upmf Grenoble
Université Pierre Mendès-France
Sciences sociales & humaines

Manipulating images
Exercise 2: **Pixel Processing**

CONTEXT

Image processing is a general term for the wide range of techniques that exist for manipulating and modifying images in various ways. In general most-processing techniques involve treating an image as a two-dimensional array of pixels for obtaining either a new image or some kind of measurements (e.g., statistics about colours).[1]

OBJECTIVES

- Learn to manipulate the pixels of an image
- Analyse images though histograms

TO DO & TO HAND IN

The HTML5 canvas element can be used to access the pixels of an image and write image filters that operate over them. The following code illustrates this by defining functions for (i) inverting the colours of an image and (ii) incrementing its brightness.

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">

    var canvas, context, image;

    window.addEventListener('load', init, false);

    function init() {
        canvas  = document.getElementById("canvas");
        context = canvas.getContext("2d");
        image = new Image();
        image.src = "Unknown.jpeg";
        image.addEventListener('load', loadImage, false);
    }

    function loadImage () {
        context.drawImage(image, 0, 0);
    }
```

[1] Wikipedia

```
    function inverseImage() {
      var img = context.getImageData(0, 0, image.width, image.height);
      var pixels = img.data;
      for (var i = 0; i < pixels.length; i += 4) {
        pixels[i]   = 255 - pixels[i  ]; // red
        pixels[i+1] = 255 - pixels[i+1]; // green
        pixels[i+2] = 255 - pixels[i+2]; // blue
      }
      context.putImageData(img, 0,0);
    }

    function incrementBrightness() {
      var img = context.getImageData(0, 0, image.width, image.height);
      var pixels = img.data;
      var brightness = 10;
      for (var i = 0; i < pixels.length; i += 4) {
        pixels[i  ] = brightness + pixels[i  ]; // red
        pixels[i+1] = brightness + pixels[i+1]; // green
        pixels[i+2] = brightness + pixels[i+2]; // blue
      }
      context.putImageData(img, 0,0);
    }
  </script>
</head>
<body>

  <div id="Buttons">
    <button onclick="loadImage()">Reload</button>
    <button onclick="inverseImage();drawHistogram();">Inverse</button>
    <button onclick="incrementBrightness();">Increment Brightness</button>
  </div>

  <div>
    <canvas id="histogram" width="100" height="100">
      Your browser does not support HTML5 Canvas.
    </canvas>
  </div>

  <canvas id="canvas" width="1000" height="1000">
    Your browser does not support HTML5 Canvas.
  </canvas>

</body>
</html>
```

Using this code as basis do the following:

- Define a function that transforms a colour image into a greyscale image. Recall that an RGB image is transformed into a greyscale image by transforming each pixel in RGB using the following equation: [2]

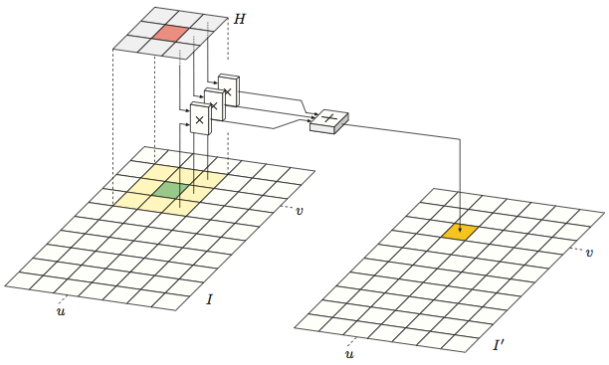
$$Y' = 0.299R' + 0.587G' + 0.114B'$$

- Define a function that transforms a colour image into a monochrome image (i.e. an image that only contains black and white pixels).

- Define the algorithm that compute the histogram of your image and describe the behaviour of the histogram when applying the functions to your image. Recall that a histogram describe the frequency of the intensity values that occur in an image. Assume that in RGB images R, G and B may have up to 256 values.

- Use a histogram library[3] and describe the behaviour of the histogram when you apply your functions to the image.

- Implement a function that "blurred" the image in your canvas. For this simple replace every pixel in your image by the average of its 9 neighbours pixels as illustrated in the following figure.



$$I'(u,v) \leftarrow \frac{p_0 + p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8}{9}$$

[2] http://en.wikipedia.org/wiki/Grayscale
[3] Ex. http://abdiassoftware.com/blog/2013/11/histogramjs-create-fast-histograms-in-javascript/

Javier Espinosa, javier.espinosa@imag.fr
Digital Multimedia