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)

visualizeImpulse.m*

```
function visualizeImpulse

x = 1:100;
y = 1:200;
[X,Y] = meshgrid(x,y);
Z = 0*X;

Z(10,20) = 1;
Z(20,40) = 1;
Z(30,60) = 1;
Z(40,80) = 1;
Z(50,100) = 1;
Z(60,80) = 1;
Z(70,60) = 1;
Z(80,40) = 1;
Z(90,20) = 1;
surf(X,Y,Z);

end
```

Figure 1: Matlab code

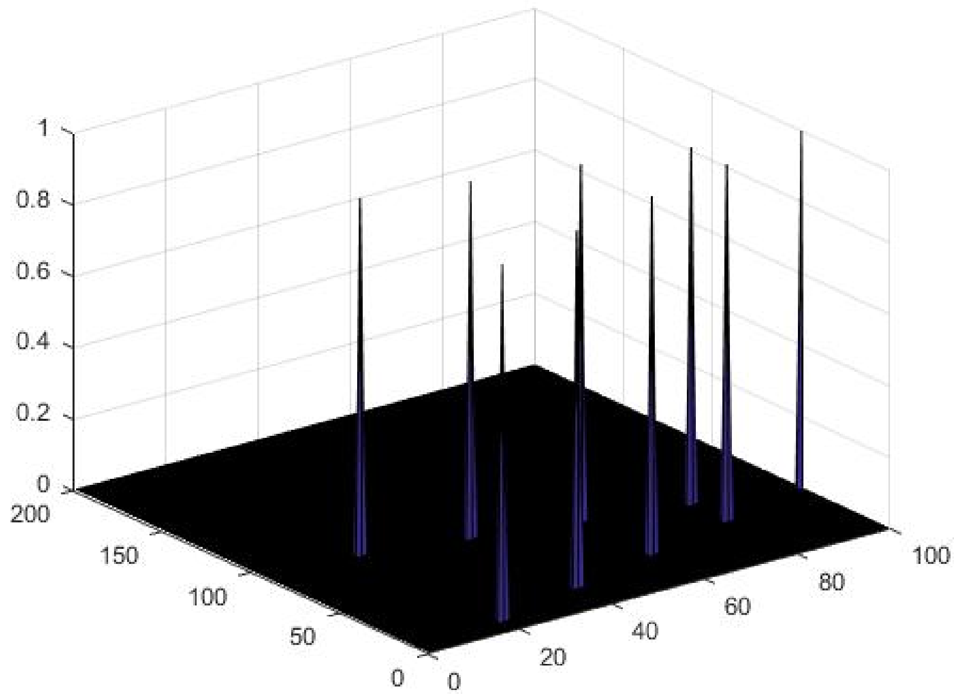


Figure 2: Output

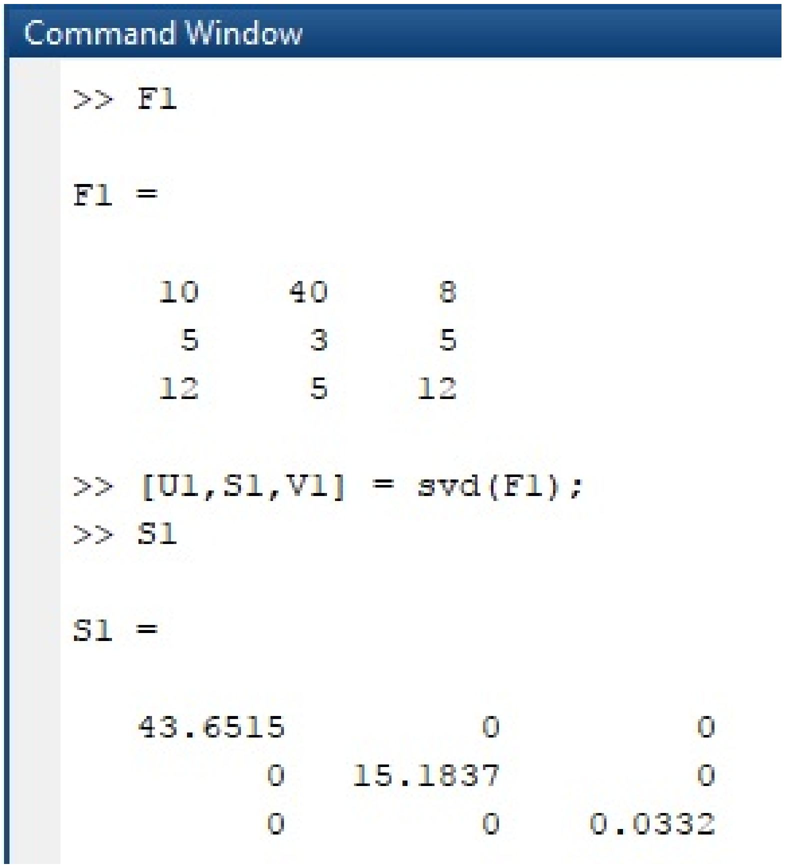
(d) $f(u,v) = \sum_{i=1}^{m} \sum_{j=1}^{n} f(u_i, v_j)\delta(u - u_i, v - v_j)$

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 seperable because it has only one singular non-zero value from the output of svd(F2) The matrix can be written as these 2 seperable 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

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

.

3

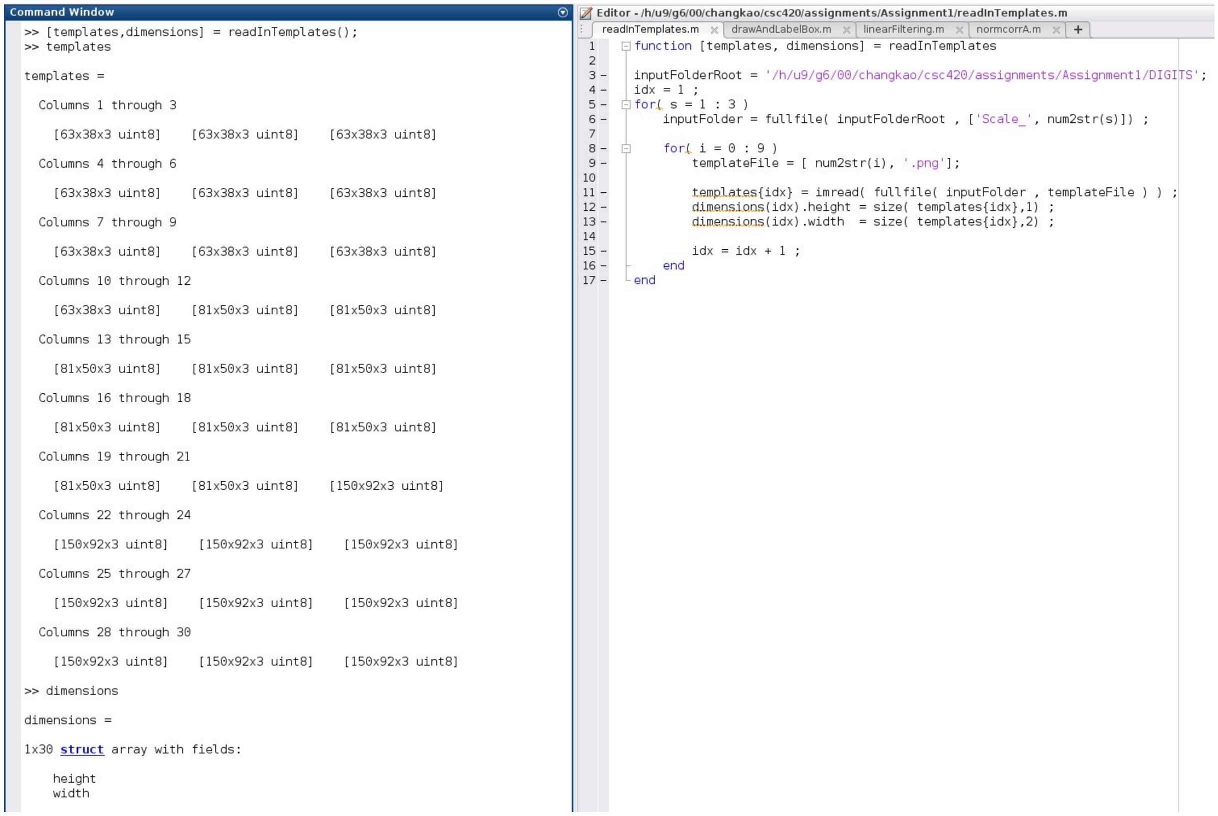


Figure 5: Code for 3b

```matlab
function normcorrA(image_url, templates, dimensions)

% production code

image = imread(image_url);
gray_image = double(rgb2gray(image));
[N, M] = size(gray_image);

gray_template = cell(1,30);
correlation_array = zeros(N, M, 30);

for i = 1 : 30

    tH = dimensions(1,i).height;
    tW = dimensions(1,i).width;
    %turn rgb matrix templates into grayscale (take 1st layer)
    gray_template{1,i} = double(templates{1,i}(:,:,1));

    offSetX_1 = round(tW/2);
    %offSetX_2 = round(tW/2) + N - 1;
    offSetX_2 = round(tW/2) + M - 1;
    offSetY_1 = round(tH/2);
    %offSetY_2 = round(tH/2) + M - 1;
    offSetY_2 = round(tH/2) + N - 1;
    output = normxcorr2(gray_template{1,i}, gray_image);

    correlation_array(:,:,i) = output(offSetY_1:offSetY_2, offSetX_1:offSetX_2);

end

[maxCorr, maxIdx] = max(correlation_array,[],3);

%THRESHOLD
%threshold = 0.618;
%threshold = 0.616;
threshold = 0.6145;
%threshold = 0.588;

candidates = struct;
candidates.map = maxCorr > threshold;
candidates.value = maxCorr(candidates.map);


%test
candidates.maxIdx = maxIdx;
%test

[size_col, size_row] = size(candidates.map);
coordinates = find(candidates.map);

[a, b] = size(coordinates);

figure;imagesc(gray_image);colormap gray;

```

Figure 6: Matlab code part 1

```
for i = 1 : a

    candidates.candX(i) = floor( (coordinates(i,1)-1) / size_col) + 1;
    candidates.candY(i) = mod(coordinates(i,1), size_col);

    template_index = maxIdx(candidates.candY(i),candidates.candX(i));

    thisCorr = correlation_array(:,:,template_index);
    result = isLocalMaximum(candidates.candX(i), candidates.candY(i), thisCorr);

    if result == 1
        drawnow;
        drawAndLabelBox(candidates.candX(i),candidates.candY(i),template_index , dimensions );
    end
end

end


function result = isLocalMaximum(x, y, thisCorr)

    thisCorr3 = thisCorr(y-1:y+1,x-1:x+1);
    [M, I] = max(thisCorr3(:));
    [I_row, I_col] = ind2sub(size(thisCorr3),I);

    if M <= thisCorr(y,x)
        result = 1;
    else
        result = 0;
    end

end
```

Figure 7: Matlab code part 2

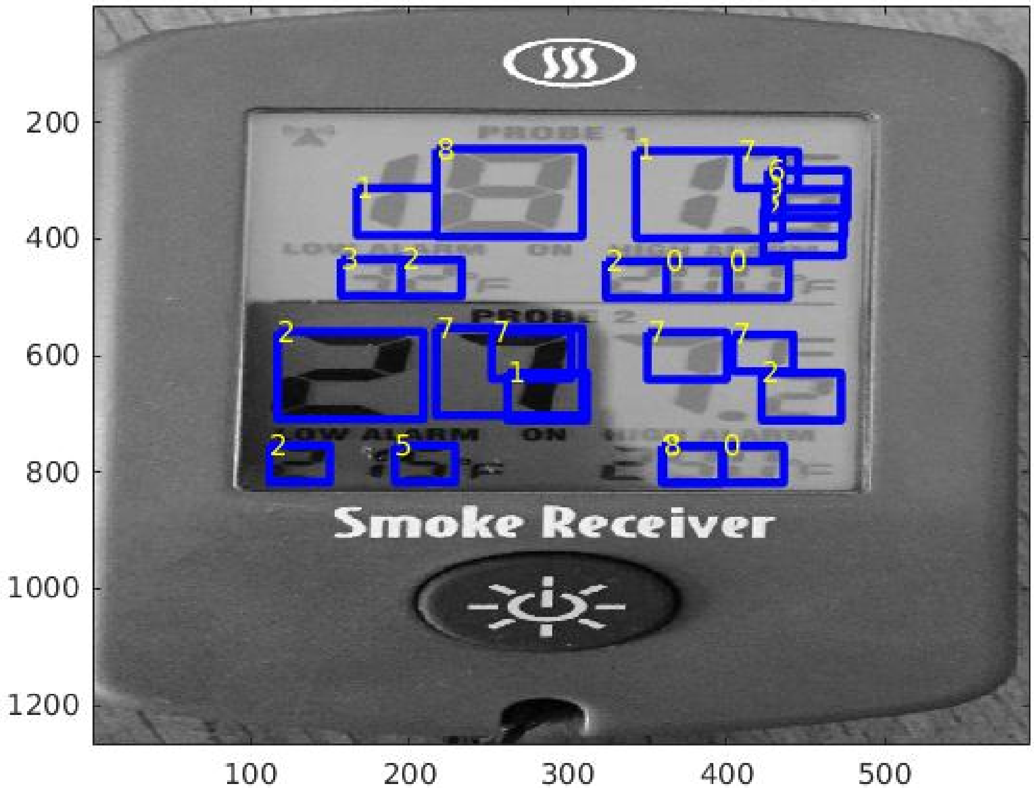


Figure 8: Output with threshold = 0.6145

(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.