COMP3204 Computer Vision, Coursework 2: Image Filtering and Hybrid Images

Reuben Ding Chao Ng, rdcn1g14@soton.ac.uk, Electronic Engineering with Artificial Intelligence

I. Introduction

In this coursework, 2 images were filtered and added together to make a hybrid image. By blending a high-pass filtered image with a low-pass filtered image, an image that change in visual interpretation at different distances can be produced. At a closer distance, only the high frequency image can be seen while at longer distance, low frequency part of the image will dominate perception.

Kernel for template convolution and algorithm for blending filtered images were built on the provided skeleton code [1].

II. Kernel Template

A template was use to perform group operation, calculating new pixel values from the pixel's neighbour pixels.The kernel template for the use of filtering image was created by generating a Gaussian kernel 2D array with Gaussian2D.createKernelImage(size, sigma). The size of each dimension of the kernel must be odd number and the sigma, which is the standard deviation of the Gaussian filter controls the cut-off frequency.

```
// Low-pass Convolution
// set Gaussian kernel
int sigma = 4;
// (this implies the window is +/- 4 sigmas from the centre of the Gaussian)
int size = (int) (8.0f * sigma + 1.0f);
if (size % 2 == 0)
    // size must be odd
        size++;

float[][] kernel = Gaussian2D.createKernelImage(size, sigma).pixels;
```

III. Convolution

The algorithm for performing convolutional group operation was built inside processImage, which controls how the kernel template is used to perform convolution on an image.

The half of the kernel for both dimensions were calculated to find the centre of the kernel. Since array starts from 0, the length of the kernel was subtracted by 1 before halving.

```
// half of kernel's length
int halfx = (kernel.length-1)/2;
int halfy = (kernel[0].length-1)/2;
```

Every pixel of the target image will be processed one by one using 2 For loops, one for each dimension of the image.

```
for(int i = 0; i < image.getHeight(); i++)
        for(int j = 0; j < image.getWidth(); j++)
```

To apply template convolution onto a pixel, the pixel needs to be a centre pixel surrounded by neighbour pixels. Convolution will not work if there are not enough pixels around the centre pixel to compute for new pixel value, which means the edges of an image will never be in the centre of the kernel. Since the kernel cannot extend beyond the image, the edges will be painted black, making the new image smaller than original image.

An If condition is used to find if a target pixel is located at the edge of the image. If it is, it will be painted black.

```
if(i < halfx || i >= image.height - halfx || j < halfy || j >= image.width - halfy) {
// set pixel to black
    imageTemp.pixels[i][j] = 0;
```

If the target pixel is not located at the edge of the image, it will be used as the centre of the kernel template to start applying convolution. By going through the kernel row by row, each kernel weight by be multiplied by its corresponding pixel value and then sum together to get the final pixel value. It is then applied onto the new image. Finally, the whole image is then normalised.

```
float sum = 0f;

for(int x = 0; x < kernel.length; x++) {
        for(int y = 0; y < kernel.length; y++) {
// find out coordinates of image pixel in each kernel cell
                int ni = i - halfx + x;
                int nj = j - halfy + y;
// every kernel weight x corresponding pixel value
                sum = sum + kernel[x][y] * image.pixels[ni][nj];
        }
}
imageTemp.pixels[i][j] = sum;
```

IV. Low Pass Filter and High Pass Filter

To produced a low-passed filtered image, high frequencies of the target image can be removed by applying Gaussian filter onto the image with convolution method mentioned above. The standard deviation, sigma of the Gaussian filter can be changed to produce desired effect. By using image.process(new MyConvolution(kernel)) on the dog image, the Gaussian filter kernel will be used to perform convolution on the it through the processImage function and output a filtered image. The sigma used was 4 for the dog image.

```
// load images
MBFImage image = ImageUtilities.readMBF(new File("./data/dog.bmp"));
MBFImage dog = image.process(new MyConvolution(kernel));
DisplayUtilities.display(dog, "Low-passed Dog");
```

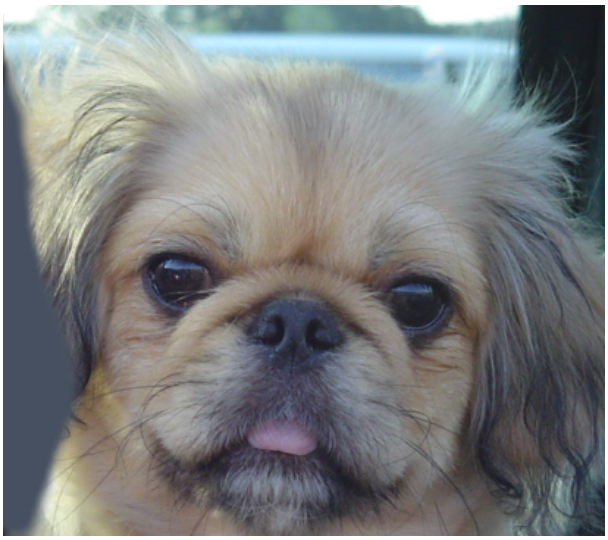

Fig. 1. Original dog image

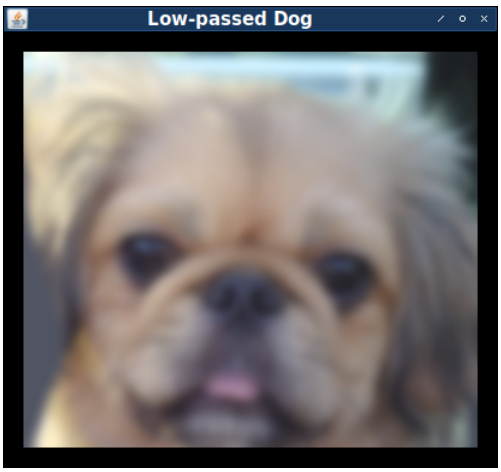

Fig. 2. Low-pass dog image

A high-passed filtered image can be achieved by subtracting a low-passed filtered of an image by the original untempered image. The method was applied to a cat image with a sigma of 8. The resulting high frequency image was very dark in colour due to having zero-mean with negative values, the pixel values of the image was added by 0.5 to each colour channel for visulisation purpose.

```
// load images
image = ImageUtilities.readMBF(new File("./data/cat.bmp"));
MBFImage cat = image.process(new MyConvolution(kernel));
// subtract original cat by low-passed cat
cat = image.subtract(cat);
// add 0.5 to every pixel for visualisation
cat.addInplace(0.5f);
DisplayUtilities.display(cat, "High-passed Cat");
```

Fig. 3. Original cat image

Fig. 4. Low-pass cat image

V. Hybrid Image

The resulting high-pass and low-pass filtered images were sum together to produce a hybrid image.

```
// add both images together
image = dog.add(cat);
image.subtractInplace(0.5f);
DisplayUtilities.display(image, "Hybrid Image");
```

The high frequency feature of the hybrid image, which is a cat, is more visible up close, while the low frequency dog image can only be seen from a far distance. The effect can also be visualised by down-sampling the hybrid image.

```
image = ResizeProcessor.halfSize(image);
DisplayUtilities.display(image, "Half size Image");
image = ResizeProcessor.halfSize(image);
DisplayUtilities.display(image, "Quarter size Image");
```

Fig. 5. Hybrid image

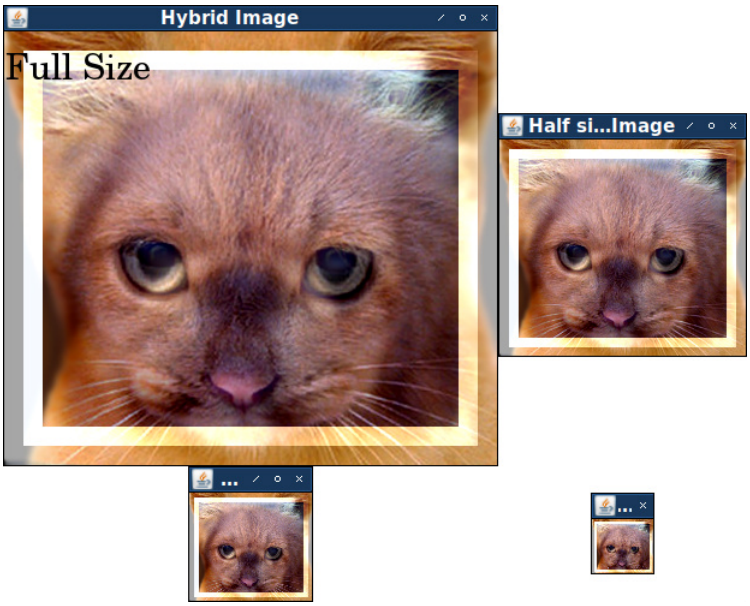

Fig. 6. Hybrid imageat different sizes

REFERENCES

[1] COMP3204 Computer Vision Coursework 2: Image Filtering and Hybrid Images. http://comp3204.ecs.soton.ac.uk/cw/coursework2.html