has the same value as y, and 0 otherwise. Other operators of the same format include `<`, `>`, `<=`, `>=`.

3 Iteration

We will now implement a function countSameParity.m. Parity is the quality of being even or odd. For part of this function you should use the `mod` function.

```
mod(x,y)
```

For positive x and y, this code will return the remainder of x/y. So if x is 10 and y is 3, `mod(x,y)` will return 1. How might you use this function to determine parity?

Now, recall the following code segment from lecture:

```
nzeros = 0;
[nrows,ncols] = size(B);
for row = 1:nrows
    for col = 1:ncols
        if (B(row,col) == 0)
            nzeros = nzeros + 1;
        end;
    end;
end;
```

This code counted the number of zeros in the matrix `B`.

For this assignment, we will count the number of cells whose cell value has the same parity (the quality of being odd or even) as the sum of the row and column value. For example, if the value 7 is in cell [2,1] (so the sum of the indices is 3), then we want to count this value because both 3 and 7 are odd. If the value 8 was in cell [2,1], we would not count it because 8 is even and 3 is odd. A test run might look something like this:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{pmatrix}$$

countSameParity should count cells [1,2] and [2,2], and return a count of 2.

You will get most of the points on this problem for getting your code to work. The obvious way to do this involves checking parity twice for each cell in the array. We will reserve a small number of points for writing a version of countSameParity that checks parity only once per cell.

You will be implementing this function from scratch.

4 Using the Robots

1. Get a robot from one of the cabinets. Turn the robot on (note that this may involve getting a battery pack). Give the robot about a full minute to boot.
2. Find the CS100R robot picker application on your Desktop and open it.
3. You will be presented with a list of robots. Find your robot's name in the list, and click "connect". Allow another fifteen seconds for the Microsoft Robotics stack to initialize and connect to the robot.
4. You should now be able to use the robot commands in Matlab on your computer.

4.1 Troubleshooting

If your robot isn't working, try "pinging" it. ("ping" is a network utility that checks if a host is connected to a network.) Open a command prompt (Start > Run; type cmd and hit Enter). At the command prompt, type: ping robotname where robotname is the name on your robot. You should see output that's something like

```
>> Reply from 192.168.3.42: bytes=32 time=18ms TTL=245
```
If you're still having problems, contact a TA.

4.2 Basic Robot Commands

We provide you with several commands to make robots do basic movement (more commands will be introduced in later assignments). We have done our best to make the interface simple and easy-to-use. Be very careful while playing with robots. Here is the current set of available Matlab commands to interact with robots:

1. `robotInit()` This sets up the robot control environment in Matlab. You must call this command before calling any of the commands below.

2. `robotDrive(distance)` This makes robot travel specified distance in forward (distance > 0) or backward direction (distance < 0). Distance is measured in feet.

3. `robotTurn(degrees)` This makes the robot turn specified degrees clockwise (degrees > 0) or counter clockwise (degrees < 0).

4. `robotIsDone()` This function will return 1 if the last movement function (A `robotDrive` or a `robotTurn`) that has been issued has finished, or 0 otherwise. This includes the case in which no movement command has been issued.

5 Regular Polygonal Movement

Using the above commands, implement the function robotPolyMove in robotPolyMove.m. robotPolyMove(n) should cause your robot to move clockwise in a regular polygon with n sides of length 2.

You will be implementing this function from scratch.

6 Grading

We will be grading the following functions:

```
threshold_student.m
robotPolyMove.m
countSameParity.m
```

Most of your grade will be directly related to whether or not the functions work. A small portion of your grade will be derived from how elegant your code is, whether or not you have used comments, if the `help` command displays anything useful when used on your functions, and how efficent your code is. Later this week we will release a grading spreadsheet detailing our grading scale.