Image Processing and Computer Vision

REPLICATING ALGORITHM
AN ADAPTIVE ALGORITHM FOR GREY IMAGE EDGE DETECTION BASED ON GREY CORRELATION ANALYSIS

April 8^{th} 2018

Engineering Psychology, Group 881
Electronic Systems, Aalborg University
18g881@es.aau.dk

1 Introduction

The purpose of this mini project in Image Processing and Computer Vision, is to find a article which use a algorithm in some area of computer vision, and try to replicate their results. The chosen article is (**Baoming2016**). The article is focused on optimizing a grey image edge detection algorithm which is based on grey correlation analysis. The optimization is based in making the thresholding of the edge detection adaptive. The goal of the improved algorithm is to get a end result with fewer falls positive edge detected, and better continuity in the detected edges. Figure 1 is a figure from (**Baoming2016**) which shows the result using the optimized algorithm, compared to the original grey correlation analysis algorithm. On Figure 1 the image in the bottom right corner is the result from the improved algorithm, while the other images is a result of different thresholds and operators.

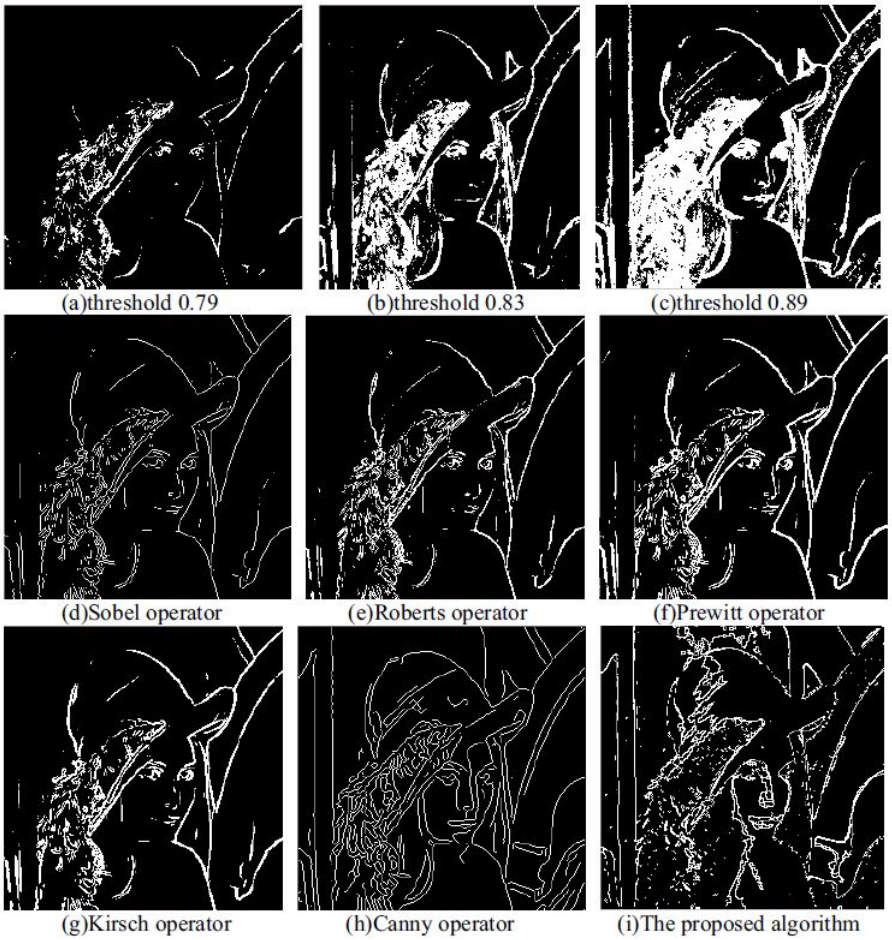

Figure 1

The article is step vise going through the mathematics of the improved algorithm and

trying to explain how the step works. A test of the algorithm is conducted and some of the results from the steps are shown in Figure 2. The step vise changes seen at Figure 2 is also therefore the also the goal for this mini project.

(a)Original image (b)Grey relational image (c)The middle image (d)The result image

Figure 2

2 The algorithm step by step

In this section will be focused on describing the different steps of the algorithm and how the different steps has been transformed into code. The coding is made in MATLAB. All the step are taking in acount that we have a grey scale image of dimensions of m times n.

2.1 Step 1

The first step step is focusing on **Grey Absolute Correlation Degree Model**. The first equation K_0 is the reference sequence and K_i is the comparative sequence. This means that K_0 is the first row of pixels, while K_i is all the pixels.

$$K_0 = K_0(s), s = 1, 2, ..., n \tag{1}$$

$$K_i = K_i(s), s = 1, 2, ..., n, i = 1, 2, ..., m \tag{2}$$

hereafter we initialize the data. This is done by dividing each row with the first pixel of the row. Which is seen in the two equations below.

$$K_0' = \frac{K_0(s)}{K_0(1)}, s = 1, 2, ..., n \tag{3}$$

$$K_i' = \frac{K_i(s)}{K_s(1)}, s = 1, 2, ..., n, i = 1, 2, ..., m \tag{4}$$

Then we calculate the absolute correlation coefficient between K_0' and K_i', which is done with the following equation.

$$r(K_0'(s), K_i'(s)) = \frac{1}{1 + |(K_0'(s+1) - K_0'(s)) - (K_i'(s+1) - K_i'(s))|} \tag{5}$$

After calculation the coefficient, we calculated the absolute degree of it.

$$R(K_0, K_i) = \frac{1}{n-1} \sum_{K} = 1^n - 1r(K_0'(s), k_i'(s)) \tag{6}$$

2.2 Step 2

This step is focused on **Grey correlation degree image**. Results from R are used to make a comparative matrix as shown in the equation below

$$K_i = [I(i,j), I(i,j+1), I(i+1,j+1), I(i+1,j-1), I(i+1,j), I(i-1,j), I(i-1,j+1), I(i,j-1), I(i-1,j-1)] \tag{7}$$

2.3 Step 3

This step is focusing on the **Adaptive Threshold Calculation Based on Human Visual Characteristics**. The purpose of this step is to make the threshold dependent of how sensitive the human eye is to the level of grey scaling. This is based on another study, showing that the human eye is more sensitive to some levels of grey. This results in three differential equations that take the mean value of the 9 pixels described subsection 2.2. Depending on the mean value only one of the following differential equations are used.

$$x\epsilon[0, 48]\frac{d_y}{d_x} - \frac{3y}{x} + 30 = 0 \tag{8}$$

$$x\epsilon(48, 206]\frac{d_y}{d_x} - \frac{3y}{x} + 0.081397x - 3.325108 = 0 \tag{9}$$

$$x\epsilon(206, 255]\frac{d_y}{d_x} - \frac{2y}{x} + 0.003193x^2 - 595.896710 = 0 \tag{10}$$

The values from the the differential equations is the stored in a new matrix T. All values in T which are between 0 and 1 is kept, while for all values greater then 1 are the integer part is removed, so that they also are between 0 and 1. The values are then stored in a final threshold matrix T_1

2.4 Step 4

This step is focusing on **Edge Extraction**. In this step the absolute correlation degree R are compared with the final threshold matrix T_1. For every pixel in R that are smaller then the corresponding pixel in T_1 the value are set to 0, which means it is a edge, otherwise the value are set to 1. This should result in a threshold matrix I_1 whit all edge pixels being 0 and all non edge pixels 1.

2.5 Step 5

This step focus on **Eliminating Isolated Pixel**. This is done by taking the 8 neighboring pixels of a edge pixel and if any of the 8 pixels are a non edge pixel then the edge pixel is set to 1. The new results are saved in a new matrix Z. This is a typical example of erosion. Hereafter we look at the 8 neighboring pixels of the remaining edge pixels in Z. If there is 3 edge pixels in the 8 neighbors then we set all the values to 0. This is a typical example of dilation.

2.6 Transforming the math to code

The following code are the result of how we have implemented the mathematics from subsection 2.1 to subsection 2.5 as MATLAB code. In the code there are some comments describing which step were looking at.

```
1
2  % Import image, and grayscale the image
3  RGB = imread('Lenna1.jpg'); % Insert your own image here
4  I = rgb2gray(RGB);
5
6  % Step 1, Determine the reference sequence and comparative
       sequence
7  I = double(I); %Convert the image into double (decimal data type
       )
8  K0 = I(1,:);
9
10 K1 = I;
11 [m,n]=size(I); % Size of image
12
13 K00=K0/K0(1);
14
15 % Step 2, Data initialization
16 for ii = 1:m
17     for jj = 1:n
18         K11(ii,jj)=K1(ii,jj)/K1(ii,1);
19     end
20 end
21
22 % Step 3, Calculate the absolute correlation coefficient between
       K0 and K1 using the following equation
23 for ii = 1:m-1
24     for jj = 1:n-1
25         r(ii,jj)=1/(1+abs((K00(jj+1)-K00(jj))-(K11(ii,jj+1)-K11(
               ii,jj))));
26     end
27 end
28
29 % Step 4, Calculating the degree of absolute correlation
       coefficient using the following equation:
30 R = (1/(n-1))*sum(r);
31
32 % B, Grey Correlation Degree Image. (Neighborhood mean
       calculation)
33 for ii = 2:m-1
34     for jj = 2:n-1
35         K1b(ii,jj)= (I(ii,jj)+I(ii,jj+1)+I(ii+1,jj+1)+I(ii+1,jj
               -1)+I(ii+1,jj)+I(ii-1,jj)+I(ii-1,jj+1)+I(ii,jj-1)+I(
               ii-1,jj-1))/9;
```

```
    end
end
TempK1BR = R.*K1b-floor(R.*K1b); % To make matrix R, the mean of the starting image was scaled with R.

% C, Adaptive Threshold Calculation Based on Human Visual Characteristics
syms y(x)
cond = y(80) == 0;
ode = diff(y,x)-(3*y/x)+30 == 0;
ySol(x) = dsolve(ode,cond);
ode2 = diff(y,x)-(3*y/x)+0.081397*x-3.325108 == 0;
ySol2(x) = dsolve(ode2,cond);
ode3 = diff(y,x)-(2*y/x)+0.003193*x^2-595.896710 == 0;
ySol3(x) = dsolve(ode3,cond);
for ii = 1:m-1
    for jj = 1:n-1
        if K1b(ii,jj) < 48
            %%1 x?[0,48]
            TempT = ySol(K1b(ii,jj));
            T(ii,jj) = TempT;
        elseif K1b(ii,jj) > 48 && K1b(ii,jj) < 206
            %%2 x?[48,206]
            TempT = ySol2(K1b(ii,jj));
            T(ii,jj) = TempT;
        else
            %%3 x?[206,255]
            TempT = ySol3(K1b(ii,jj));
            T(ii,jj) = TempT;
        end
    end
end
T = double(T); % Tranfrom into a variable that is workable for the rest of the code


% Final threshold matrix
for ii = 1:m-1
    for jj = 1:n-1
        if T(ii,jj) == 1
            T1(ii,jj) = T(ii,jj);
        else
            T1(ii,jj) = T(ii,jj)-floor(T(ii,jj));
```

```
76          end
77      end
78  end
79
80  % D, Edge Extraction
81  % I1 = edge pixel
82  I1 = TempK1BR < T1;
83
84  % E, Eliminating Isolated Pixel
85  % Step 1 Erosion
86  for ii = 2:m-2
87      for jj = 2:n-2
88          TempI= I1(ii,jj)+I1(ii,jj+1)+I1(ii+1,jj+1)+I1(ii+1,jj-1)
                +I1(ii+1,jj)+I1(ii-1,jj)+I1(ii-1,jj+1)+I1(ii,jj-1)+
                I1(ii-1,jj-1);
89          if TempI >= 1
90              Z(ii,jj) = 1;
91          else
92              Z(ii,jj) = 0;
93          end
94      end
95  end
96
97  % Step 2 Dilation
98  for ii = 2:m-3
99      for jj = 2:n-3
100         if Z(ii,jj) == 0
101             TempZ= Z(ii,jj)+Z(ii,jj+1)+Z(ii+1,jj+1)+Z(ii+1,jj-1)
                    +Z(ii+1,jj)+Z(ii-1,jj)+Z(ii-1,jj+1)+Z(ii,jj-1)+Z
                    (ii-1,jj-1);
102             if TempZ <= 3
103                 Z1(ii,jj) = 0;
104                 Z1(ii,jj+1) = 0;
105                 Z1(ii+1,jj+1)= 0;
106                 Z1(ii+1,jj-1)=0;
107                 Z1(ii+1,jj)=0;
108                 Z1(ii-1,jj)=0;
109                 Z1(ii-1,jj+1)=0;
110                 Z1(ii,jj-1)=0;
111                 Z1(ii-1,jj-1)=0;
112             end
113         else
114             Z1(ii,jj) = 1;
```

```
115             end
116         end
117 end
```

3 Results

To test the code we have used a image almost similar to the one used in (**Baoming2016**), the only difference is the size of the image. We have chosen to use a image with a lower resolution, because the program was running wary slow, when the image was larger then 300 x 300 pixels.
On Figure 3 the grey scale image used in this mini project can be seen.On Figure 4 the resulting images from Equation 5 in subsection 2.1 can be seen. On Figure 5 the resulting image of the differential equations in subsection 2.3 , before the thresholding have been done. Hereafter the images is only noise which mostly looks like a QR code. This is a result of one or more errors in the code, which we haven't been able to detect.

Figure 3

Figure 4

Figure 5

4 Conclusion

The results from the original article have not been successfully replicated in this mini-project. The resulting image Figure 4 is somewhat similar to the corresponding image in Figure 2.