

# CSC420 Assignment 1

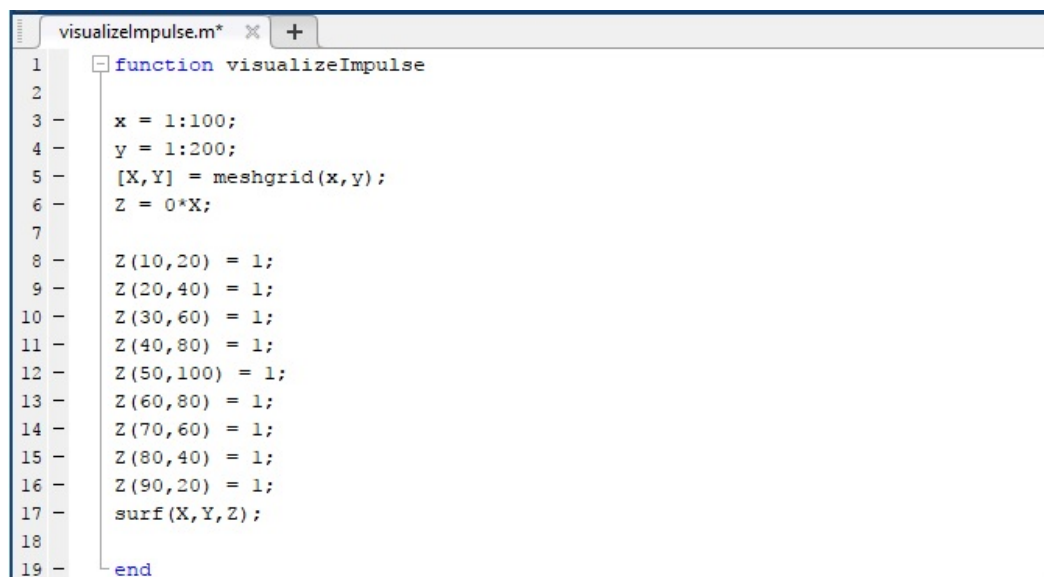
Alex (Kao-Tsun) Chang

September 27, 2017

1

- (a)  $\delta(u, v) = \begin{cases} 1 & \text{when } u = 0 \text{ and } v = 0 \\ 0 & \text{otherwise} \end{cases}$
- (b)  $\delta(u - m, v - n) = \begin{cases} 1 & \text{when } u = 0 \text{ and } v = 0 \\ 0 & \text{otherwise} \end{cases}$

(c)



```
1 function visualizeImpulse
2
3     x = 1:100;
4     y = 1:200;
5     [X,Y] = meshgrid(x,y);
6     Z = 0*X;
7
8     Z(10,20) = 1;
9     Z(20,40) = 1;
10    Z(30,60) = 1;
11    Z(40,80) = 1;
12    Z(50,100) = 1;
13    Z(60,80) = 1;
14    Z(70,60) = 1;
15    Z(80,40) = 1;
16    Z(90,20) = 1;
17    surf(X,Y,Z);
18
19 end
```

Figure 1: Matlab code

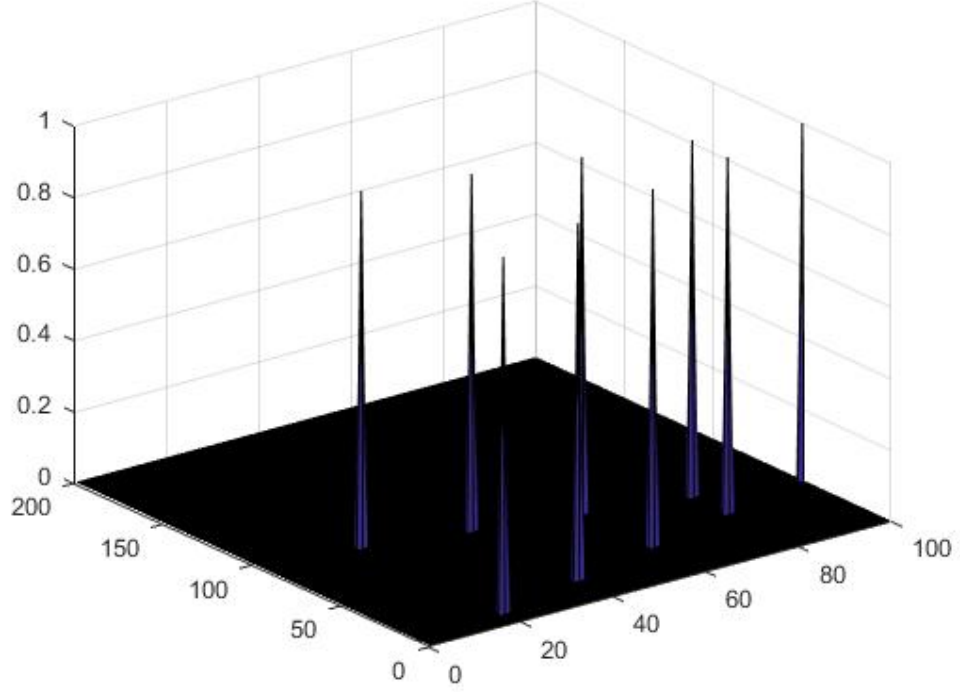


Figure 2: Output

(d)  $f(u, v) = \sum_{i=1}^m \sum_{j=1}^n f(u_i, v_j) \delta(u - u_i, v - v_j)$

## 2

(a) The computational cost of computing convolution is  $O(m^2n^2)$ . This is because there are  $n * n = n^2$  pixels which needs to be multiplied with  $m * m = m^2$  pixels in the filter

(b) If  $h$  is a seperable filter into 2  $m * 1$  sized filters, then the computational cost is  $2m$  multiplied with  $n * n$  pixels, which is  $O(mn^2)$

(c1)  $F_1$  is not seperable because using the matlab function `svd(F1)`, we can see that the output  $S1$  does not only have one singular non-zero value. It must have only one singular non-zero value in the matrix as a requirement to be seperable.

```
Command Window

>> F1

F1 =

    10    40     8
     5     3     5
    12     5    12

>> [U1,S1,V1] = svd(F1);
>> S1

S1 =

  43.6515         0         0
         0  15.1837         0
         0         0   0.0332
```

Figure 3: Matlab code

(c2)  $F_2$  is separable because it has only one singular non-zero value from the output of  $\text{svd}(F_2)$ . The matrix can be written as these 2 separable filters

```
Command Window

>> F2

F2 =

     6     3     6
     2     1     2
     6     3     6

>> [U2,S2,V2] = svd(F2);
>> S2

S2 =

13.0767         0         0
         0    0.0000         0
         0         0    0.0000
```

Figure 4: Matlab code

$$F_a = \begin{bmatrix} 3 & 1 & 3 \\ 2 & 1 & 2 \end{bmatrix}$$

3

The image shows a MATLAB environment with a Command Window on the left and an Editor on the right. The Command Window displays the output of the `readInTemplates()` function, which returns two variables: `templates` and `dimensions`. The `templates` variable is a 3D array of 30 grayscale images, grouped by column ranges and their dimensions. The `dimensions` variable is a 1x30 struct array containing the height and width of each image.

```
>> [templates,dimensions] = readInTemplates();
>> templates

templates =

Columns 1 through 3

    [63x38x3 uint8]    [63x38x3 uint8]    [63x38x3 uint8]

Columns 4 through 6

    [63x38x3 uint8]    [63x38x3 uint8]    [63x38x3 uint8]

Columns 7 through 9

    [63x38x3 uint8]    [63x38x3 uint8]    [63x38x3 uint8]

Columns 10 through 12

    [63x38x3 uint8]    [81x50x3 uint8]    [81x50x3 uint8]

Columns 13 through 15

    [81x50x3 uint8]    [81x50x3 uint8]    [81x50x3 uint8]

Columns 16 through 18

    [81x50x3 uint8]    [81x50x3 uint8]    [81x50x3 uint8]

Columns 19 through 21

    [81x50x3 uint8]    [81x50x3 uint8]    [150x92x3 uint8]

Columns 22 through 24

    [150x92x3 uint8]    [150x92x3 uint8]    [150x92x3 uint8]

Columns 25 through 27

    [150x92x3 uint8]    [150x92x3 uint8]    [150x92x3 uint8]

Columns 28 through 30

    [150x92x3 uint8]    [150x92x3 uint8]    [150x92x3 uint8]

>> dimensions

dimensions =

1x30 struct array with fields:

    height
    width
```

The Editor shows the source code for the `readInTemplates` function:

```
function [templates, dimensions] = readInTemplates
1
2
3     inputFolderRoot = '/h/u9/g6/00/changkao/csc420/assignments/Assignment1/DIGITS';
4     idx = 1 ;
5     for( s = 1 : 3 )
6         inputFolder = fullfile( inputFolderRoot , ['Scale_', num2str(s)] ) ;
7
8     for( i = 0 : 9 )
9         templateFile = [ num2str(i), '.png'];
10
11         templates{idx} = imread( fullfile( inputFolder , templateFile ) ) ;
12         dimensions{idx}.height = size( templates{idx},1) ;
13         dimensions{idx}.width = size( templates{idx},2) ;
14
15         idx = idx + 1 ;
16     end
17 end
```

Figure 5: Code for 3b

```

Editor - /h/u9/g6/00/changkao/csc420/assignments/Assignment1/normcorrA.m
readInTemplates.m drawAndLabelBox.m linearFiltering.m normcorrA.m +
1 function normcorrA(image_url, templates, dimensions)
2
3 % production code
4
5 image = imread(image_url);
6 gray_image = double(rgb2gray(image));
7 [N, M] = size(gray_image);
8
9 gray_template = cell(1,30);
10 correlation_array = zeros(N, M, 30);
11
12 for i = 1 : 30
13
14     tH = dimensions(1,i).height;
15     tW = dimensions(1,i).width;
16     %turn rgb matrix templates into grayscale (take 1st layer)
17     gray_template{1,i} = double(templates{1,i}(:,:,1));
18
19     offSetX_1 = round(tW/2);
20     %offSetX_2 = round(tW/2) + N - 1;
21     offSetX_2 = round(tW/2) + M - 1;
22     offSetY_1 = round(tH/2);
23     %offSetY_2 = round(tH/2) + M - 1;
24     offSetY_2 = round(tH/2) + N - 1;
25     output = normxcorr2(gray_template{1,i}, gray_image);
26
27     correlation_array(:, :, i) = output(offSetY_1:offSetY_2, offSetX_1:offSetX_2);
28
29 end
30
31 [maxCorr, maxIdx] = max(correlation_array,[],3);
32
33 %THRESHOLD
34 %threshold = 0.618;
35 %threshold = 0.616;
36 threshold = 0.6145;
37 %threshold = 0.588;
38
39 candidates = struct;
40 candidates.map = maxCorr > threshold;
41 candidates.value = maxCorr(candidates.map);
42
43
44 %test
45 candidates.maxIdx = maxIdx;
46 %test
47
48 [size_col, size_row] = size(candidates.map);
49 coordinates = find(candidates.map);
50
51 [a, b] = size(coordinates);
52
53 figure;imagesc(gray_image);colormap gray;
54

```

Figure 6: Matlab code part 1

```

55 - for i = 1 : a
56 -
57 -     candidates.candX(i) = floor( (coordinates(i,1)-1) / size_col) + 1;
58 -     candidates.candY(i) = mod(coordinates(i,1), size_col);
59 -
60 -     template_index = maxIdx(candidates.candY(i),candidates.candX(i));
61 -
62 -     thisCorr = correlation_array(:, :, template_index);
63 -     result = isLocalMaximum(candidates.candX(i), candidates.candY(i), thisCorr);
64 -
65 -     if result == 1
66 -         drawnow;
67 -         drawAndLabelBox(candidates.candX(i),candidates.candY(i),template_index , dimensions );
68 -     end
69 - end
70 -
71 - end
72 -
73 -
74 - function result = isLocalMaximum(x, y, thisCorr)
75 -
76 -     thisCorr3 = thisCorr(y-1:y+1,x-1:x+1);
77 -     [M, I] = max(thisCorr3(:));
78 -     [I_row, I_col] = ind2sub(size(thisCorr3),I);
79 -
80 -     if M <= thisCorr(y,x)
81 -         result = 1;
82 -     else
83 -         result = 0;
84 -     end
85 -
86 - end
87 -

```

Figure 7: Matlab code part 2

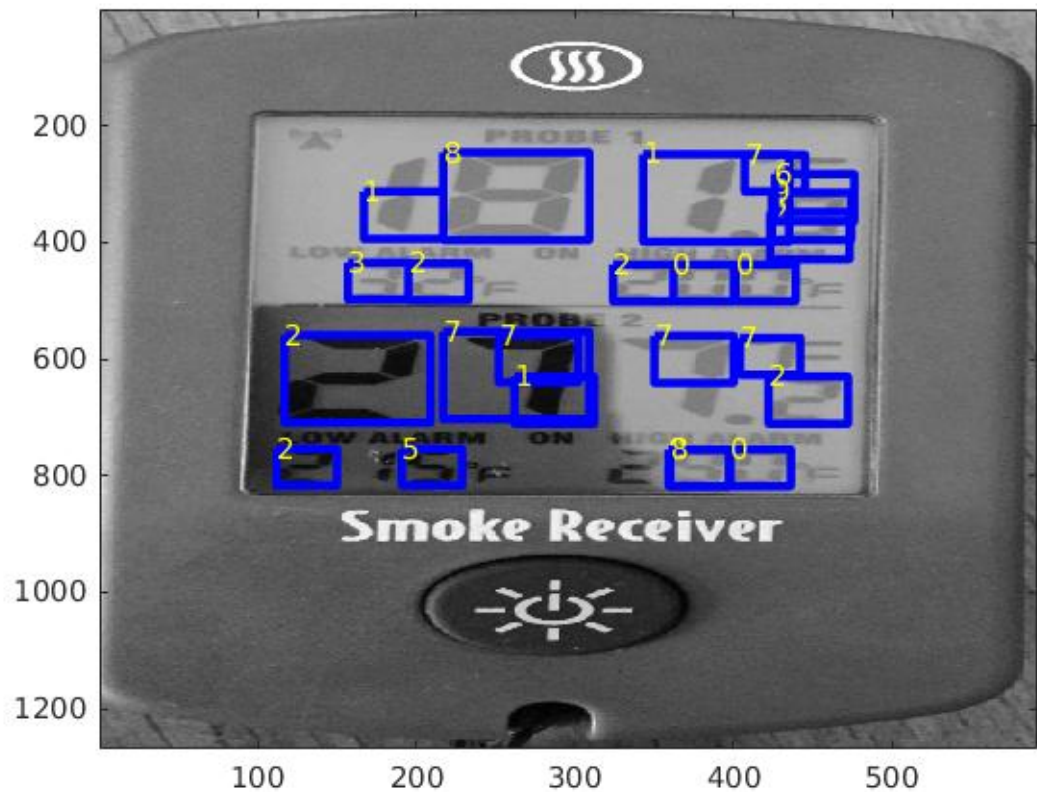


Figure 8: Output with threshold = 0.6145

(3v)

The threshold used was 0.6145, the numbers correctly identified were 1,2,3,7,8. Most of the numbers are correctly identified, only the 1 and 2 in the picture were missed and not matched.

(3vi)

If we crop our own templates from the picture, we would get better matching because the number templates would have the same intensity as the numbers on the actual image. Thus matching would become more accurate.