

CSE 398/498-010

Real-time Image Processing for Autonomous Robot Systems

Laboratory Assignment 2: Convolution, Smoothing and Edge Detection

Report Due Date: 24 Sep 04

A. Objective: In this assignment, we will implement a pair of gradient based edge detectors. The first will be based upon the first-order gradient approximation. The second will be based upon the Sobel kernels discussed in class.



B. The Implementation: Implementation will be consistent with the procedures discussed in class, and will encompass three phases:

- 1) Smoothing out high frequency noise
- 2) Estimating the gradient
- 3) Choosing the proper threshold

To accomplish this, you are to implement the following functions:

```
void myGaussSmoothH(MILImage* pImSrc1, MILImage* pImDest, int h, int w)
```

```
void myGaussSmoothV(MILImage* pImSrc1, MILImage* pImDest, int h, int w)
```

These will convolve the source image with the smoothing kernels $[1\ 4\ 6\ 4\ 1]/16$ and $[1\ 4\ 6\ 4\ 1]^T/16$, respectively.

```
void myGradV(MILImage* pImSrc1, MILImage* pImDest, int h, int w);  
void myGradH(MILImage* pImSrc1, MILImage* pImDest, int h, int w);
```

These will calculate the ABSOLUTE VALUE of the image gradients in the horizontal (I_x) and vertical (I_y) directions, respectively .

```
void myGrad(MILImage* pImSrc1, MILImage* pImSrc2, MILImage* pImDest,  
int h, int w);
```

This will calculate the total image gradient based upon $I = \sqrt{I_x^2 + I_y^2}$

```
void mySobelV(MILImage* pImSrc1, MILImage* pImDest, int h, int w);  
void mySobelH(MILImage* pImSrc1, MILImage* pImDest, int h, int w);
```

These will calculate the ABSOLUTE VALUE of the horizontal and vertical image gradients based upon a convolution with the respective Sobel kernels.

You will also need to make use of the following functions from the first lab:

```
void myAverage(MILImage* pImSrc1, MILImage* pImSrc2, MILImage* pImDest,  
int h, int w);  
void myThresh(MILImage* pImSrc, MILImage* pImDest, int thresh,  
int h, int w);  
void myMedianFilter(MILImage* pIm1, MILImage* pIm2, int h, int w);  
void makeFakeImage(MILImage* pIm1, int h, int w);
```

C. Putting it all together: If you have correctly implemented the above functions, the edge detection can be accomplished by the following process

1) Smoothing High Frequency Noise: This will be done by convolving the original camera image with a gaussian filter. To do this, you will call these 2 functions successively on the camera image. For example, after

```
myGaussSmoothH(pIm1, pIm2, 480, 640);  
myGaussSmoothV(pIm2, pIm3, 480, 640);
```

pIm3 will now contain the smoothed image.

2) Estimating the Gradient:

```
myGradV(pIm1, pIm2, 480, 640);  
myGradH(pIm1, pIm3, 480, 640);  
myGrad(pIm2, pIm3, pIm1, 480, 640);
```

After this series, the magnitude of the image gradient should be in image plm1.

```
mySobelV(pIm1, pIm2, 480, 640);  
mySobelH(pIm1, pIm3, 480, 640);  
myGrad(pIm2, pIm3, pIm1, 480, 640);
```

After this series, the magnitude of the Sobel based image gradient should be in image plm1. Alternately, the below implementation will contain an approximation of the Sobel that is computationally more efficient.

```
mySobelV(pIm1, pIm2, 480, 640);  
mySobelH(pIm1, pIm3, 480, 640);  
myAverage(pIm2, pIm3, pIm1, 480, 640);
```

YOU ARE RESPONSIBLE FOR ALL THREE IMPLEMENTATIONS !!!

3) Choosing the Appropriate Threshold: This value will vary significantly dependent upon which edge detector variant.

Your end result will be to display the filtered, binary image which corresponds to the EDGE SEGMENTATION of the scene.

READ THE REPORT REQUIREMENTS BELOW FOR ADDITIONAL LAB REQUIREMENTS THAT WILL NEED TO BE ADDRESSED.

D. Your Report: You and your partner are to submit a lab report with the results of your implementation. **TAKE THIS SERIOUSLY AS THIS ASSIGNMENT IS ON THE ORDER OF 10% OF YOUR GRADE. I WILL MARK IT SERIOUSLY.** It should include

- Your source code
- Appropriate images as described below
- At a minimum, responses to the following questions

1. For the same image scene, you are to segment edges using the gradient based technique and the full Sobel under the appropriate thresholds. Save a TIFF for each image. Subjectively contrast the performance of both approaches. NOTE: THRESHOLD SO THAT THE EDGES ARE BLACK AND THE BACKGROUND IS WHITE TO SAVE ON PRINTER TONER

2. Repeat 1 for the full Sobel and the approximate Sobel.
3. Run only the 2 Sobel variants (separately) with no displays open and without smoothing. What is the CPU utilization for the 2 alternatives?
4. What range of threshold values did you find were useful for segmenting edges. Did they vary by approach? If so, hypothesize as to why.
5. What are the effects of running the median filter on the segmented image? Was this useful in segmenting edges from the background?
6. What is the effect of running the smoothing filters consecutively on the original image? Explain what is happening here MATHEMATICALLY.
7. Why does smoothing have the effect of reducing noise in the image? What types of noise does it eliminate? At what cost?
8. The Sobel kernels are derived from the convolution of a smoothing kernel with the gradient kernel. For example:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

As a result, do we really need to do the smoothing if we are going to use the Sobel edge detection? What are your thoughts? Validate/invalidate these by experiment. It is OK to be wrong! Just think!

9. Your ideas for improving the process.

FINAL NOTE: SHOW ME YOUR END RESULT BEFORE LEAVING THE LAB TO VERIFY THAT YOUR IMPLEMENTATION IS CORRECT!