Computer Vision (COMP3204)

Hybrid Image Creation

Anish Katariya (Student ID: 27561879)

Introduction

In our second coursework for our module Computer Vision we were given the task of making Hybrid Images as presented by Olivia, Torralba & Schyns [2006][2] . According to their paper an image can have to different perceptions as a function of the distance they are being viewed from. This can be achieved by "superimposing images at two different spatial scales". In this course work this was achieved by adding the low-pass filtered version of one image to the high – passed filtered version of another aligned image.

Convolution

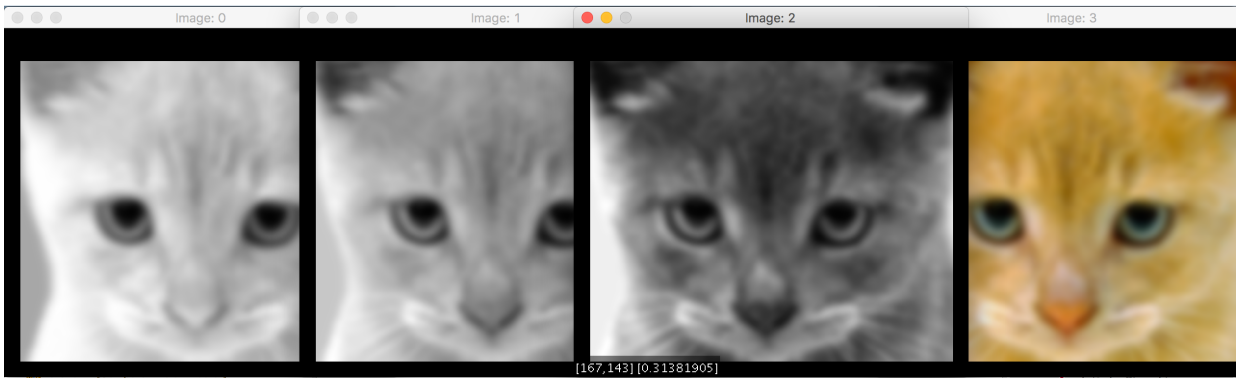

Fig 1 : Calling MyConvolution process on cat.bmp Image. This shows the convolution applied across each band of the image.

For the coursework the method used for achieving a law-pass filter version of the image was by convolving it with a 2-D weighted images(Containing Gaussian Values). To perform the convolution function we were asked to right a convolution algorithm from scratch. The algorithm that has been used by me has been adapted from the convolution algorithm give in the book Feature Extraction & Image Processing[3]. The adapted version of the algorithm is given below.

```
@Override
public void processImage(FImage image) {
    //Getting image dimensions
    int imageRows = image.height;
    int imageCols = image.width;

    //Getting template dimensions
    int templateRows = kernel.length;
    int templateCols = kernel[0].length;

    //Creating a temporary image to manipulate
    FImage tempImage = image.newInstance(image.width, image.height);

    //Calculating half of the templates dimensions and rounding them off.
    int templateRowsHalf = (int)Math.floor(templateRows/2);
    int templateColsHalf = (int)Math.floor(templateCols/2);

    for(int x=templateRowsHalf+1;x<(imageCols-templateRowsHalf);x++) { //Manipulating all columns except the border
        for(int y=templateCols+1;y<(imageRows-templateColsHalf);y++) { //Manipulating all rows except the border
            float sum =0;//Setting sum to be 0 initially
            for(int i =1;i<templateRows;i++) { //for every template columns
                for(int j = 1;j<templateCols;j++) { //for every template rows
                    //Finding the pixel value by accumulating sum and multiplying by template value at that point
                    sum = sum + image.getPixel(x+i-templateRowsHalf-1, y+j-templateColsHalf-1) * kernel[j][i];
                }

            }

            tempImage.setPixel(x, y, sum);//setting the pixel at (x,y) as the accumulated sum

        }
    }

    tempImage.normalise();//Normalising the convolved image(pixels in range 0-1)
    //setting the values of the pixels of the image to that of the temporary image
    image.internalAssign(tempImage);

}
```

Fig 2: This shows the implementation of the Convolution function used. Adapted from Feature Extraction & Image Processing

Hybrid Images

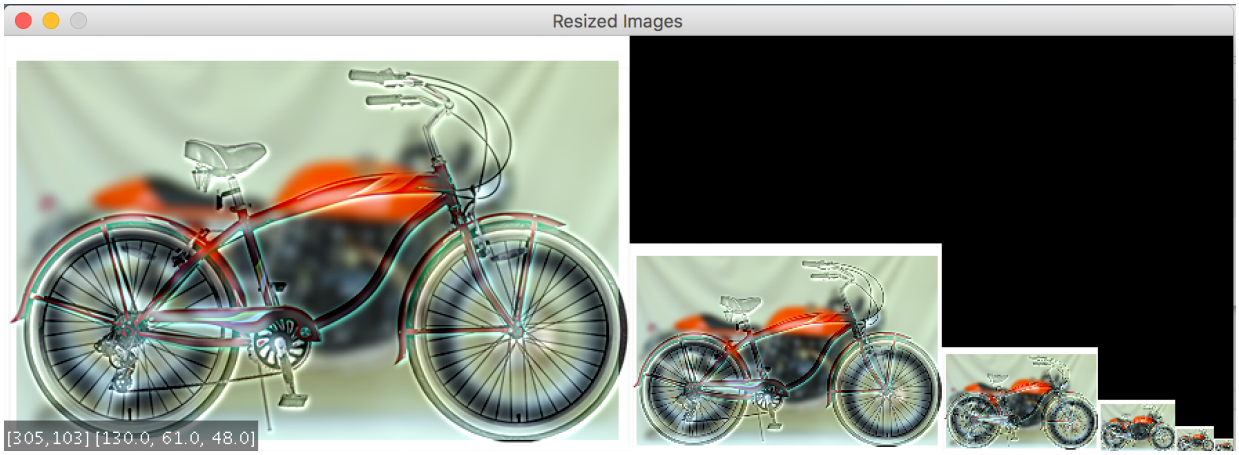

Fig 3 : Shows the hybrid images created using bicycle.bmp and motorcycle.bmp

To produce the hybrid images the images that were to be process were first convolved with a 2 Dimensional Gaussian matrix to get a low pass filtered version of the image. The low pass filter version of the image was then subtracted from the real image to obtain the high pass filter version of the image. Once this was done the low pass filtered version of one image was added to the high pass filtered version of the other image in order to produce two hybrid images. The algorithm used to do the same is given below.

```
public MBFImage[] createHybridImage() {

    //getting size as a function of sigma value
    size = (int) (8.0f * cutOff1 + 0.1f);//window is +/- 4 sigmas from the centre of the Gaussian
    if(size % 2==0) {
        size+=1;
    }


    //Defining the 2 gaussian 2d templates to convolve with images to create lowpass images
    FImage gaussian1 = Gaussian2D.createKernelImage(size, cutOff1);
    FImage gaussian2 = Gaussian2D.createKernelImage(size,cutOff2);


    //creating 2 lowpass images by convolving with 2d gaussian matrices
    lowPass1=image1.process(new MyConvolution(gaussian1.pixels));
    lowPass2= image2.process(new MyConvolution(gaussian2.pixels));

    //creating highpass images by subtracting lowpass band from them
    highPass1=image1.subtract(lowPass1);
    highPass2=image2.subtract(lowPass2);

    //creating 2 hybrid images by adding the highpass bands of one image to the low pass of the other
    hybridImage1= highPass1.add(lowPass2);
    hybridImage2 = highPass2.add(lowPass1);

    MBFImage [] hybrids = {hybridImage1,hybridImage2};
    return hybrids;
}
```

Fig 4: Method used to create the two hybrid images

Results

Some of the hybrid images I created are given below:

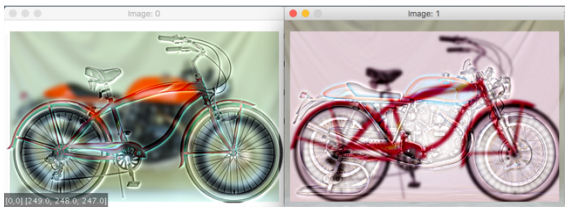

Fig 5: Hybrid between Motor cycle and bicycle

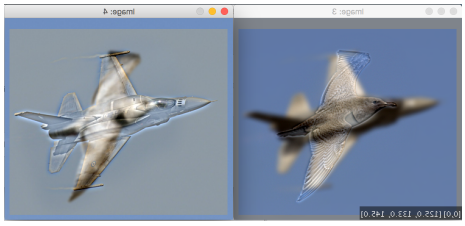

Fig 6: Is it a bird? Is it a plane? It's a hybrid!

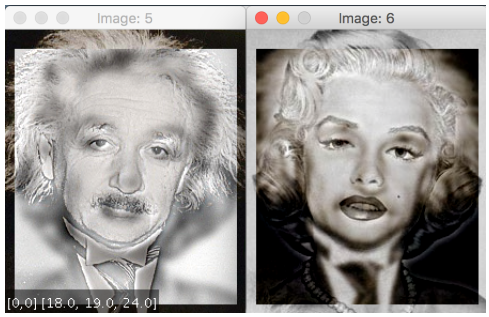

Fig 7: A picture I like to call Marilynstein (Hybrid b/w Marilyn and Einstein)

Fig 8 : High and low pass filter version of cat and dog image.

Discussions:

After implementation, the algorithms are successfully able to produce Hybrids of two images having the same size at a high speed however the hybrid images look the best when the two images have similar alignments and backgrounds. As the template used for convolving the image is smaller than the real image (Due to constrains) the convolved image is smaller than the actual image. Thus, there is a visible border in the hybrid images. The size of the template (and thus the convolved image) is dependent on the sigma values entered by the user.

References:

[1] -The OpenImaj Java Docs , Available from: http://openimaj.org/apidocs/index.html

[2] - Olivia, Torralba & Schyns (2006) SIGGRAPH ACM 2006 Available from : http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf

[3] – Nixon , M . And Aguardo , A. (2012) – Feature Extraction & Image Processing – 3rd Edition Oxford: Elsevier