

## CSE 398/498-010

### Real-time Image Processing for Autonomous Robot Systems

#### Laboratory Assignment 3: Color Segmentation

Report Due Date: 28 Sep 04

**A. Objective:** In this assignment, we will implement a simple YUV space color segmentation algorithm, along with supporting morphological operators



**B. The Implementation:** Implementation will encompass three phases:

- 1) Generate upper and lower bound thresholds for the Y, U, and V channels
- 2) Threshold the source image to a binary image
- 3) Apply morphological operators to enhance the segmentation.

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

```
void makeLUT(int lut[], int min, int max);
```

This function will take a 256 element array *lut*, along with min and max indices. Array elements from *min* to *max* are to be set to 1. All others are to be set to 0

```
void myColorSeg(MILImage* pImSrc1, MILImage* pImDest, int lutY[], int  
lutU[], int lutV[], int h, int w);
```

This will do the actual segmentation. Recall that we are using **YUV16 colorspace!** This means that there is horizontal compression of the U and V chroma channels. Thus, pixels are grouped as

Y1 U1 Y2 V1 Y3 U2 Y4 V2...

and it takes 4 bytes to form 2 YUV16 pixels. In the example above, Y1 and Y2 both use the values of U1 and V1 to form a color pixel.

You will then use these values with the look up tables *lutY*, etc. to segment the image. For example, let us assume that pointer *p1* is pointing to a YUV pixel. Then *lutY[\*p1]* will equal 0 if the pixel value is outside our *min/max* values, and 1 if it is inside.

Segmentation then comes from intersecting the 3 channels. That is, the color has to be in the valid range for the *Y, U, and V* channels!

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

This function operates on a binary image, and uses a 3x3 neighborhood for each pixel. If ANY pixel in that neighborhood is **foreground**, the current pixel is assigned to the foreground. Otherwise, it remains in the background.

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

This function operates on a binary image, and uses a 3x3 neighborhood for each pixel. If ANY pixel in that neighborhood is **background**, the current pixel is assigned to the background. Otherwise, it remains in the foreground.

For example, in the diagram below we are at the \*0\* pixel

```
255 255 255
255 *0* 255
255 255 0
```

For *myDilate*, it will be set to 255 in the new image. For *myErode*, it will be set to 0.

### C. Putting it all together:

1) **Generating look-up tables (LUTs):** To accomplish this, you first need to establish appropriate ranges in Y, U, and V that will segment your object from the background. You can generate these from Matlab using the SaveTIFF function. Alternately you can use trial and error or anything in between. The choice is yours.

2) **Segmenting the Image:** Using the LUTs generated above, perform the appropriate segmentation to a new GRAYSCALE IMAGE. Valid pixels are set to

255, background pixels to 0. Note that the camera has to be declared as YUV and the destination image as MONO.

```
ICamera *cam = ImagingFactory::GetCamera(ImageParameters(640, 480, YUV, 8), 15, false);
```

```
MILImage *img2 = new MILImage(ImageParameters(640, 480, MONO, 8));
```

**WARNING:** The pixel order for some cameras is YUYV YUYV ... where for others it is UYVY UYVY ! You need to figure out what the correct pixel order for your camera is for proper segmentation!

**3) Enhancing the Segmentation:** Using some combination of *myErode* and *myDilate*, attempt to optimize the segmentation of your object from the background.

Your end result will be to display the enhanced binary image which corresponds to the COLOR SEGMENTATION of your object.

**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. Show the original (color) image and the segmented image WITHOUT ENHANCEMENT. What were the threshold values used?

2. QUALITATIVELY (from looking at your image), what is the effect of applying the *myErode* function? What about *myDilate*?

3. Show the segmented and enhanced images (taken at the same time). What combination of erosions and dilations were used? What is the effect?

4. What if instead of erosion & dilation, we used the median filter

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

to improve the segmentation? What is the effect? In which areas is one better than the other?

5. Your ideas for improving the process.

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