Project 3

Lab Partners: Ryan Myers, Mihir Deshmukh

Sections: 428-01 (Deshmukh), 428-01 (Myers)

Winter Quarter

2/1/2020

Professor: Jane Zhang

Part A

Part A required identifying all the pieces on a go board. The image of the board (Figure 1) below is shown below.

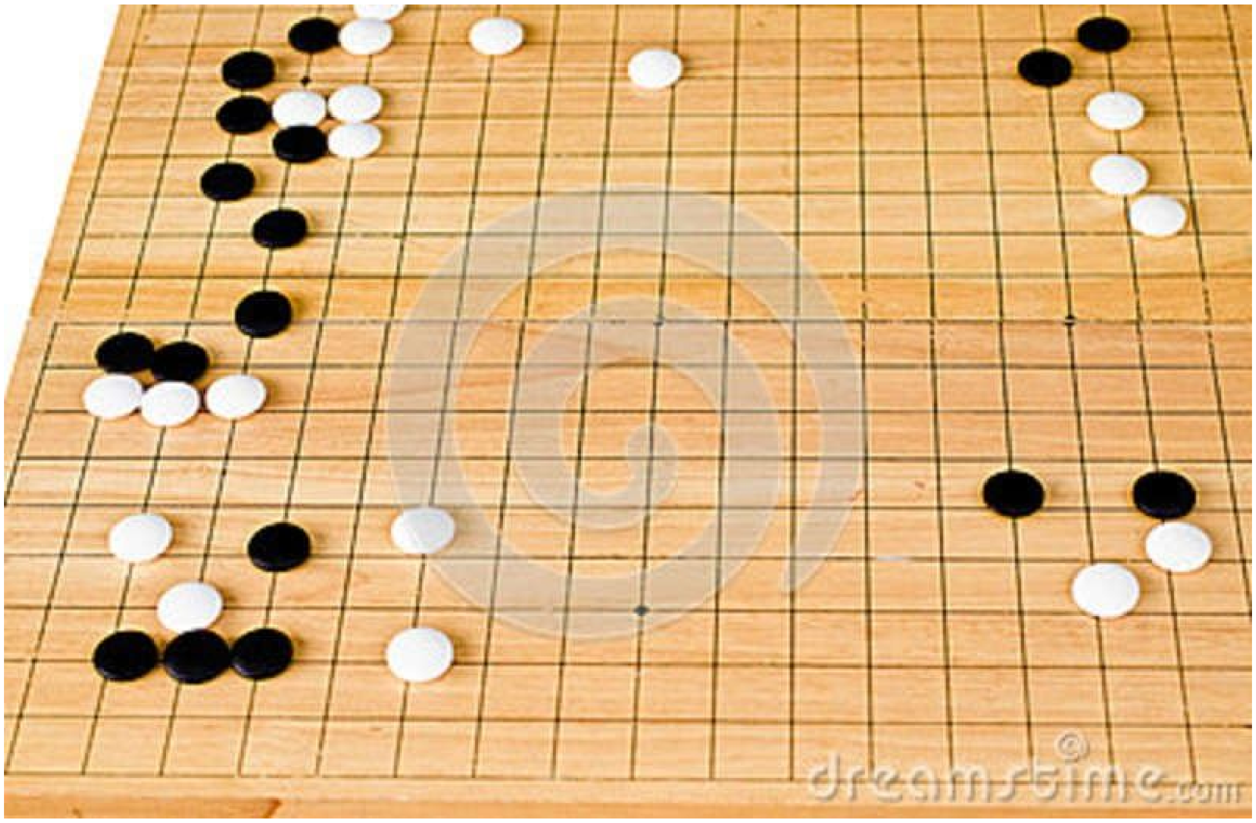

Figure 1: Go Board

To identify the pieces we used a technique called template matching. Template matching is implemented using normalized cross correlation. To perform normalized cross correlation we need images of the pieces that we're trying to identify on the board. Figure 2 and 3 show the images of the black and white go pieces we were trying to template match on the board.

Figure 2: Black Go Piece

Figure 3: White Go Piece

For the images of the pieces (Figure 2 & 3) we want to minimize the amount of board that is in them so that it more strongly correlates with the actual pieces than the board to better identify the pieces. We then perform normalized correlation with the pieces on the board. We then find all the regions where the pieces strongly correlate with the board by looking at the maximum correlation value and seeing all regions where there is a correlation within a threshold of the maximum through trial and error. We also applied a median filter to the board to smooth it out as some of the white pieces are pixelated and were being wrongly identified. The code where we do this can be found in the Appendix under "Part A main Code". Figure 4 shows the image of the board with bounding boxes over the pieces that our code identified.

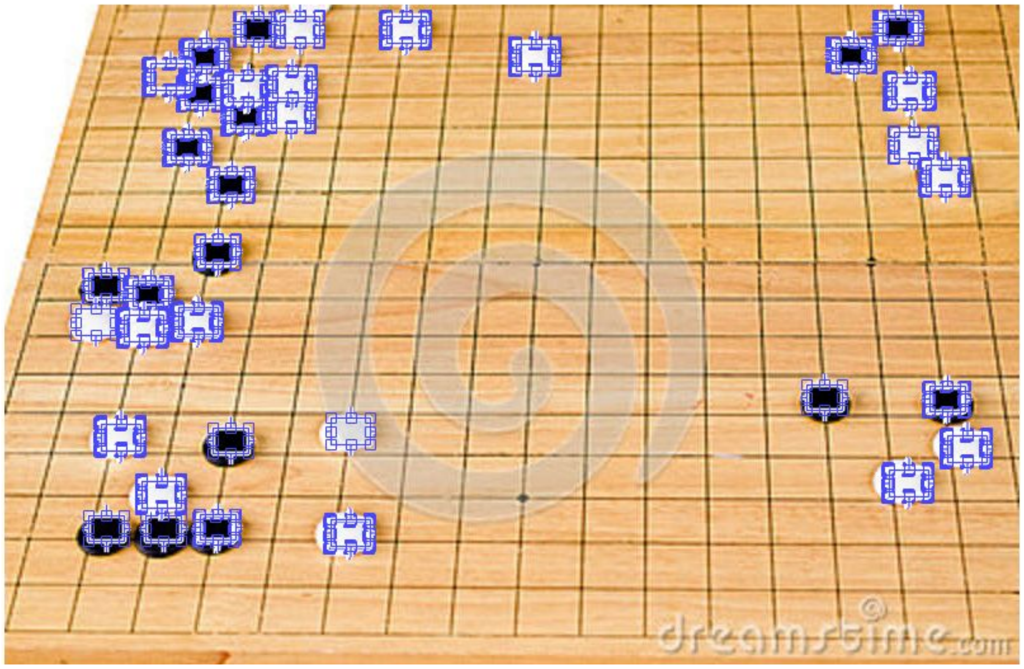

Figure 4: Go Board with Bounding Boxes over Identified Images

Part A 1)

As we can see our algorithm did pretty well in identifying most pieces. Our code miscategorized one region as having a piece when it did not. We believe this might be due to the board being converted to a grayscale image which converts the wood to a light grey and the white piece also appears light grey and because that region of the board has pieces close to it. Also we failed to identify one piece at the very top and we believe that might be due to that piece being cropped and is only half a piece.

Part A 2)

Template matching works best when the images provided are of a high quality and the template has as less of the background as possible. This ensures that the template doesn't try and correlate the background and rather what we want it to(the go pieces in this case and not the board itself).

Part B

To implement the proof-of-concept we took images of one of our hands in a rock, paper, and scissors position and then made each of the images similar sizes with as much background possible cropped out. We then took test images of both of our hands in 8 different positions and different distances away from the camera so the template was not always the same size as the hands in the image. The algorithm we came up with to provide accurate template matching for test images with different skin tones and distances away from the camera was then applied to each image and the result compared to the expected answer. To determine what hand shape was chosen, we correlated each template with the image three times, each time being resized by half its original resolution. Doing this allowed us to compensate for very close or very far away images. This would produce 9 correlation matrices due to three hand shapes each being resized 3 times. We then took the maximum value of each of these matrices and compared them to see which shape, rock, paper, or scissors had the highest correlation. Using this method we were able to accurately determine 5 out of the 8 test shapes given.

The three templates used are as shown below:

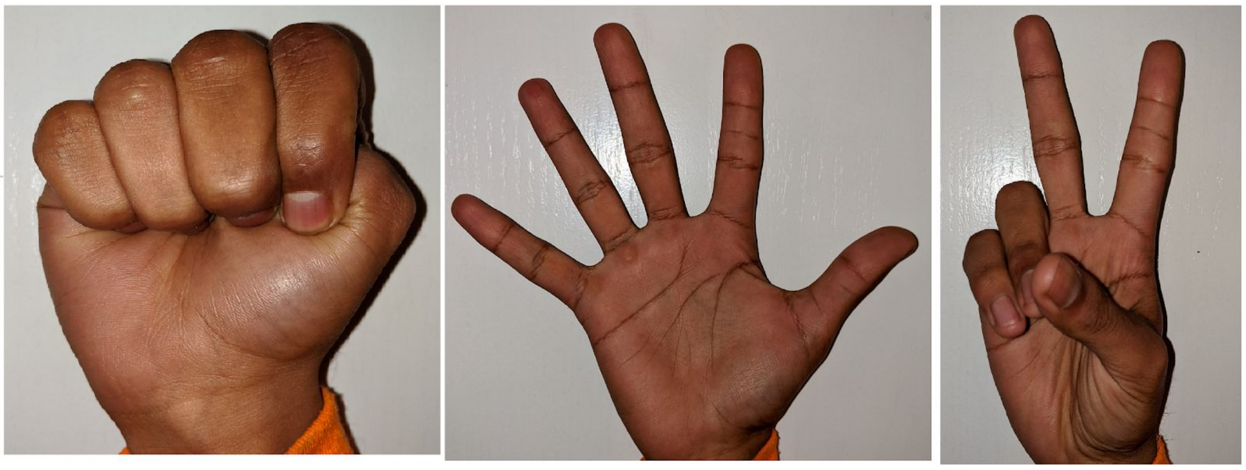

Figure 5: Templates used for matching the rock paper scissors test images

The following are the results:

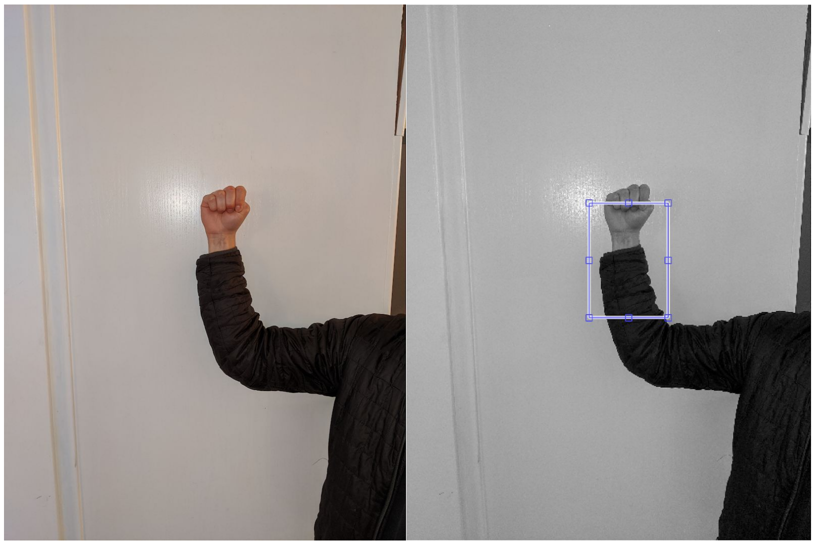
Figure 6: Original image on the left, template matched image on the right
Expected shape: Rock, Predicted shape: Scissors

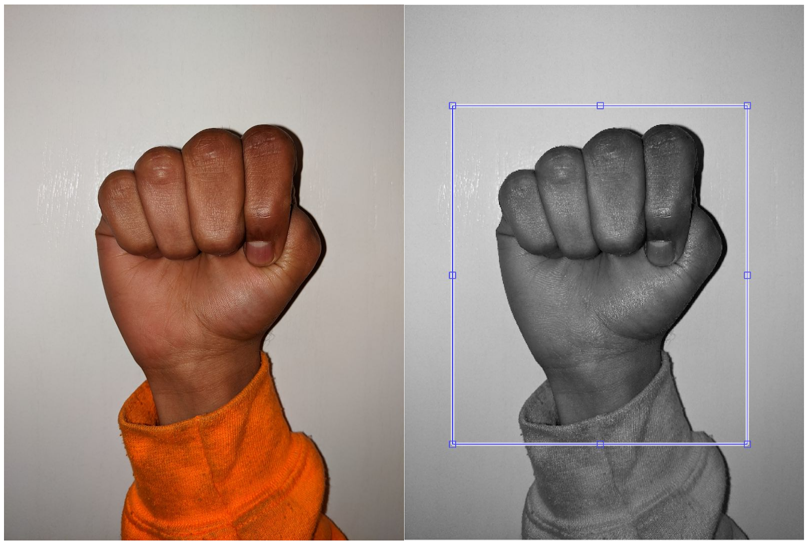
Figure 7: Original image on the left, template matched image on the right
Expected shape: Rock, Predicted shape: Rock

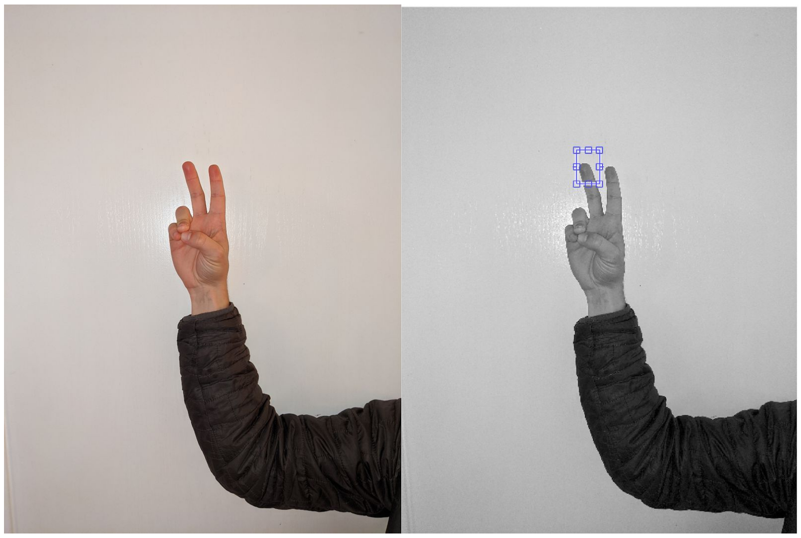

Figure 8: Original image on the left, template matched image on the right
Expected shape: Scissors, Predicted shape: Scissors

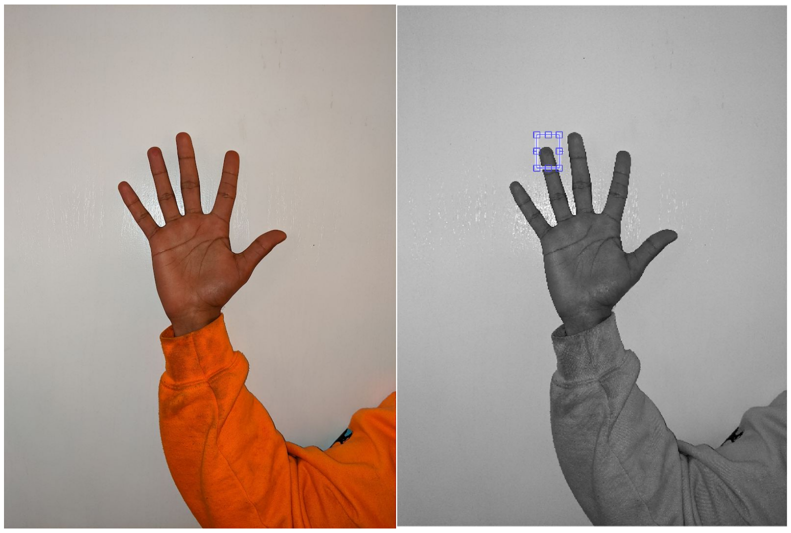

Figure 9: Original image on the left, template matched image on the right
Expected shape: Paper, Predicted shape: Scissors

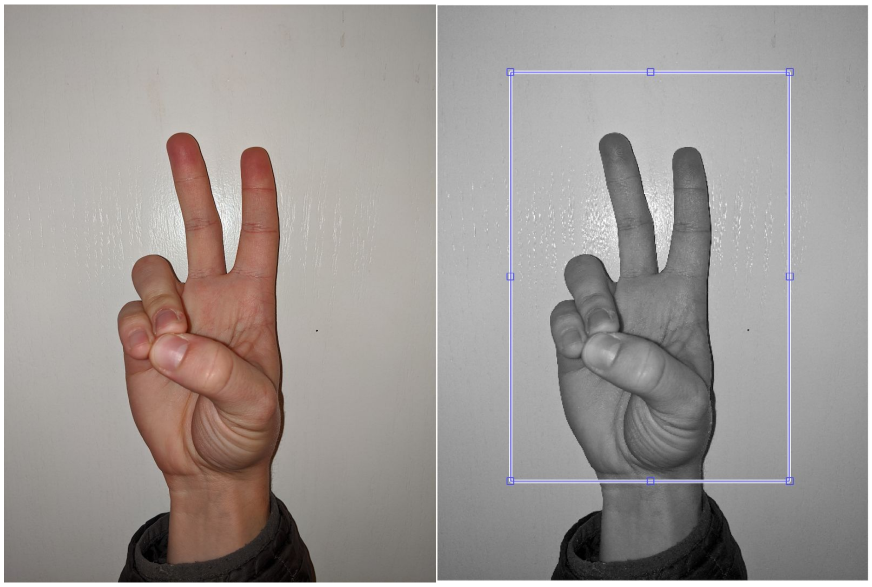

Figure 10: Original image on the left, template matched image on the right
Expected shape: Scissors, Predicted shape: Scissors

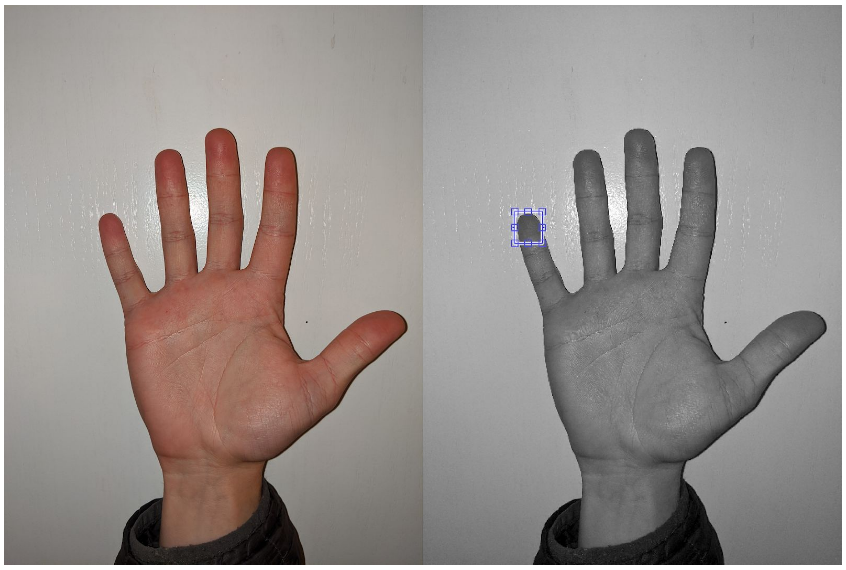

Figure 11: Original image on the left, template matched image on the right
Expected shape: Paper, Predicted shape: Rock

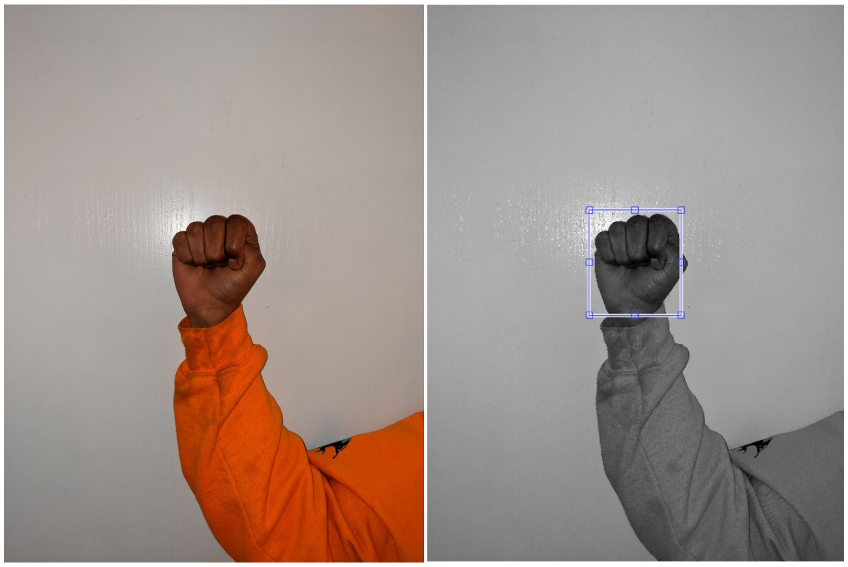

Figure 12: Original image on the left, template matched image on the right
Expected shape: Rock, Predicted shape: Rock

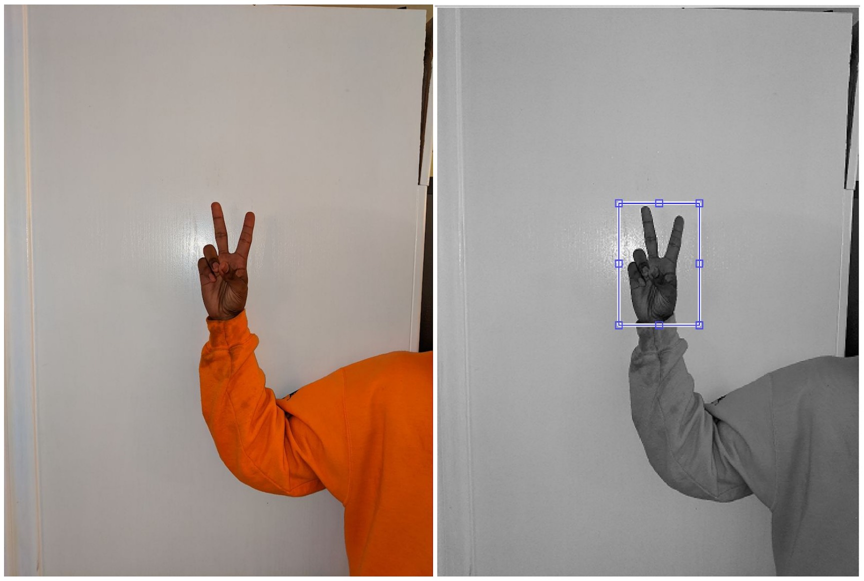

Figure 13: Original image on the left, template matched image on the right
Expected shape: Scissors, Predicted shape: Scissors

As can be seen from the images, almost all the rock and scissors shapes could be matched with reasonable accuracy using our algorithm. However, the issues came with the paper shape where we were not able to correctly identify any images. This is likely due to the fact that the scissors template was similar to the paper shape and we could have made the tests more accurate if we altered the way we made the paper shape. Using this method we were able to correctly identify 5 out of the 8 test images, giving us room for improvement but overall good results.

Code Appendix

Part A Main Code

```
clc;
clear m;
clear n;
close all;
%RGB image of board
rgb_board = imread('go1.jpg');
%BW image of black go piece
black_piece = rgb2gray(imread('go_piece_black.jpg'));
%BW image of white go piece
white_piece = rgb2gray(imread('go_piece_white_1.jpg'));
%BW image of board
board = rgb2gray(imread('go1.jpg'));
%Apply a median filter to smooth out the pieces
board = median_filter(board,5);

%Perform template matching to the board using the white and black go
pieces
black_corr = normxcorr2(black_piece,board);
white_corr = normxcorr2(white_piece,board);
%figure, surf(c), shading flat

%Find locations of pieces that are within the realm of the highest
%correllation
[ypeak_b, xpeak_b] = find(black_corr>=0.737*max(black_corr(:)));
[ypeak_w, xpeak_w] = find(white_corr>=0.82*max(white_corr(:)));

%Find the number of correlated coordinates for each piece
[m_b,n_b] = size([ypeak_b, xpeak_b]);
[m_w,n_w] = size([ypeak_w, xpeak_w]);

%Adjust every coordinate by the size of the original image
for row = 1:m_b
    ypeak_b(row) = ypeak_b(row) - size(black_piece,1);
    xpeak_b(row) = xpeak_b(row) - size(black_piece,2);
end
hFig = figure;
hAx = axes;
```

```
for row = 1:m_w
    ypeak_w(row) = ypeak_w(row) - size(white_piece,1);
    xpeak_w(row) = xpeak_w(row) - size(white_piece,2);
end

%Overlay a rectangle on the board for every region of correlation
imshow(rgb_board,'Parent', hAx);
for row = 1:m_b
    if(row > 1)
        if(xpeak_b(row) > 9+xpeak_b(row-1) || ypeak_b(row) > 5+
ypeak_b(row-1) || xpeak_b(row) < xpeak_b(row-1)-9 || ypeak_b(row) <=
ypeak_b(row-1)-1)
            imrect(hAx, [xpeak_b(row), ypeak_b(row),
size(black_piece,2), size(black_piece,1)]);
        end
    end
end

for row = 1:m_w
    imrect(hAx, [xpeak_w(row), ypeak_w(row), size(white_piece,2),
size(white_piece,1)]);
end
```

Supplementary Median Filter Function for Part A

```
function [filtered] = median_filter(im, mask_size)
%MEDIAN_FILTER Summary of this function goes here
%   Detailed explanation goes here
if mod(mask_size, 2) == 0 || mask_size < 3
    return
end

im_size = size(im);
filtered = uint8(zeros(im_size));
pad_size = floor(mask_size / 2);
im = padarray(im, [pad_size pad_size], 0, 'both');

for i = pad_size + 1:im_size(1)
    for j = pad_size + 1:im_size(2)
        neighbors = im(i-pad_size:i+pad_size, j-pad_size:j+pad_size);
        neighbors = neighbors(:)';
        filtered(i - pad_size, j - pad_size) =
uint8(floor(median(neighbors)));
    end
end
```

Part B Main Code

```
clc
close all

correct = 0;

im = imread('rock_far.jpg');
result = rps(im);
fprintf('\nExpected answer: rock');
fprintf('\nanswer: %s\n', result);
if strcmp(result, 'rock')
    correct = correct + 1;
end

im = imread('rock_close.jpg');
result = rps(im);
fprintf('\nExpected answer: rock');
fprintf('\nanswer: %s\n', result);
if strcmp(result, 'rock')
    correct = correct + 1;
end

im = imread('scissors_medium.jpg');
result = rps(im);
fprintf('\nExpected answer: scissors');
fprintf('\nanswer: %s\n', result);
if strcmp(result, 'scissors')
    correct = correct + 1;
end

im = imread('paper_medium.jpg');
result = rps(im);
fprintf('\nExpected answer: paper');
fprintf('\nanswer: %s\n', result);
if strcmp(result, 'paper')
    correct = correct + 1;
end

im = imread('scissors_close.jpg');
result = rps(im);
fprintf('\nExpected answer: scissors');
fprintf('\nanswer: %s\n', result);
if strcmp(result, 'scissors')
    correct = correct + 1;
end

im = imread('paper_close.jpg');
result = rps(im);
fprintf('\nExpected answer: paper');
fprintf('\nanswer: %s\n', result);
if strcmp(result, 'paper')
    correct = correct + 1;
end

im = imread('rock_medium.jpg');
result = rps(im);
fprintf('\nExpected answer: rock');
fprintf('\nanswer: %s\n', result);
if strcmp(result, 'rock')
    correct = correct + 1;
end

im = imread('scissors_medium.jpg');
result = rps(im);
fprintf('\nExpected answer: scissors');
fprintf('\nanswer: %s\n', result);
if strcmp(result, 'scissors')
    correct = correct + 1;
end
fprintf('Correct: %d/8\n', correct);
```

Part B Supplementary Function RPS

```
function [result] = rps(im)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
r_corrs = 1:3;
p_corrs = 1:3;
s_corrs = 1:3;
results = 1:3;

im = rgb2gray(im);
im = imrotate(im, 270);

%% find maximum rock correlation
r_temp = imread('rock_template.jpg');
r_temp = rgb2gray(r_temp);
%r_temp = median_filter(r_temp, 3);

r_corr = normxcorr2(r_temp, im);
%Keep track of correlation and template max for each run
r_corr_copy = r_corr;
r_temp_copy = r_temp;
r_corrs(1) = max(r_corr(:));

r_temp = imresize(r_temp, 0.3);
r_corr = normxcorr2(r_temp, im);
r_corrs(2) = max(r_corr(:));
%Keep track of correlation and template max for each run
if(r_corrs(2) > max(r_corr_copy(:)))
    r_corr_copy = r_corr;
    r_temp_copy = r_temp;
end

r_temp = imresize(r_temp, 0.3);
r_corr = normxcorr2(r_temp, im);
r_corrs(3) = max(r_corr(:));
%Keep track of correlation and template max for each run
if(r_corrs(3) > max(r_corr_copy(:)))
    r_corr_copy = r_corr;
    r_temp_copy = r_temp;
end
```

```
results(1) = max(r_corrs);

disp(r_corrs)

%%

%% find maximum paper correlation
p_temp = imread('paper_template.jpg');
p_temp = rgb2gray(p_temp);
%p_temp = median_filter(p_temp, 3);

p_corr = normxcorr2(p_temp, im);
%Keep track of correlation and template max for each run
p_corr_copy = p_corr;
p_temp_copy = p_temp;
p_corrs(1) = max(p_corr(:));


p_temp = imresize(p_temp, 0.3);
p_corr = normxcorr2(p_temp, im);
p_corrs(2) = max(p_corr(:));
%Keep track of correlation and template max for each run
if(p_corrs(2) > max(p_corr_copy(:)))
    p_corr_copy = p_corr;
    p_temp_copy = p_temp;
end

p_temp = imresize(p_temp, 0.3);
p_corr = normxcorr2(p_temp, im);
p_corrs(3) = max(p_corr(:));
%Keep track of correlation and template max for each run
if(p_corrs(3) > max(p_corr_copy(:)))
    p_corr_copy = p_corr;
    p_temp_copy = p_temp;
end

results(2) = max(p_corrs);

disp(p_corrs)


%%
```

```matlab
%% find maximum scissors correlation

s_temp = imread('scissors_template.jpg');
s_temp = rgb2gray(s_temp);
%s_temp = median_filter(s_temp, 3);

s_corr = normxcorr2(s_temp, im);
%Keep track of correlation and template max for each run
s_corr_copy = s_corr;
s_temp_copy = s_temp;
s_corrs(1) = max(s_corr(:));

s_temp = imresize(s_temp, 0.3);
s_corr = normxcorr2(s_temp, im);
s_corrs(2) = max(s_corr(:));
%Keep track of correlation and template max for each run
if(s_corrs(2) > max(s_corr_copy(:)))
    s_corr_copy = s_corr;
    s_temp_copy = s_temp;
end

s_temp = imresize(s_temp, 0.3);
s_corr = normxcorr2(s_temp, im);
s_corrs(3) = max(s_corr(:));
%Keep track of correlation and template max for each run
if(s_corrs(3) > max(s_corr_copy(:)))
    s_corr_copy = s_corr;
    s_temp_copy = s_temp;
end

results(3) = max(s_corrs);


disp(s_corrs);

%%


disp(results);
[M, I] = max(results);
if I == 1
    %Display the bounding box for the most correlated image using the
    %copied correlation matrix and resized template we kept track of
earlier
    disp(max(r_corr_copy));
    [ypeak_r, xpeak_r] = find(r_corr_copy==max(r_corr_copy(:)));
    yoffSet_r = ypeak_r-size(r_temp_copy,1);
    xoffSet_r = xpeak_r-size(r_temp_copy,2);
    hAx = axes;
    figure('Name', 'Rock Winner Template');
    imshow(im,'Parent', hAx);
    imrect(hAx, [xoffSet_r, yoffSet_r, size(r_temp_copy,2),
size(r_temp_copy,1)]);
    result = 'rock';
elseif I == 2
    %Display the bounding box for the most correlated image using the
    %copied correlation matrix and resized template we kept track of
earlier
    disp(max(p_corr_copy));
    [ypeak_p, xpeak_p] = find(p_corr_copy==max(p_corr_copy(:)));
    yoffSet_p = ypeak_p-size(p_temp_copy,1);
    xoffSet_p = xpeak_p-size(p_temp_copy,2);
    hAx = axes;
    figure('Name', 'Paper Winner Template');
    imshow(im,'Parent', hAx);
    imrect(hAx, [xoffSet_p, yoffSet_p, size(p_temp_copy,2),
size(p_temp_copy,1)]);
    result = 'paper';
elseif I == 3
    %Display the bounding box for the most correlated image using the
    %copied correlation matrix and resized template we kept track of
earlier
    disp(max(s_corr_copy));
    [ypeak_s, xpeak_s] = find(s_corr_copy==max(s_corr_copy(:)));
    yoffSet_s = ypeak_s-size(s_temp_copy,1);
    xoffSet_s = xpeak_s-size(s_temp_copy,2);
    hAx = axes;
    figure('Name', 'Scissors Winner Template');
    imshow(im,'Parent', hAx);
    imrect(hAx, [xoffSet_s, yoffSet_s, size(s_temp_copy,2),
size(s_temp_copy,1)]);
    result = 'scissors';
end
end
```