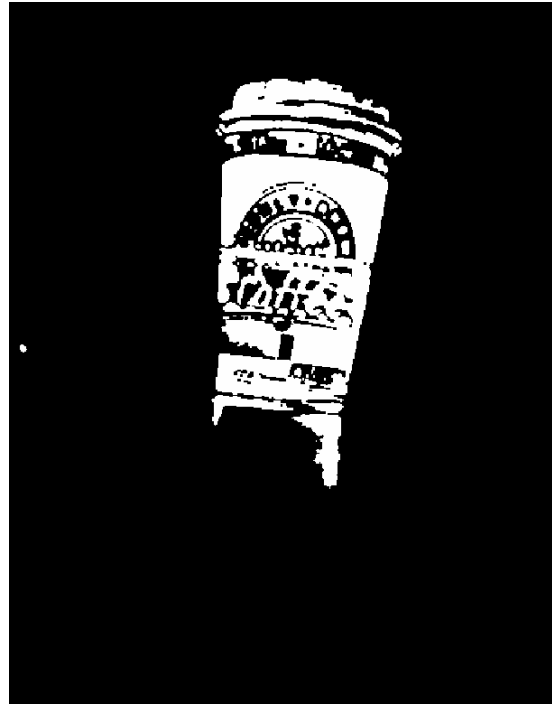


CSE 398/498-010

Real-time Image Processing for Autonomous Robot Systems

Laboratory Assignment 1: Autonomous Surveillance and Monitoring

Report Due Date: 21 Sep 04



A. Objective: The goal of this assignment is to implement the necessary image processing algorithms for segmenting scene changes. Such a capability might be used for intruder detection in an autonomous surveillance and monitoring task

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

- 1) Acquire a template for the scene background
- 2) Perform background subtraction with the template and the current image
- 3) Improved segmentation through median filtering

C. Some preliminaries: We will be using the Apple Isight web cameras in this class. The advantage of these is that they provide perhaps the best optics in the price range. The disadvantages are:

1. They are for a Mac, and we are using a hacked driver
2. They automatically focus (for us, this is a negative as you will see)

3. They automatically adjust the image intensity

We take the good with the bad...

The easiest way to instantiate a new image is to use the camera as the basis

```
MILImage *img1 = new MILImage(cam->_pImage->_params);
```

Remember that you will have to delete this image at the end of the program since it was created dynamically.

If you want the image to display, do:

```
img1->SetDisplay(true);
```

You can get a new image from the camera into img1 with the following call sequence.

```
cam->UpdateImage();  
*img1 = *(cam->_pImage);
```

The pointer to the image data can be found at

```
img1->_pBuffer;
```

Note that you do NOT want to modify this pointer EVER. When you are doing pointer arithmetic, you want to first copy the address to another pointer:

```
unsigned char *p1;  
p1 = img1->_pBuffer;
```

Now you can do whatever you want with p1. Note you can assign a value at an address in memory and then increment the address by one byte with a single statement. For example:

```
*p1++ = 255 ;
```

sets the value at address p1 to 255 and then increments the address by 1. This is nice for grayscale char images, where each pixel is only 1 byte deep.

D. Statement of Work:

You will implement the following functions:

```
void makeFakeImage(MILImage* pIm1, int h, int w)
```

This function will take an image buffer as an argument and put in fake values. This will be used to test subsequent functions for proper behavior.

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

This function calculates the average of 2 images and places it in pImDest. h & w are the height and width of the image

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

Calculates the absolute difference between 2 images, and places the result in pImDest

```
void myThresh(MILImage* pImSrc, MILImage* pImDest, int thresh,
               int h, int w);
```

Performs a threshold operation of the source image, and places the result in pImDest. If the value is \geq thresh, the destination pixel is 255, otherwise 0.

```
void myMedianFilter(MILImage* pIm1, MILImage* pIm2, int h, int w);
```

Performs a median filter on pIm1 with the result in pIm2

Once these are complete, the process then proceeds as follows:

- 1) Acquire a template for the scene background:** You will need to maintain an image of the background. To reduce noise, this should be based upon the average of several image over time.
- 2) Perform background subtraction with the template and the current image:** For each new image, you will calculate the absolute difference of it with your background. Large differences correspond to something in the scene that has changed. You will need to segment the background from scene changes with your threshold operation
- 3) Improved segmentation through median filtering:** There will be spurious pixels that correspond to artifacts in the image. You will minimize these through a median filter on the binary thresholded image.

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

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

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

Grayscale and segmented images from your process. There is a member function to the MILImage class to facilitate this:

```
void SaveTIFF(char * pFilename);
```

At a minimum, the following questions need to be addressed:

1. The number of images that were used for the background. Why? What is your basis for this? Compared to a single image, was there any significant difference in noise? You will need to look at low threshold values to see this.
2. What range of threshold values did you find were useful for segmenting the images. Show your results for different values. Which would you choose and why?
3. Were there any perceivable benefits from median filtering.
4. What was the CPU utilization for your process (from the Windows Task Manager)? You should calculate this with the displays shut down. Is it relatively constant or does it oscillate? Why?
- 5. Combine the background subtraction and thresholding into a single function. What is the new CPU utilization?**
6. Could you have written the threshold function with only a single image as an argument (meaning overwrite the image with its threshold) and still obtain the correct result? How about the median filter? Why or why not?
7. A weakness of this approach is that it is dependent on a single, global threshold operation. Suppose that your background was subjected to NOT INSIGNIFICANT changes in illumination. SPECULATE as to how you would improve your model to accommodate these.
8. What if you needed to accommodate 2 significantly different illumination levels (e.g. day and night, cloudy day and sunshine, etc.). Speculate as to how you might attack such a problem.
8. How might one infer information about target velocity from the image?
9. An alternative approach for background subtraction would be to take the difference of every 2 consecutive images. What might be some of the pros and cons of such an approach?

10. Your ideas for improving the process in terms of robustness, reducing false positives/negatives, etc.

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