Project 3 Report

Template Matching

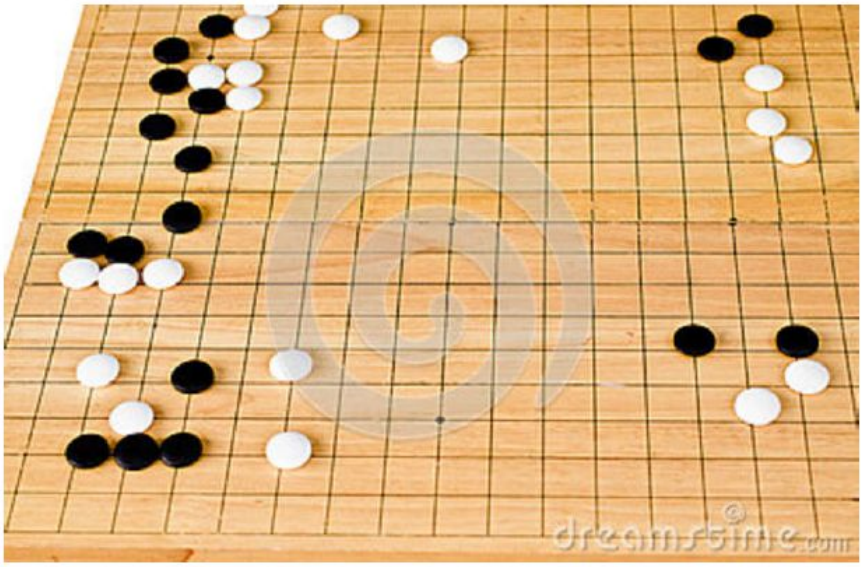
CPE428-01,02 - Computer Vision

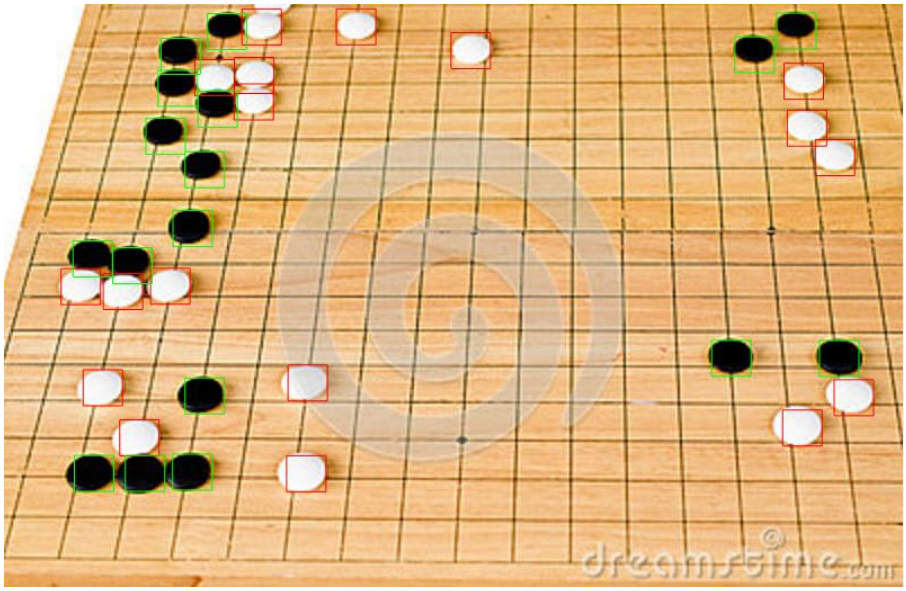
Prepared for:	Professor Xiaozheng Zhang
Prepared By:	The “A” Team; Nikhil Patolia, Alec Hardy
Date Submitted:	Saturday, February 1, 2020

Table of Contents

Part A

Black and white stones on a Go board are identified and demarcated using cross-correlation template matching. The original image and demarcated image are shown below. In the demarcated image, black stones are bound by a green box and white stones are bound by a red box.

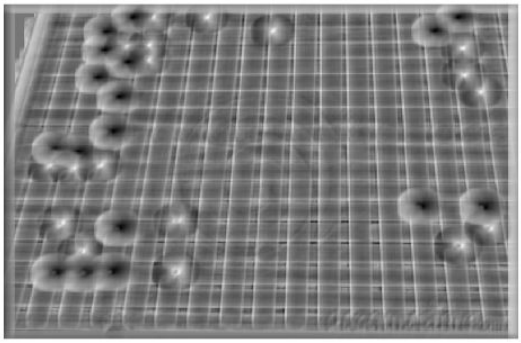




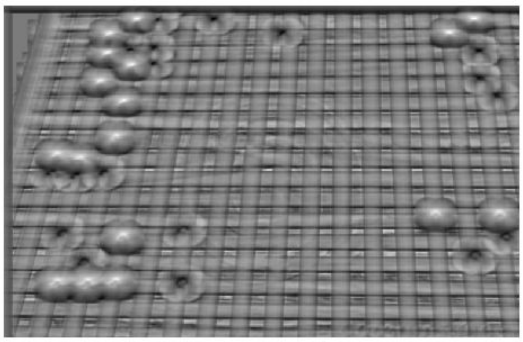
In order to detect the different stones, the following two templates were used in cross-correlation:

The original image and two template images were converted to grayscale before processing. Different templates were experimented with (some containing more pixels of the surrounding board, some more cropped to only contain the flat part of the stone) to determine the best template that could be used to

identify all of the stones. Identifying stones where groups of white and black stones are adjacent took template tuning to perfect. The correlation maps generated from the above selected templates are shown in the figure below.



Correlation map for white stones



Correlation map for black stones

Different correlation maps were used for each template. After generating the correlation map for each template, threshold values used to create a binary map of detected matches. These threshold values needed to be tuned manually. White stones use a threshold of 0.65 while black stones use a threshold of 0.75. An example of the binary map generated after thresholding, for white stones, is shown in the figure below. Each white dot corresponds to a white stone.

The detection map worked great for identifying known objects that were very similar in size, shape, and color. For instance, from looking at the correlation maps, it is very apparent that the detector is able to distinguish from white and black stones. However, if the stones were of two colors which had very similar grayscale values, then differentiation would be difficult using the above method. Accordingly, if the stones were of different sizes, then the detector would only be able to detect stones that were of the same size as the template (unless the template is scaled).

All of the code used in Part A is shown below.

Code for Part A

```
%% Part A
clc;
close all;

% Constants that will affect performance (these need to be manually tuned)
THRESHOLD_WHITE = 0.65;
THRESHOLD_BLACK = 0.75;

I = rgb2gray(imread("go1.jpg"));
T_white = rgb2gray(imread("WhiteStone.JPG"));
T_black = rgb2gray(imread("BLackStone.JPG"));

% Perform correlation
C_white = normxcorr2(T_white, I);
C_black = normxcorr2(T_black, I);

% Correct the image size
size_diff_x = size(C_white,1)-size(I,1);
size_diff_y = size(C_white,2)-size(I,2);
C_white =
C_white(floor(1+size_diff_x/2):floor(size(C_white,1)-size_diff_x/2),floor(1+size_diff
_y/2):floor(size(C_white,2)-size_diff_y/2));

size_diff_x = size(C_black,1)-size(I,1);
size_diff_y = size(C_black,2)-size(I,2);
C_black =
C_black(floor(1+size_diff_x/2):floor(size(C_white,1)-size_diff_x/2),floor(1+size_diff
_y/2):floor(size(C_white,2)-size_diff_y/2));

% Display the correlation map
figure();
set(gcf,'color','w');
subplot(1,2,1);
imshow(C_white,[]);
xlabel("Correlation map for white stones");
subplot(1,2,2);
imshow(C_black,[]);
xlabel("Correlation map for black stones");

% Apply threshold to find template matches
C_white(C_white < THRESHOLD_WHITE) = 0;
C_white(C_white > 0) = 1;
C_black(C_black < THRESHOLD_BLACK) = 0;
C_black(C_black > 0) = 1;
```

```
% Find centroids of template matches so we can draw boxes
cc_w = bwconncomp(C_white);
s_w = regionprops(cc_w, 'Centroid', 'FilledArea');
centroids_w = cat(1,s_w.Centroid);
cc_b = bwconncomp(C_black);
s_b = regionprops(cc_b, 'Centroid', 'FilledArea');
centroids_b = cat(1,s_b.Centroid);

% Show the figure, draw the boxes around the centroids
figure();
imshow(imread("go1.jpg"));
for i = 1:size(centroids_w,1)
    C = centroids_w(i,:);
    rectangle('Position',[C(1)-20,C(2)-15,35,32],'EdgeColor',[1 0 0])
end
for i = 1:size(centroids_b,1)
    C = centroids_b(i,:);
    rectangle('Position',[C(1)-17,C(2)-10,35,32],'EdgeColor',[0 1 0])
end
```

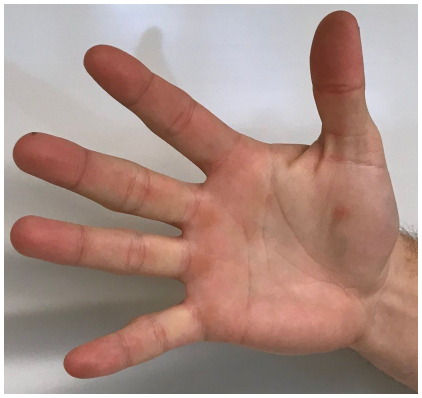
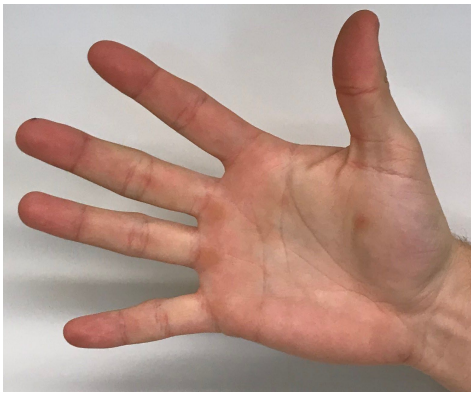
Part B

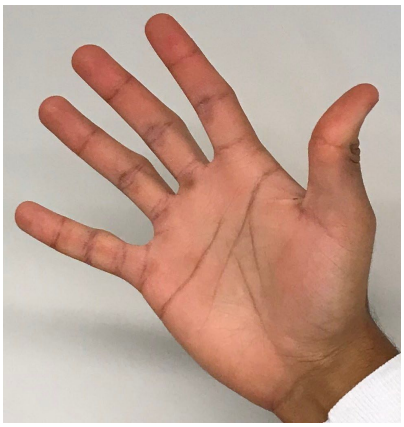
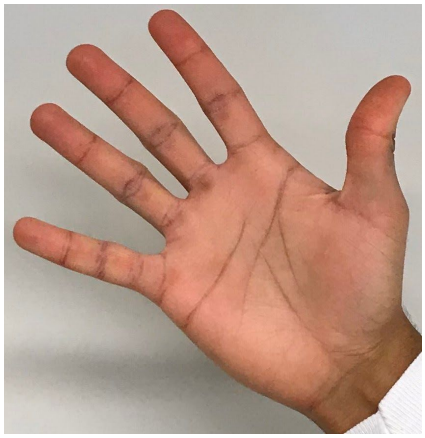
Objective

For this part of the project, we were supposed to take what we learned from Part A and make a program that detects whether a picture has a rock, paper, or scissors move played.

Data Collection

In order to determine whether an image is a rock, paper, or scissors move, we had to take several pictures as templates for each. We took pictures of each move against the white background of the whiteboard in class and cropped each image to fit our hands exactly. The images are shown below:

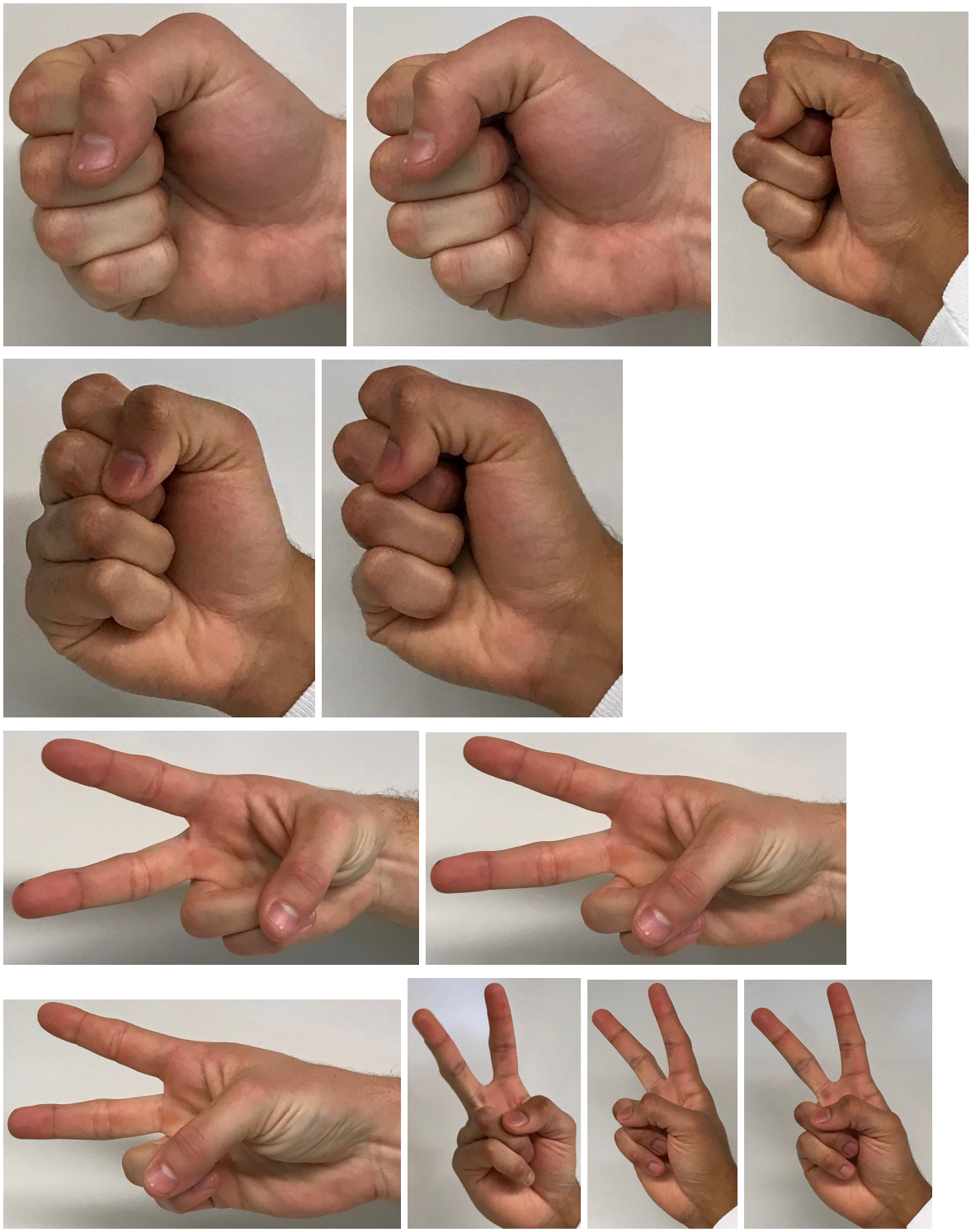



Image Processing

We processed the images by converting both the templates and the input image to grayscale images and rotating the input image by 90 degrees. The image must be rotated 90 degrees since MATLAB rotates the original image by 90 degrees when using the image so this needs to be counteracted. We then use normxcorr2 on the images to get the correlation between the two images. Next we remove the padding put on by the normxcorr2 and add this value up for the correlation between the input image and each

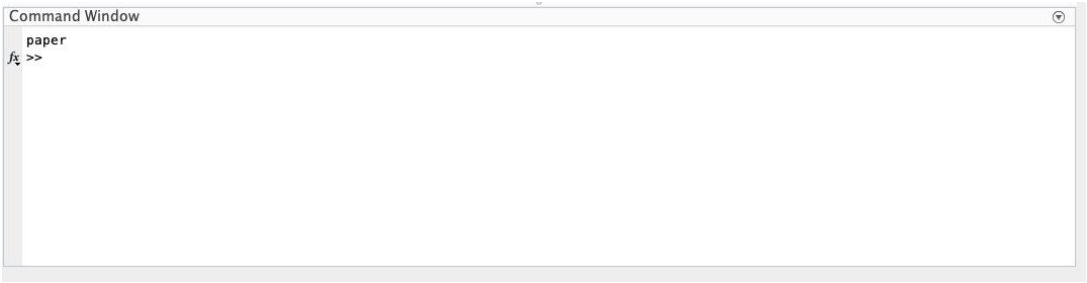
template. The group of templates that have the highest correlation value is what the program guesses the move is.

Input/Output

For the input we simply input the name of the image and get a print message as an output. An example is shown below:

Input:

Output:



Part B Code

```
%% Part B
clc;
close all;
rock = 0;
paper = 0;
scissors = 0;
```

```
I = "test.JPG";

for i = 1:6
    %Gets the correlations of the input compared to each template image
    %It then keeps a running total of the correlation values
    rock = rock + find_match(I, sprintf('r%i.jpg', i));
    paper = paper + find_match(I, sprintf('p%i.jpg', i));
    scissors = scissors + find_match(I, sprintf('s%i.jpg', i));

end

%finds the maximum correlation from the three options
m = max([rock, paper, scissors]);

%prints the highest correlated option
if m == rock
    disp("rock")
elseif m == paper
    disp("paper")
else
    disp("scissors")
end

function O = find_match(I, t)

%Convert the image to grayscale and rotate it 90 degrees
Image = rgb2gray(imrotate(imread(I),-90));

%Convert the templates to grayscale
template = rgb2gray(imread(t));

c = normxcorr2(template, Image);

%Get the ypeak and xpeak
[ypeak, xpeak] = find(c==max(c(:)));

%return the unpadded ypeak and xpeak
yoffSet = ypeak - size(template, 1);
xoffSet = xpeak - size(template, 2);

O = yoffSet * xoffSet;
end
```

Reflection

Using cross-correlation template matching is a basic yet powerful technique used to identify regions in an image that are similar to a provided template. The cross-correlation technique works when the template is

of the same size as the regions to be identified. As discovered from the stone matching exercise, there are some parameters which must be tuned in order to achieve desired performance, such as thresholding value and template region. Changing either of these parameters yielded different end behaviors. Accordingly, as discovered from the rock/paper/scissors exercise, the cross-correlation technique is susceptible to geometric differences in the target image - changes of scale, rotation, etc. cause low correlation values. Given a bounded set of possibilities, cross-correlation can be successfully used as a classifier because it provides a value (between -1 and 1 if normalized) between the template and each region of an image. However, significant differences between the target image and provided templates can cause poor performance. In order to account for this, multiple templates or the geometric modification of the provided template must be utilized.

- Alec Hardy

By working on this project, I learned that image recognition is a lot more resource intensive than I previously thought. For the rock, paper, scissors project, we only used 6 templates to compare to in order to determine what symbol the input image is making. There is only a choice of three options(rock, paper, or scissors) but there was a lot of work put in but the final result is only specific to this game. Our program may not even fair well against input images that are taken from different angles or that have different sized hands but even with 6 template images for each move to be played, the program took a considerable amount of time. I feel like I have a new appreciation for image recognition that happens quickly and on several different types of subjects after doing this project.

- Nikhil Patolia