HW5: Feature Detection and Matching

Assigned: Tuesday, November 4

Due: Tuesday, November 18

```
void ComputeHarris(CFloatImage &image, CFloatImage &harris)
```

The ComputeHarris function calculates the Harris response matrix (R) for the RGB image. It will come from the formula det(M) – k*trace(M)^2.

`CFloatImage &image`	RGB image input
`CFloatImage &harris`	Harris response matrix (a matrix of size height x width) which you need to fill in.

```
void ComputeHarris(CFloatImage &image, CFloatImage &harris)
```

```
// Convert to grey
CFloatImage greyImage;
ConvertToGrey(image, greyImage);
CShape shape = greyImage.Shape();
```

shape is a struct keeping the height, the width and the number bands of the image.

```
// Compute Harris matrix
harris.ReAllocate(shape);
```

`harris` matrix is allocated with the same size as `greyImage` (which is height x width x 1)

```
harris = greyImage;
```

This is to make the program run, replace this with your code, which should fill in `harris` with the R values.

```
void ExtractDescriptor(int x, int y, CFloatImage &img,
      FeatureSet &features)
```

The ExtractDescriptor method calculates the feature vector for a point from the RGB image and adds this vector to the feature set of the image.

`int x`	x coordinate of the point
`int y`	y coordinate of the point
`CFloatImage &img`	RGB image to which the point (x, y) belongs
`FeatureSet &features`	A data structure that keeps all the features calculated for the image. It is a vector class.

```
void ExtractDescriptor(int x, int y, CFloatImage &img,
       FeatureSet &features)
```

```
Feature f;
f.type = 1;
f.id += 1;
f.x = x;
f.y = y;
```

A new feature for the point is created and necessary fields are populated.
Do not change these.
Each feature f has a data vector of size 243.

```
f.data.resize(1);
```

Set the size of the feature vector.
Yours should be 243.

```
float pix = img.Pixel(x, y, 0);
```

This is how you access the R(ed) value of the pixel at (x, y). [1 is G, 2 is B]

```
f.data[0] = pix;
```

This is how you set the value of the feature vector at position 0.

```
features.push_back(f);
```

Once you fill in the feature vector f, this call adds it to feature set.

```
void dummyComputeFeatures(CFloatImage &image,
       FeatureSet &features)
```

The dummyCompute Features method first calls the ComputeHarris method for the image.

It then finds all the pixels that are local maxima in the harris response matrix calculated by the ComputeHarris method. These are the Harris corner detections.

Finally it calls the ExtractDescriptor method for the detected pixels and fills the feature set for this image.

`CFloatImage &image`	RGB input image.
`FeatureSet &features`	A data structure that keeps all the feature vectors calculated for this image.

```
void dummyComputeFeatures(CFloatImage &image,
       FeatureSet &features)
```

```
// Compute the interest function

CShape shape0 = image.Shape();
CShape shape  = shape0;
shape.nBands  = 1;

CFloatImage harris(shape);

// TODO: write your interest point detector in ComputeHarris()
ComputeHarris(image, harris);
```

The Harris matrix is created with the same dimensions as the input image but with 1 band (channel).

Then your function is called.

```
void dummyComputeFeatures(CFloatImage &image,
       FeatureSet &features)
```

```
~~// Find local maxima in a 3x3 window, and extract descriptors~~


~~CFloatImage greyImage;~~
~~ConvertToGrey(image, greyImage);~~
~~CFloatImage blurImage(shape);~~
~~ConvolveGaussian(greyImage, blurImage, 2.0f);~~
```

You don’t need this. You are going to use the RGB image to calculate the feature descriptors, not the gray-tone image.

```
void dummyComputeFeatures(CFloatImage &image,
      FeatureSet &features)
```

```
int border = 20;


for (int x = border; x < shape.width - border; x++){
      for (int y = border; y < shape.height - border; y++){
…
            if (pix > pix0)
                  isMax = false;

…
```

You can change this to pix >= pix0 to get 1 point as the local maxima

OR

Add a condition to check if pix0 is higher than threshold

```
void dummyComputeFeatures(CFloatImage &image,
        FeatureSet &features)
```

```
if (isMax)
{

// TODO: Write your feature descriptor in ExtractDescriptor()
ExtractDescriptor(x, y, ~~blurImage~~, features);

} // end if isMax
```

Change blurImage to image, since you are going to calculate features from the RGB image.

Tips

To get the height and width of an image:

```
CShape shape = greyImage.Shape();
int height = shape.height;
int width = shape.width;
int band = shape.nBands;
```

See image.h, line 52 for the declaration of CShape.

Tips

To get and set the pixel at (x, y) in the i’th channel of an image:

```
float pix = harris.Pixel(x, y, i);
harris.Pixel(x, y, i) = val;
```

Remember that gray tone images have 1 channel (band), but RGB images have 3 channels (0,1,2).

We usually store response matrices of detectors as images. That’s why the harris matrix is stored as an image. (Remember the Hough transform homework.)

Tips

- You may change the skeleton code as much as you want depending on your implementation.

- In order to implement the extra credit parts, you need to add your own functions and change the `matchFeatures` and `computeFeatures` functions to call your new functions as the second/third cases.