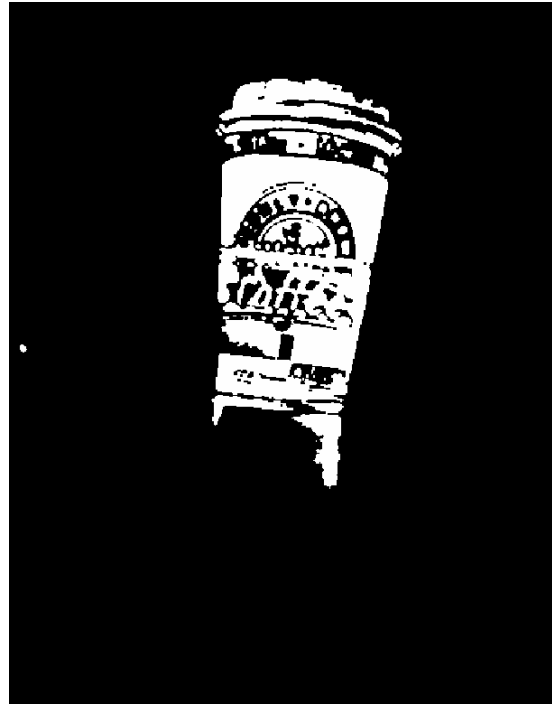


CSE 398/498-010

Real-time Image Processing for Autonomous Robot Systems

Laboratory Assignment 4: Background Subtraction Revisited: Color & OpenCV

Report Due Date: 30 Sep 04



**A. Objectives:**

1. Take advantage of color information to improve the foreground segmentation
2. Introduction to OpenCV Library to demonstrate the advantages of SIMD implementations

**B. The Implementation:**

1. The segmentation problem will be consistent with the first laboratory. However, our source images will be YUV16 instead of grayscale. That is:
  - a. Acquire a YUV template for the scene background
  - b. Perform background subtraction with the YUV image stream and the template image
  - c. Apply appropriate thresholds to each YUV channel.
2. In the second portion of the laboratory, we will repeat the first laboratory (USING MONO IMAGES) while replacing our *myAbsDiff* and *myThreshold* functions with the corresponding OpenCV versions. We will

then benchmark both applications to estimate the speedup from the SIMD extensions

### C. Some preliminaries:

1. First, if you have not done so add the following header files to your main

```
#include <conio.h>
#include <windows.h>
```

The former will allow you to replace your *for* loop with the following *while* loop. You can then stop grabbing by hitting any key (when the console window is active).

```
while(!kbhit()) { // take in a new frame
```

The latter will allow you to use the Windows *Sleep(int milliseconds)* function, which pauses the main thread for *milliseconds*.

2. One of the members of the MILImage class is *\_iplHeader*. This is a pointer to an Intel IPL (Image Processing Library) image. This is the standard argument for Intel OpenCV image processing functions.

### D. Statement of Work:

1. You will implement the following functions:

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

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

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

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

```
void myThreshYUV(MILImage* pImSrc, MILImage* pImDest, int threshY, int
threshU, int threshV, int h, int w);
```

Thresholds the YUV image with different values for each channel. If ANY channel exceeds the threshold, the output is 255, 0 otherwise.

Note that although the U and V channels are subsampled, you are to associate them with their respective color pixels for purposes of thresholding.

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

- a. *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.
- b. *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 for each YUV channel.
- c. *Thresholding the YUV Channels:* You will need to choose a separate threshold for each YUV channel. If any color pixel's exceeds its respective threshold, the new pixel value will be 255, 0 otherwise. Is this an AND or an OR operation?

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

**NOTE: You do not need to apply the median filtering step for the purposes of this approach.**

## 2. First look at OpenCV

- a. Return to your original MONO implementation
- b. Replace the *myAbsDiff* and *myThreshold* functions with *cvAbsDiff* and *cvThreshold* functions. You can find the OpenCV reference manual under OpenCV in your start menu.
- c. Benchmark the two different projects IAW the requirements below

**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:

- Your source code
- Images of a simultaneous MONO and YUV background segmentation WITHOUT MEDIAN FILTERING APPLIED

Also, at a minimum, the following questions need to be addressed:

1. What range of threshold values did you find useful for the YUV segmentation.
2. Subjectively characterize the performance difference between the segmentations. When was it useful? When not?
3. Determine what the CPU utilization is with all displays closed and only the initial image being copied from the camera. This corresponds to the baseline CPU utilization without ANY image processing - just to grab and copy images.
4. What was the *additional* CPU utilization for the MONO and YUV versions? **WARNING:** Make certain that both are grabbing at 15 fps. Do you feel the performance improvement warranted the additional CPU utilization?
5. What was the *additional* CPU utilization for your process on the MONO version when the OpenCV functions were employed? What is the speedup?

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