Template Matching

By:

Ahmad Nadeem Saigol

Contents

Working of Code: 3

Results: 5

- Images Used: 5
  - Case 1: 5
  - Case 3: 5
- Using Image Processing Toolbox: 5
  - Case 1: 5
  - Case 2: 5
  - Case 3: 7
- Using Custom Function and NCC: 7
  - Case 1: 7
  - Case 2: 8
  - Case 3: 9
- Using Custom Function and ZNCC: 9
  - Case 1: 9
  - Case 2: 9
  - Case 3: 10

Code: 12

Using Correlation for Template Matching

Correlation is one of the tools that is used for extracting information from images. It involves passing a filter over the image while calculating sum of products of region inside the filter to obtain a resulted image. When it comes to template matching, the objective is to find part of the image which is most like the filter. Different kinds of similarity measures can be used for comparing equal sized image regions. However, the more common and the more reliable measures include Normalized Cross Correlation (NCC) and Zero Mean Normalized Cross Correlation (ZNCC). These two have been implemented in MATLAB for matching different templates in an image. The mathematical formulas are available in the following table:

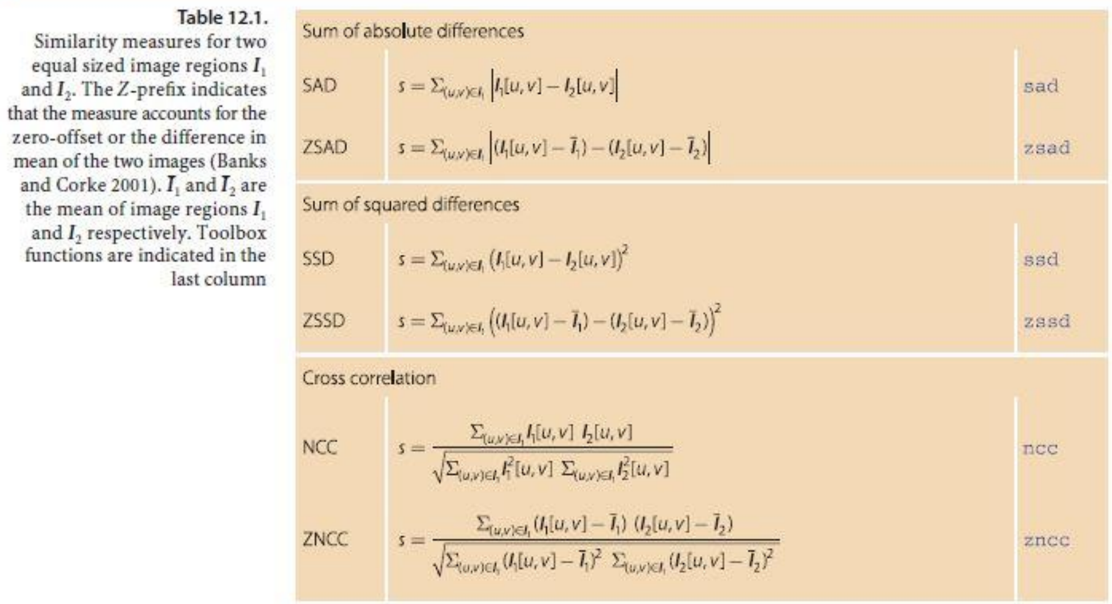


Table 12.1. Similarity measures for two equal sized image regions I_1 and I_2. The Z-prefix indicates that the measure accounts for the zero-offset or the difference in mean of the two images (Banks and Corke 2001). $\overline{I}_1$ and $\overline{I}_2$ are the mean of image regions I_1 and I_2 respectively. Toolbox functions are indicated in the last column

Sum of absolute differences		
SAD	$s = \Sigma_{(u,v)\in I_1} \left| I_1[u,v] - I_2[u,v] \right|$	sad
ZSAD	$s = \Sigma_{(u,v)\in I_1} \left| (I_1[u,v] - \overline{I}_1) - (I_2[u,v] - \overline{I}_2) \right|$	zsad
Sum of squared differences		
SSD	$s = \Sigma_{(u,v)\in I_1} (I_1[u,v] - I_2[u,v])^2$	ssd
ZSSD	$s = \Sigma_{(u,v)\in I_1} \left((I_1[u,v] - \overline{I}_1) - (I_2[u,v] - \overline{I}_2) \right)^2$	zssd
Cross correlation		
NCC	$s = \dfrac{\Sigma_{(u,v)\in I_1} I_1[u,v]\ I_2[u,v]}{\sqrt{\Sigma_{(u,v)\in I_1} I_1^2[u,v]\ \Sigma_{(u,v)\in I_1} I_2^2[u,v]}}$	ncc
ZNCC	$s = \dfrac{\Sigma_{(u,v)\in I_1} (I_1[u,v] - \overline{I}_1)\ (I_2[u,v] - \overline{I}_2)}{\sqrt{\Sigma_{(u,v)\in I_1} (I_1[u,v] - \overline{I}_1)^2\ \Sigma_{(u,v)\in I_1} (I_2[u,v] - \overline{I}_2)^2}}$	zncc

Figure 1: Extracted from book 'Robotics, Vision and Control by Peter Corke'

Note: Some images have been omitted due to privacy reasons.

Working of Code:

The code is implemented in MATLAB R2018b. A high-level explanation of the code is provided below:

- The code inquires from the user as to which function and which similarity measure to use for the operation of template matching as there are two possibilities:
  - <u>Image Processing Toolbox function:</u> this uses '*normxcorr2()*', a function provided by Image Processing Toolbox for calculating ZNCC. To use this, the user must input *'I'*.
  - <u>Custom Function:</u> this uses '*mycustomfnt()*', a function written by me for the calculation of Similarity Matrix. To use this, the user must input *'C'*. The code will then further inquire whether to use NCC (pass *'N'*) or ZNCC (pass *'Z'*).
- All images are read into the workspace, are converted to grayscale image, and are mapped to range [0, 1].

- The code calculates similarity measure and draws bounding boxes on the image for three different cases:
    1) <u>Case 1 (Templates from and Search on same image):</u> This approach reads the human image, creates templates of ears, eyes, and nose (using hard coded values), calculates similarity matrix, finds coordinates with maximum similarity, adjusts for the zero-padding introduced during the correlation operation, draws boxes of different colors on the same human image and saves the images in the directory.
    2) <u>Case 2 (Search given templates on given image):</u> It is quite similar to the last case except in this case given templates of SMME, Eiffel Tower and Robodog are read in the workspace and are searched in the provided image.
    3) <u>Case 3 (Templates from one image and Search them on another image):</u> In this case, the templates created in the case 1 are searched in another image of human (not the one from which features were obtained).
- An account of transformations applied is given below:
    - In case of custom function, the images are scaled down to improve the processing speed. (by factor of 5 in 1st and 3rd case, and by factor of 6 in 2nd case).
    - Reflection of eye and ear was done to obtain second eye and second ear.
    - In case 2 and case 3, templates were scaled up so that the matching could be performed properly. In case 2, image of SMME and RD was scaled up by 2.1 in both directions while in case 3, it was found that only nose feature required the scaling, and it was increased by factor of 1.5 in horizontal direction and 1.12 in vertical direction.
- When performing full correlation operation on an image, it must be padded with some values to avoid getting undefined behavior at the border. Thus, the images are zero padded. Further, after finding the coordinates which had maximum value in similarity matrix, these coordinates are adjusted to remove padding introduced before.
- For each case, the code creates and saves three different images in the local directory:
    1) Image with bounding boxes
    2) Figure of templates
    3) Figure of similarity matrices of each template slid over the image
- A brief description of ‘mycustomfnt()’ is given: It takes template, image and type of similarity measure as input and returns the similarity measure matrix. If similarity measure chosen is ZNCC then it calculates mean of template and subtracts it from each pixel value of template. Same is applied to the region of image under consideration. Afterwards, for resulting matrix from full correlation operation, it creates matrix of zeros of size [c,d]:

  [x,y] =input image size

  [a,b]=template size

  [c,d]=resulted matrix size

  [c,d]=[x+a-1, y+b-1]

  Then it applies zero padding to input image by creating a matrix of zeros of size [c+a-1, c+b-1] and placing input image inside it. Then, using two nested loops, it extracts a part of image (size equal to template), applies the corresponding formula (as shown above in Table 1) and stores the value on the corresponding location in resulting matrix, created previously.

Results:

In this part the results obtained by applying all functions to all cases are given below. Moreover, the results can also be found in the folder provided with this report.

It may be noted that all functions were unable to find the left ear correctly in 3rd Case.

Images Used:

Case 1:

(image omitted)

Case 3:

Templates obtained from image of Case 1.

Search Image:

(image omitted)

Using Image Processing Toolbox:

Case 1:

Templates:

(image omitted)

Similarity Measure Output:

(image omitted)

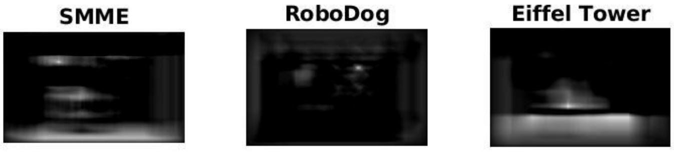
Output Image:

(image omitted)

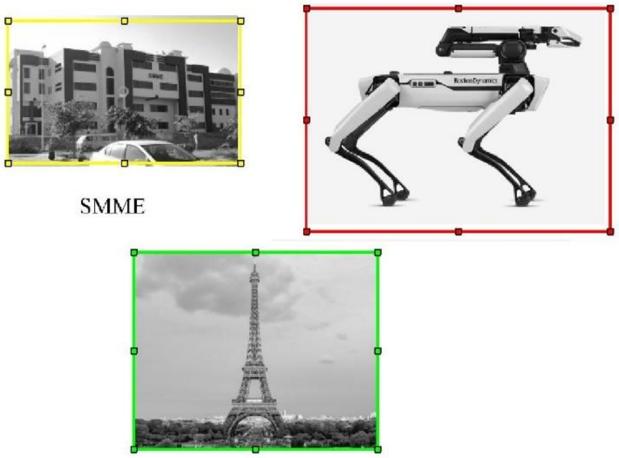
Case 2:

Templates:

Similarity Measure Output:



Output Image:



Case 3:

Templates:

(image omitted)

Similarity Measure Output:

(image omitted)

Output Image:

(image omitted)

Using Custom Function and NCC:

Case 1:

Templates:

(image omitted)

Similarity Measure Output:

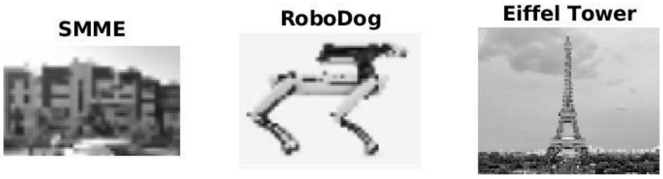
(image omitted)

Output Image:

(image omitted)

Case 2:

Templates:

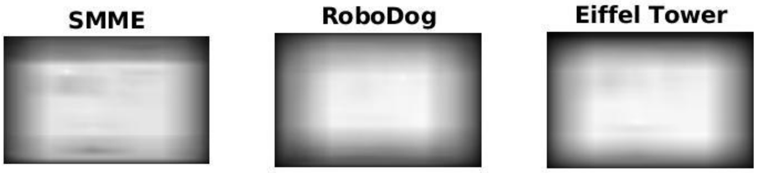


SMME

RoboDog

Eiffel Tower

Similarity Measure Output:

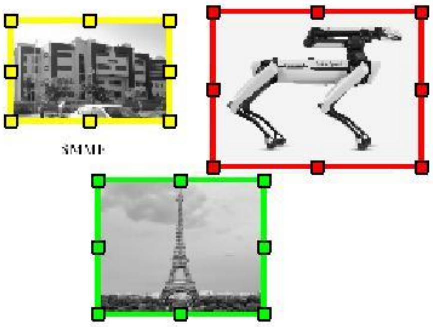


SMME

RoboDog

Eiffel Tower

Output Image:



Case 3:

Templates:

(image omitted)

Similarity Measure Output:

(image omitted)

Output Image:

(image omitted)

Using Custom Function and ZNCC:

Case 1:

Templates:

(image omitted)

Similarity Measure Output:

(image omitted)

Output Image:

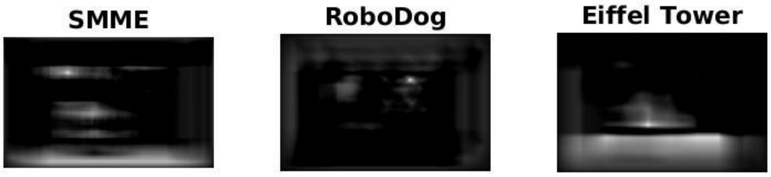
(image omitted)

Case 2:

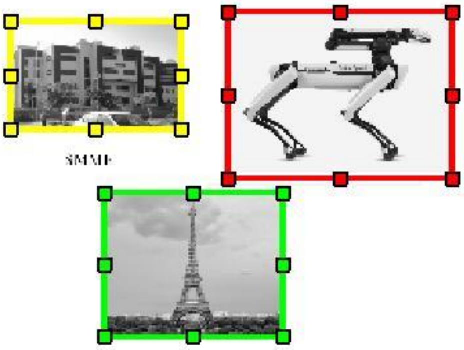
Templates:

(image omitted)

Similarity Measure Output:



Output Image:



Case 3:

Templates:

(image omitted)

Similarity Measure Output:

(image omitted)

Output Image:

(image omitted)

Code:

The following section provides the complete code. Furthermore, .m file is also provided with this report.

```
clear

%Figures
f1 = figure;
f2 = figure;
f3 = figure;

%% -------- Input -------

disp('Template Matching')

disp('Which function would you like to use? ')
lib = input('I for Image Processing Toolbox, C for custom function (in single quotes) ');

%Image Processing Toolbox function
if lib == 'I'
  disp('Zero mean Normalized Cross Correlation will be used as Similarity Measure')
  SM = 'Z';

%Custom function
elseif lib == 'C'
  disp('Which Similarity Measure do you want to use? ')
  SM = input('N for Normalized Cross Correlation and Z for Zero Mean Normalized Cross Correlation (in single quotes): ');

  %error check
  if SM ~= 'N' && SM ~= 'Z'
    error('Invalid Value. Enter N or Z only (in capital, in single quotes)')

  end

else

  %error chekc
  error('Invalid Value. Enter I or C only (in capital, in single quotes)')

end

%% ------- Main loop ------

for k = 1 : 3

  %Case 1: Template from and Search on the same image of human
  if k == 1

    % read images, convert it to grayscale and map it to [0, 1]
    Image = im2double( rgb2gray( imread('Picture.jpg')));
    %Templates
    EyeLeft = Image( 540:665, 650:850);
    EarLeft = Image( 600:833, 540:647);
    Nose = Image( 560:800, 780:912);

    %for custom function, scale down the image by a factor of 5 and
    %find corresponding templates
    if lib == 'C'

        T = affine2d([1/5 0 0;0 1/5 0; 0 0 1]);

        Image = imwarp(Image, T);

        %Templates
        EyeLeft = Image(108:133,130:170);
        EarLeft = Image(120:166, 108:129);
        Nose = Image(112:160, 156:182);

    end

    % Transformation for reflection of features
    T= affine2d([-1 0 0; 0 1 0; 0 0 1]);

    %Templates
    EyeRight = imwarp(EyeLeft, T);
    EarRight = imwarp(EarLeft,T);

    Features = {EarLeft, EyeLeft, Nose, EyeRight, EarRight};
    Names = ["Left Ear", "Left Eye", "Nose", "Right Eye", "Right Ear"];
end

%Case 2: Search for SMME, RD, and ET
if k == 2

    % read images, convert it to grayscale and map it to [0, 1]
    Image = im2double( rgb2gray( imread('Sample.jpg')));
    SMME = im2double( rgb2gray( imread('SMME.jpg')));
    RD = im2double( rgb2gray( imread('RD.jpg')));
    ET = im2double( rgb2gray( imread('ET.jpg')));

    %for custom function, scale down the images by a factor of 6
    if lib == 'C'

        T = affine2d([1/6 0 0;0 1/6 0; 0 0 1]);

        Image = imwarp(Image, T);
        SMME = imwarp(SMME, T);
        RD = imwarp(RD, T);
        ET = imwarp(ET, T);

    end

    %Transformation for scaling templates
    T = affine2d([2.1 0 0;0 2.1 0; 0 0 1]);

    %Templates
    SMME = imwarp(SMME, T, 'FillValues', SMME(1,1));
    RD = imwarp(RD, T, 'FillValues', RD(1,1));

    Features = {SMME, RD, ET};
    Names = ["SMME", "RoboDog", "Eiffel Tower"];
end

%Case 3: Templates from one image and Search on another image of human
if k == 3

    % read images, convert it to grayscale and map it to [0,1]
    Image = im2double(rgb2gray(imread('SearchImg.jpg')));

    %for custom function, scale down the images by a factor of 5
    if lib == 'C'

        T = affine2d([1/5 0 0;0 1/5 0; 0 0 1]);
        Image = imwarp(Image, T);

    end

    %Transformations for scaling templates
    T = affine2d([1.5 0 0; 0 1.12 0; 0 0 1]);

    %Template
    Nose= imwarp(Nose, T);
    Features = {EarLeft, EyeLeft, Nose, EyeRight, EarRight};
    Names = ["Left Ear", "Left Eye", "Nose", "Right Eye", "Right Ear"];

end

% different colors for bounding boxes
Color = ['y','r','g','b','k'];

noOfTemp = size(Names);

%Show Image for Searching
figure(f1);
imshow(Image);

for i = 1:noOfTemp(2)

    %Plot Template
    figure(f2);
    subplot(1,noOfTemp(2), i)
    imshow(Features{i})
    title(Names(i))

    %Find Similarity Matrix
    if lib == 'I'
        score = normxcorr2(Features{i}, Image);
    else
        score = mycustomfnt(Features{i}, Image, SM);
    end

    %Plot Similarity Measure
    figure(f3);
    subplot(1,noOfTemp(2),i)
    imshow(score)
    title(Names(i))

    %Find cordinates with max value
    [y,x] = find( score == max (score(:)));

    %Removing Zero padding
    y = y - size( Features{i}, 1);
    x = x - size( Features{i}, 2);

    % draw bounding boxes
    figure(f1);
    drawrectangle(gca,'Position', [x,y,size(Features{i},2), size(Features{i},1)], 'Color', Color(i), 'FaceAlpha',
0);

  end

  % save figures
  saveas(f1,['fig1_Case ', num2str(k),'_','lib_', lib, '_', 'SM_', SM,'.jpg']);
  saveas(f2,['fig2_Case ', num2str(k),'_','lib_', lib, '_', 'SM_', SM,'.jpg']);
  saveas(f3,['fig3_Case ', num2str(k),'_','lib_', lib, '_', 'SM_', SM,'.jpg']);
end


%% ------- My Custom Function ------

function new_img = mycustomfnt (temp, img, sm)

%zero centering for ZNCC
if sm == 'Z'
  T = temp - mean( temp(:));
else
  T = temp;
end

%store size of template and image
temp_size = size(temp);
img_size = size(img);

%calculate size of new image
new_img_size = [temp_size(1)+img_size(1)-1, temp_size(2) + img_size(2)-1];

%create new image of zeros
new_img = zeros(new_img_size);

%apply zero padding
Img = zeros(new_img_size+temp_size-1);
Img(temp_size(1):new_img_size(1), temp_size(2):new_img_size(2))=img;

%NCC/ZNCC
for i = 1:new_img_size(1)

  for j = 1:new_img_size(2)
```

```
        %extract part of image
        p = Img(i:i+temp_size(1)-1, j:j+temp_size(2)-1);

        %zero center for ZNCC
        if sm == 'Z'
            P = p - mean(p(:));
        else
            P = p;
        end

        %find cross correlation coefficients
        C = P.*T;
        new_img(i,j) = sum(C(:)) / ((sum(P(:).^2)) * (sum(T(:).^2))) .^0.5;
    end
end
end
```