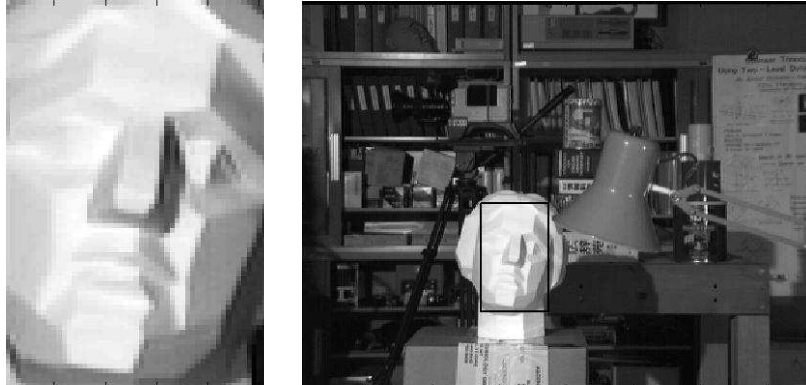


CSE 398/498-010
Real-time Image Processing for Autonomous Robot Systems

Laboratory 6: 2D Pattern Recognition (Part 1)
Report Due Date: 11 Nov 04



A. Objective: Localize 2D patterns in an image using normalized cross-correlation (CC) techniques.

B. Lab Procedure:

1. The end product of this lab is a reliable means for calculating the point of maximum correlation between an image and a template pattern. To verify the correctness of your implementation, you will need an image/template pair with known correlation. To this end, you will save an image file and construct a template from a window of this SAME IMAGE. These will then serve as the test data set.
2. You can have confidence that your implementation is working properly if when you run the CC procedure, it returns the maximum correlation at the same window coordinates from which you cut your template - and the correlation value is equal to 1.
3. Saving an image can be done using the `SaveTIFF` member function. This can then be manipulated in Matlab to construct a template window. I have encountered errors when trying to read in TIFF files. Instead, you should import them as Windows Bitmaps IAW the function call below:

```
MILImage *temp = new MILImage(ImageParameters(w, h, MONO, 8));  
MbufImport("temp.bmp", M_BMP, M_LOAD, M_DEFAULT, &(temp->_milBuffer));
```

After the call to `MbufImport`, the image `temp.bmp` will be placed in the `MILImage` buffer `temp`.

4. Alternately, you can construct the template buffer dynamically each time from within Visual Studio. You should display the template and image to verify the result.
5. Calculating the CC for the entire image is computationally expensive. As a result, a small template size can *initially* be used. Something on the order of 10x10 pixels should be sufficient to demonstrate the correctness of the approach.
6. Once you are confident that the CC procedure is working correctly, increase the template size to 80x60. **Determine how much time is required to calculate the CC for such a template/image pair.**
7. You should use the `DrawRect` member function to identify the image correlation window of maximum correlation. **Save the resulting image.**
8. At a minimum, the following functions will be implemented (NOTE: The prototypes are not cast in stone. Go your own way)

```
void uSigmaIm(MILImage *temp, int h, int w, double& uX, double& sigX);
```

Given an image `temp`, this function will calculate the mean and standard deviation of the image intensity values. These are assigned to `uX` and `sigX` which are passed by reference.

```
double myCrossCorr(MILImage *im, int imParams[], MILImage *temp,  
double tempParams[]);
```

This function will calculate the cross-correlation between a template and a correlation window in the image. The CC is defined as:

$$\rho = \frac{\sum_i [(x_i - \mu_x)(y_i - \mu_y)]}{\sqrt{\sum_i (x_i - \mu_x)^2} \sqrt{\sum_i (y_i - \mu_y)^2}} \equiv \frac{C_{xy}}{\sqrt{C_{xx}} \sqrt{C_{yy}}} \equiv \frac{C_{xy}}{\sigma_x \sigma_y}$$

The `imParams` array will contain the image height and width, along with the coordinates of the image correlation window. The `tempParams` array will contain the template height and width, as well as the mean and standard deviation of the template as calculated by `uSigmaIm`. Note that the template mean and SD only need be calculated once!

This function **MUST** be written to accommodate ANY valid image and template sizes.

C. Report Requirements: A single report will be submitted in conjunction with the Laboratory 7 assignment. Report requirements will be provided for this in the sequel.