Image processing
Exercise 1

MARC BISGAARD PETERSEN
MARPET18@STUDENT.AAU.DK

Questions:

Anything to do with image manipulation

1. What is the difference between image processing and image analysis
   a. Image processing is when the image is communicating what is on the given image. It could be like the colors, emotions and gesture as a form of information and trying to make a transformation of that image to a new image. Image analysis is when one is trying to find the meaning behind the choices of the selected image.
2. What is computer vision (CV)? Name some application of CV
   a. CV is where we “teach” the computer to see objects like a human does. This includes describe the image, recognize a face or read emotions. Application where the CV are being used could be in self-diving cars, fingerprint recognition or augmented reality
3. What is the visible electromagnetic spectrum?
   a. it is the range of wavelength that is visible to the human eye, and this range from 380 to 740 nanometer or 430-770 thz in terms of frequency
4. Describe the image acquisition process. That is, from light to a digital image in a computer
   a. The light is cast on an object which then reflects into the optical system like a camera. The camera has a sensor with transform the reflection into a digital image.
5. What is a pixel?
   a. A pixel is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device
6. Infrared light has a higher frequency than visual light. True or false?
   a. false, infrared light has a lower frequency than visible light, since the infrared frequency range from 430thz to 300ghz
7. Each pixel can have 30 different values in a 5-bit image. True or false?
   a. False, since bit is counted by 2^n where n is the number of bits. In this case it would be $2^5 = 32$
8. A grayscale image of 9 bits/pixel has how many gray levels?
   a. $2^9 = 512$
9. What is the minimum and maximum values of a pixel in an 8-bit image?
   a. Minimum is 0 and max is 255 due to $2^8 = 256$ and the start value is 0 therefore maximum is 255
10. What is the value of the absolute white color of a grayscale image of 10 bits/pixels?
    a. $2^{10} = 1024$ so the

Programming:

11. Write a sketch in Processing where you read a coloured image and increase the red intensities by 50

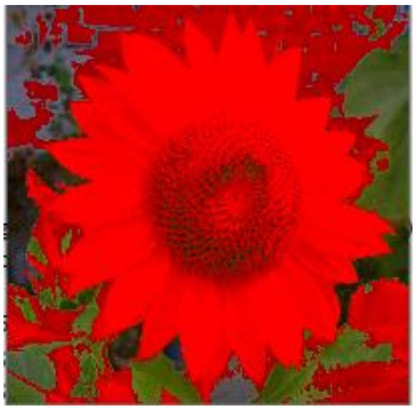

Figure 1: output of code

Code snippet for red intensifier:

```
PImage img;

float colorThresh = 100f; // threshold for the colour

void setup()

{

  size(200,200);

img = loadImage("fuckoff.jpg");  // tells what image should be loaded

 }

void draw()

{

  loadPixels();

  img.loadPixels();

  for(int x =0; x<width;x++)

  {

     for(int y =0; y<height;y++)

     {
      int loc = y+x*width;
      float r = red(img.pixels[loc]); // gets the red value of the selected image
      float g = green(img.pixels[loc]);  // gets the green value of the selected image
      float b = blue(img.pixels[loc]);  // gets the blue value of the selected image
      if(r > colorThresh)
      {
        r += 50; // add value to the pixel’s red
        b = 0; // set value to 0 value to the pixel’s blue
        g = 0; // set value to 0 value to the pixel’s green
      }
      pixels[loc] = color(r,g,b); // sets the colour for selected pixel
     }
  }
  updatePixels();  //updates the pixel
}
```

11. Write a sketch in Processing where you read a coloured image, convert it to grayscale image, apply “salt and pepper” noise first and then apply the Median filter (hint: use the sort() function)
12. **Image:**

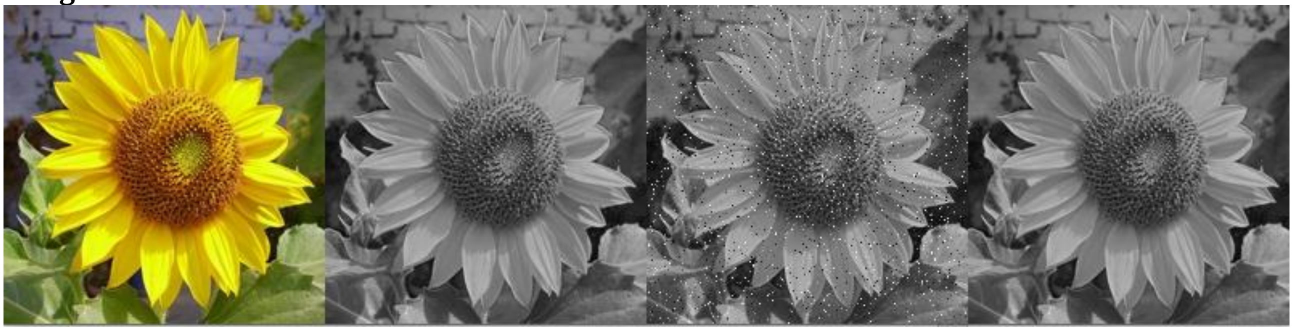

Figure 2: from color to greyscale to salt/pepper noize to median filter

a. Code snippet

```
PImage img,greyScale,saltedImg,medianImg;

float colorThresh = 100f;

void setup()
```

```
{
  size(800,200);
img = loadImage("fuckoff.jpg");
greyScale = createImage(img.width,img.height,RGB);
saltedImg = createImage(img.width,img.height,RGB);
medianImg = createImage(img.width,img.height,RGB);
  greyScale();
  saltPeppernoize();
  median();
}
void greyScale()
{
    img.loadPixels();
  for(int x = 0; x<img.width;x++)
  {
    for(int y = 0; y<img.height;y++)
    {
      int loc = y+x*img.width;
      float r = red(img.pixels[loc]);
      float g = green(img.pixels[loc]);
      float b = blue(img.pixels[loc]);
      float averageColor = (r+g+b)/3;

      r = averageColor;
      g = averageColor;
      b = averageColor;


      greyScale.pixels[loc] = color(r,g,b);
    }
  }
    greyScale.updatePixels();
}
void saltPeppernoize()
{
   img.loadPixels();
  for(int x = 0; x<img.width;x++)
  {
    for(int y = 0; y<img.height;y++)
    {
      int loc = y+x*img.width;
      float r = red(img.pixels[loc]);
      float g = green(img.pixels[loc]);
      float b = blue(img.pixels[loc]);
      float averageColor = (r+g+b)/3;


      r = averageColor;
      g = averageColor;
      b = averageColor;
      saltedImg.pixels[loc] = color(r,g,b);
    }
  }
  int limiter = 2000;
   for(int x = 0; x<limiter;x++)
  {
    int randomheight = (int)random(0,img.height);
    int randomwidth = (int)random(0,img.width);
    saltedImg.set(randomwidth,randomheight,color((int)random(0,255)));
    saltedImg.updatePixels();
  }
  saltedImg.updatePixels();
}
void median ()
{
  remakeGreyScale();
  color[] pixel = new color[9];
  int[] R = new int[9];
  int[] G = new int[9];
  int[] B = new int[9];
  int limiter = pixel.length;
  for(int x =1;x<img.width;x++)
  {
    for(int y =1;y<img.height;y++)
    {
      int pos = x+y*img.width;
      float r = red(greyScale.pixels[pos]);
      float g = green(greyScale.pixels[pos]);
      float b = blue(greyScale.pixels[pos]);
      for(int counter =0;counter<limiter;counter++)
      {
        pixel[counter] = color(r,g,b);
      }
      int newColor =(int)findMedian(pixel,limiter);
      medianImg.pixels[pos] = color(newColor);
    }
  }

}

double findMedian(int[] a,int n)
{
  sort(a);
  if(n%2!=0)
  {
    return(double)a[n/2];
  }
  else
  {
    return (double)(a[n-1]/2+a[n/2]/2);
  }
}
void remakeGreyScale()
{
   saltedImg.loadPixels();
  for(int x = 0; x<saltedImg.width;x++)
  {
    for(int y = 0; y<saltedImg.height;y++)
    {
      int loc = y+x*saltedImg.width;
      float r = red(saltedImg.pixels[loc]);
      float g = green(saltedImg.pixels[loc]);
      float b = blue(saltedImg.pixels[loc]);
      float averageColor = (r+g+b)/3;

      r = averageColor;
      g = averageColor;
      b = averageColor;

      medianImg.pixels[loc] = color(r,g,b);
    }
    }


  medianImg.updatePixels();

}

void draw()

{

  image(img,0,0);

  image(greyScale,200,0);

  image(saltedImg,400,0);

  image(medianImg,600,0);

}
```

13. Write a sketch in Processing where you read a grayscale image and you apply the Sobel filter

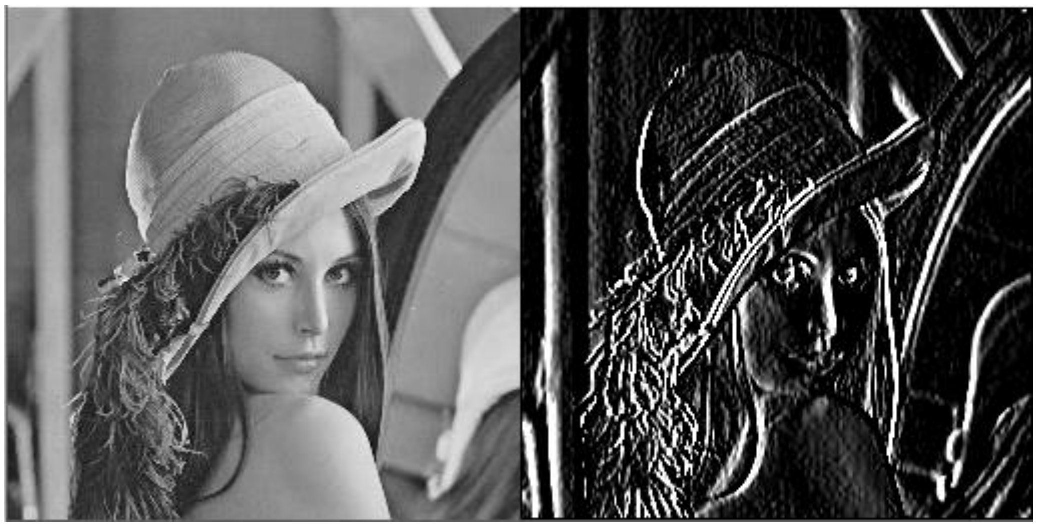

Figure 3 from greyscale to sobel filter

Code snippet:

```
float[][] kernel = {{ -2, 0, 2},

                  { -4, 0, 4},

                  { -2, 0, 2}};
```

```
PImage img;

void setup() {
  size(512, 256);
  img = loadImage("lena.jpg"); // Load the original image
}

void draw() {
  image(img, 0, 0); // Displays the image from point (0,0)
  img.loadPixels();

  // Create an opaque image of the same size as the original
  PImage edgeImg = createImage(img.width, img.height, RGB);

  // Loop through every pixel in the image.
  for (int y = 1; y < img.height-1; y++) { // Skip top and bottom edges
    for (int x = 1; x < img.width-1; x++) { // Skip left and right edges
      float sum = 0; // Kernel sum for this pixel
      for (int ky = -1; ky <= 1; ky++) {
        for (int kx = -1; kx <= 1; kx++) {
          // Calculate the adjacent pixel for this kernel point
          int pos = (y + ky)*img.width + (x + kx);
          // Image is grayscale, red/green/blue are identical
          float val = red(img.pixels[pos]);
          // Multiply adjacent pixels based on the kernel values
          sum += kernel[ky+1][kx+1] * val;
        }
      }
      // For this pixel in the new image, set the gray value
      // based on the sum from the kernel
      edgeImg.pixels[y*img.width + x] = color(sum, sum, sum);
    }
  }
  // State that there are changes to edgeImg.pixels[]
  edgeImg.updatePixels();
  image(edgeImg, width/2, 0); // Draw the new image}
  }
```