# Edge Detection Documentation

Stephan Fourie, 24060984

01/09/2018

# Contents

# 1 Formulas Used

## 1.1 Gaussian

$$G(x,y) = e^{-\frac{(x^2+y^2)}{(2\sigma^2)}}$$

## 1.2 LoG

$$\nabla^2 G(x,y) = \frac{(x^2+y^2-2\sigma^2)}{\sigma^4} \times e^{-\frac{(x^2+y^2)}{(2\sigma^2)}}$$

# 2 General Procedures

The basic initialization of the program is to read in the image. The original images are already in grey scale. The applied are applied with the following convolution code:

```
def convolve(image, mask):
    width = image.shape[0]
    height = image.shape[1]
    mask_s = mask.shape[0]
    mask = np.flipud(np.fliplr(mask))
    img_out = np.zeros(image.shape)
    contstraints = int((mask_s - 1)/2)
    #pad image
    padded_img = np.zeros((height + mask_s - 1, width + mask_s - 1))
    padded_img[contstraints:-contstraints, contstraints:-contstraints] = image

    for x in range(height):
        for y in range(width):
            img_out[y,x] = (mask*padded_img[y:y+mask_s,x:x+mask_s]).sum()

    return img_out
```

## 2.1 Gaussian then Laplacian Procedure

The Gaussian algorithm is used to generate 7x7, 13x13, and 25x25 masks:

```
def gauss_Formula(x, y, sig):
    return -1 * (1/(sig * np.sqrt(2 * np.pi)))
    * (np.exp(-1 * ((np.power(x, 2) + np.power(y, 2)) / (2 * np.power(sig, 2)))))

def gen_GaussMask(maskSize, sig):
    mask = np.zeros((maskSize, maskSize))
    parameter = int((maskSize - 1) / 2);

    for i in range(maskSize):
        for j in range(maskSize):
            mask[i, j] = gauss_Formula(i - parameter, j - parameter, sig)

    return mask / np.sum(mask)
```

The algorithm uses sigma values of 1, 2, and 4 respectively. A 3x3 Laplacian mask is created of the form:

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

The image is first convoluted with the Gaussian mask and then the Laplacian mask.

### 2.2 LoG Procedure

The LoG algorithm is used to generate 7x7, 13x13, and 25x25 masks:

```
def LoG_Formula(x, y, sig):
    return ((np.power(x, 2) + np.power(y, 2) - (2 * np.power(sig, 2))) / np.power(sig, 4))
    * np.exp(-1 * ((np.power(x, 2) + np.power(y, 2)) / (2 * np.power(sig, 2))))

def gen_LoGMask(maskSize, sig):
    mask = np.zeros((maskSize, maskSize))
    parameter = int((maskSize - 1) / 2)

    for i in range(maskSize):
        for j in range(maskSize):
            mask[i, j] = LoG_Formula(i - parameter, j - parameter, sig)

    return mask
```

The algorithm uses sigma values of 1, 2, and 4 respectively. The image is convoluted with the generated LoG mask.

### 2.3 Zero Crossing

The zero crossing is the intersection between the zero intensity and a line extending between the extreme of the second derivative. The algorithm that is used finds the maximum value of the image pixels and times it to a constant value, 0.04, provide by the user to form the threshold. The threshold determines the sensitivity of the filter. A small neighbourhood of the size 3x3 is then used to filter the image in order to detect where zero crossing occur. Different sized masks were tested and 3x3 were found to have the best result. The following algorithm was developed for the zero crossing image:

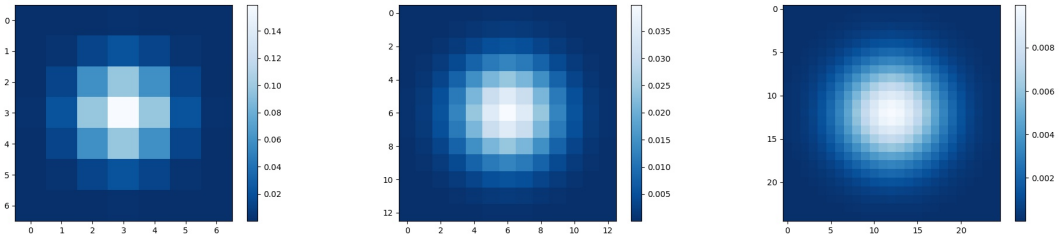
```
def zeroCrossing(img_in, constant):
    width = img_in.shape[0]
    height = img_in.shape[1]
    img_out = np.zeros(img_in.shape)
    thresh = float(np.absolute(img_in).max()) * constant

    for x in range(1, height - 1):
        for y in range(1, width - 1):
            pixel = img_in[x,y]
            pad = img_in[x-1:x+2, y-1:y+2]
```

```
            max_pad = pad.max()
            min_pad = pad.min()

            if np.sign(pixel) > 0 and np.sign(min_pad) < 0:
                zeroCross = True
            elif np.sign(pixel) < 0 and np.sign(max_pad) > 0:
                zeroCross = True
            else:
                zeroCross = False

            if zeroCross and ((max_pad - min_pad) > thresh):
                img_out[x,y] = 1
            else:
                img_out[x,y] = 0

    return img_out
```

# 3 Photos

## 3.1 Masks

### 3.1.1 Gaussian



(a) 7x7 Gaussian Mask (b) 13x13 Gaussian Mask (c) 25x25 Gaussian Mask

Figure 1: Gaussian masks

### 3.1.2 Laplacian

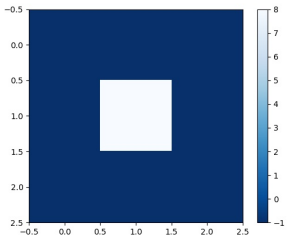
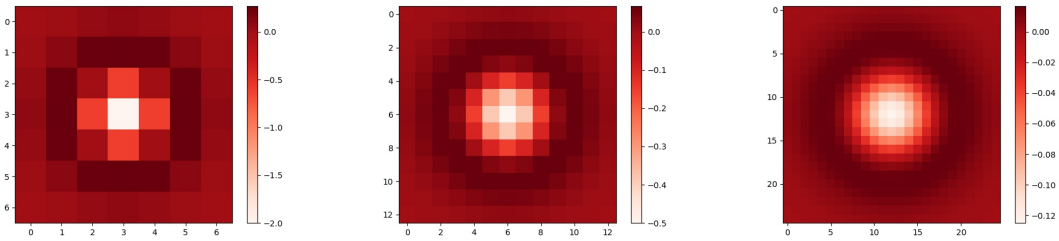


Figure 2: 3x3 Laplacian Mask

### 3.1.3 LoG

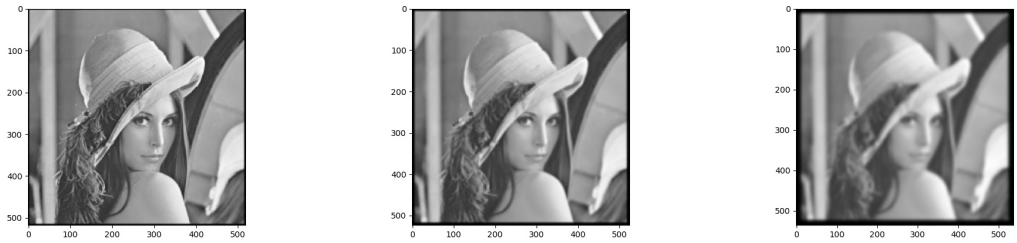


(a) 7x7 LoG Mask (b) 13x13 LoG Mask (c) 25x25 LoG Mask

Figure 3: LoG masks

## 3.2 Lena

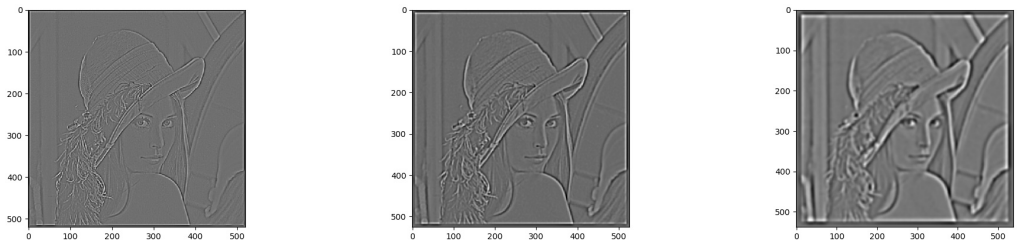
### 3.2.1 Gaussian



(a) 7x7 Gaussian Mask (b) 13x13 Gaussian Mask (c) 25x25 Gaussian Mask

Figure 4: After image has been filtered with respected sized Gaussian mask

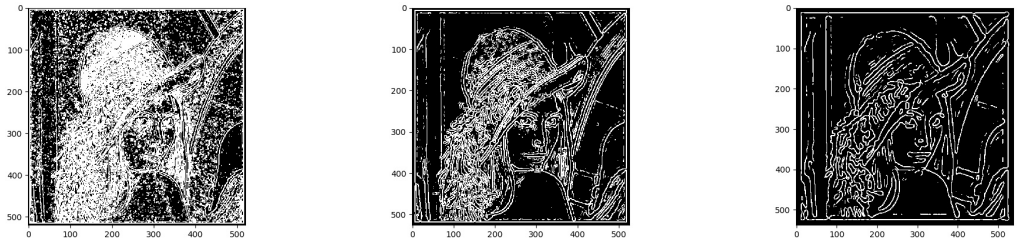
### 3.2.2 Laplacian



(a) 7x7 Gaussian Mask (b) 13x13 Gaussian Mask (c) 25x25 Gaussian Mask

Figure 5: After Gaussian image has been filtered with 3x3 Laplacian mask

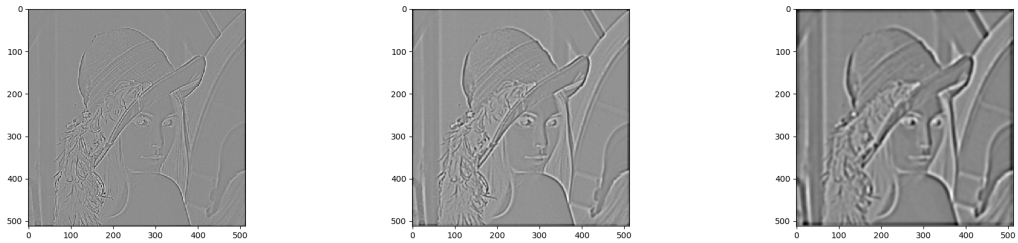
### 3.2.3 Laplacian - Zero Crossing



(a) 7x7 Gaussian Mask (b) 13x13 Gaussian Mask (c) 25x25 Gaussian Mask

Figure 6: After zero crossing has been applied

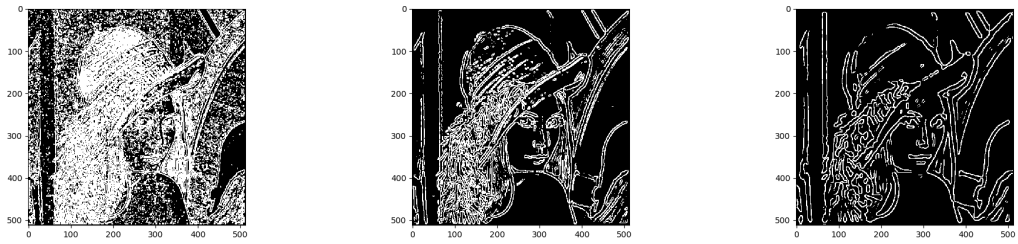
### 3.2.4 LoG



(a) 7x7 LoG mask (b) 13x13 LoG mask (c) 25x25 LoG mask

Figure 7: After convoluting with LoG mask

### 3.2.5 LoG - Zero Crossing

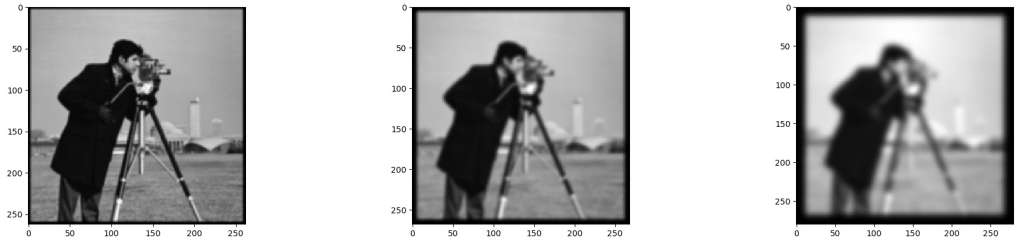


(a) 7x7 LoG mask (b) 13x13 LoG mask (c) 25x25 LoG mask

Figure 8: After zero crossing has been applied

## 3.3 Cameraman

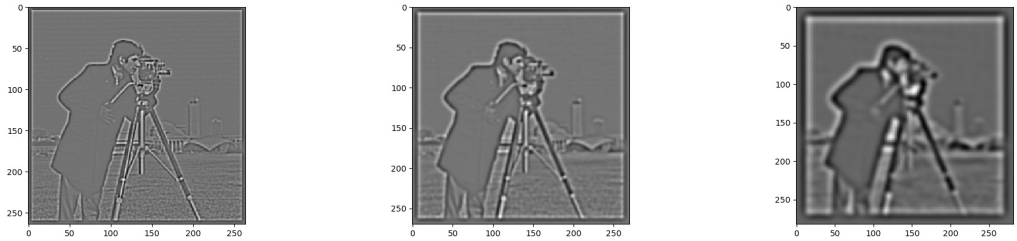
### 3.3.1 Gaussian



(a) 7x7 Gaussian Mask (b) 13x13 Gaussian Mask (c) 25x25 Gaussian Mask

Figure 9: After image has been filtered with respected sized Gaussian mask

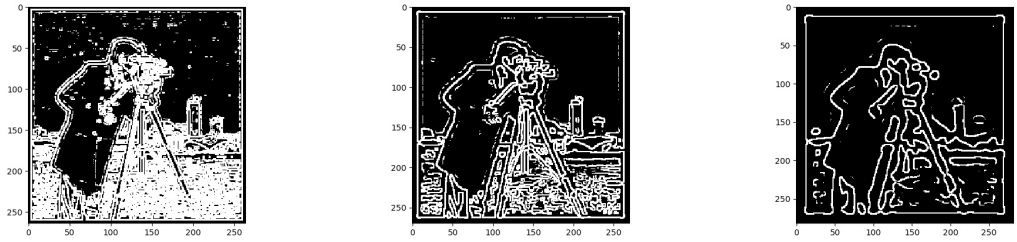
### 3.3.2 Laplacian



(a) 7x7 Gaussian Mask (b) 13x13 Gaussian Mask (c) 25x25 Gaussian Mask

Figure 10: After Gaussian image has been filtered with 3x3 Laplacian mask

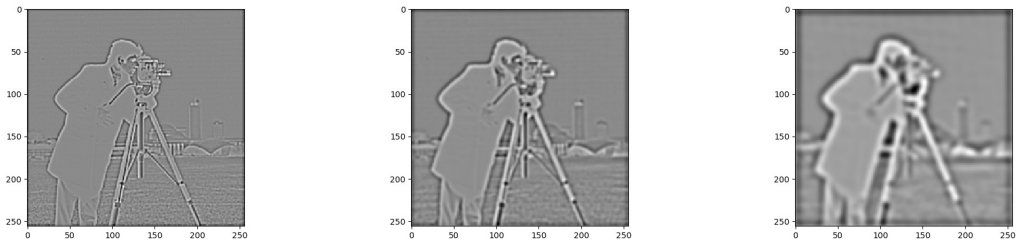
### 3.3.3 Laplacian - Zero Crossing



(a) 7x7 Gaussian Mask (b) 13x13 Gaussian Mask (c) 25x25 Gaussian Mask

Figure 11: After zero crossing has been applied

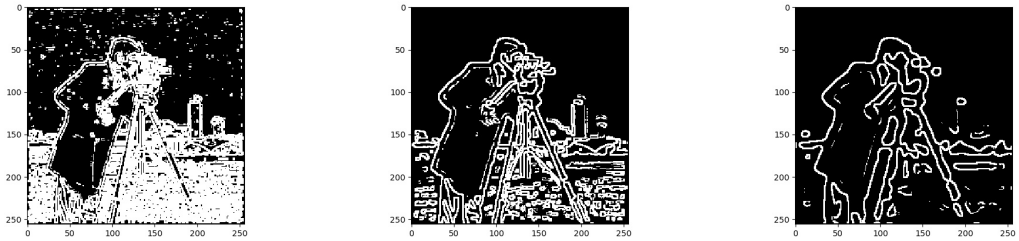
### 3.3.4 LoG



(a) 7x7 LoG mask (b) 13x13 LoG mask (c) 25x25 LoG mask

Figure 12: After convoluting with LoG mask

### 3.3.5 LoG - Zero Crossing



(a) 7x7 LoG mask (b) 13x13 LoG mask (c) 25x25 LoG mask

Figure 13: After zero crossing has been applied

## 4 Conclusion

After comparing the results, little difference were evaluated between the two methods. LoG, however, did detect fewer edges than applying the Gaussian first and then the Laplacian. However, the difference is extremely small.