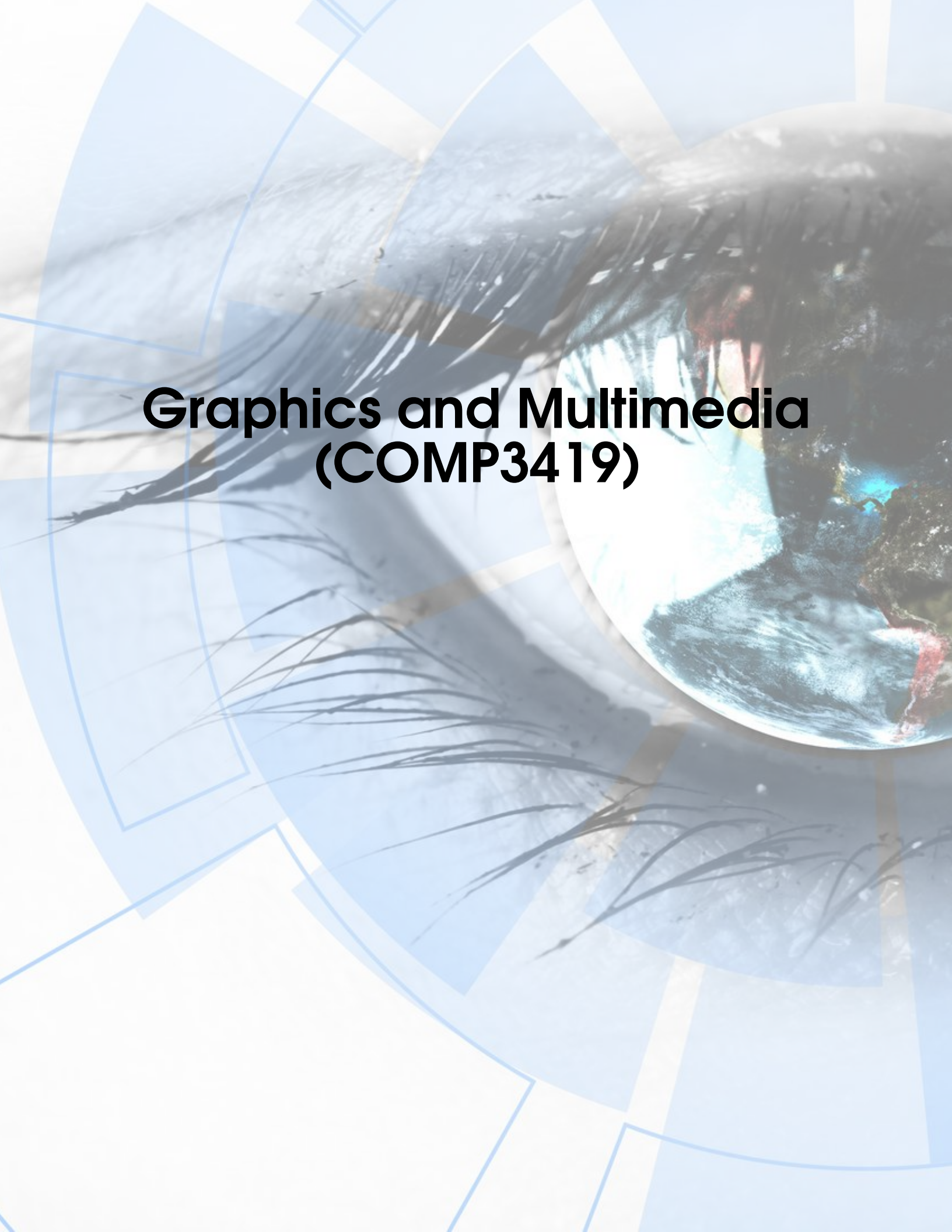
Graphics and Multimedia (COMP3419)



1. Image Enhancement

Welcome to the practical labs of COMP3419. In this lab and the following labs, we will be demonstrating the algorithms and concepts with Processing, which is available at `https://processing.org/`. Processing is a flexible Java syntax based software platform specialised for visual computing first released in 2001. You are also welcome to use your favourite programming languages, for example, Python, or other frameworks for your lab work and assignments as long as you can handle the APIs related to this course.

1.1 Convolutional Filtering

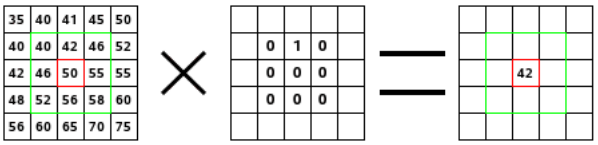


Figure 1.1: A brief illustration of the image convolution operation.

Perform convolution on *opera.png* with the average filter and the sharpening filter respectively. You can find the definitions of these filters in Fig. 1.2. To play with more filters, go to `https://en.wikipedia.org/wiki/Kernel_(image_processing)`. Please compare the effects generated by different filters. When the convolution on the whole image works, try to constrain the convolution within a square area defined by your mouse pointer in real time.

R Decompose the whole convolution problem into easy parts such as the concept of image, the image variable declaration and the algorithm of convolution.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

(a) Average (b) Blur

Figure 1.2: The definitions of two example filters.

R An image is essentially a matrix indicating the red, green and blue (RGB) values at a corresponding location. A pixel is just an element of Image at a particular location. In other words, Image is the collection of Pixels and Pixel is a part of the image.

R During the implementation of convolution in Processing, the *constrain* function may be used to ensure the RGB values are within the range 0-255. This task can also be achieved via *numpy.min* and *numpy.max* in Python.

R Try to write the convolution operation by yourself first. In case you get stuck, a complete example with Processing can be found at `https://processing.org/examples/convolution.html`. With this example, please remember to adjust the canvas size to your input image. For python version, the example can be found at `http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html`.

R A decent demonstration of convolution in general can be found at `http://mathworld.wolfram.com/Convolution.html`.

1.2 Histogram Equalization

Histogram equalization (HE) is a technique for adjusting image intensities to enhance contrast. Refer to the example at `https://en.wikipedia.org/wiki/Histogram_equalization` for a clear demonstration of HE. Please implement HE to enhance the contrast of *parking.jpg* in grey-scale. You can expect effects as shown in Fig. 1.3.

1. To simplify the task, convert the input image to grey scale with $G(x) = 0.21267R + 0.715160G + 0.072169B$. In Processing, if you initialize *color* with only one argument (*color(G)*), it is set as grey-scale by default.
2. Compute the histogram of the input image. It means you need to count the appearances of each grey-scaled value ranging from 0 to 255.
3. Compute the cumulative distribution function (CDF) of the histogram.
4. The meaning of cumulative distribution function can be found at `http://www.itl.nist.gov/div898/handbook/eda/section3/eda362.htm`

Figure 1.3: Example effects of histogram equalisation.

5. Rescale the histogram as $h(v) = round(\frac{CDF(v)-CDF_{min}}{W*H-CDF_{min}} * (L-1))$ where $h(v)$ is the new pixel value for grey level v; W and H are the width and height of the image; L is the total number of grey levels in our image, in our case 256.

After performing HE on the grey-level image, try to perform HE to each colour channel (RGB) separately and visualise your contrast enhanced image.