1. Solving a Jigsaw Puzzle

We present one possible algorithm for solving the jigsaw puzzle. Other clearly explained algorithms which achieve the same results are encouraged and also given full credit.

Step 1: We binarize the pieces image, with the result shown below.

Step 2: We extract an edge map from the binary image (result shown below), which we will use later to draw the boundaries of the jigsaw pieces on top of the reference image.

Step 3: When the jigsaw pieces are fully assembled, the assembled image may have a different size than the reference image. To account for this mismatch, we resize the reference image (maintaining the same aspect ratio) to have approximately the same number of pixels as the total number of white pixels in the binary pieces image.
**Step 4:** We perform region labeling on the binary pieces image.

**Step 5:** For each region in the binary pieces image, we compare the corresponding region in the original color pieces image to different regions in the reference image. Note that the regions being compared are non-rectangular in shape, because we utilize the regional shapes obtained from the segmentation in Step 1. We find the region in the reference image that best matches (has the smallest mean squared difference relative to) each piece region. For this matching region in the reference image, we (i) label that region with the same number that identifies the jigsaw piece, and (ii) overlay the boundary of the jigsaw piece using the edge map from Step 2. Below, we show the annotated solution of the jigsaw puzzle.
MATLAB Code:

% EE368/CS232
% Homework 5
% Problem: Jigsaw Puzzle
% Script by David Chen, Huizhong Chen

clc; clear all;

% Load images
imgPieces = im2double(imread('hw5_puzzle_pieces.jpg'));
imgPiecesGray = rgb2gray(imgPieces);
imgRef = im2double(imread('hw5_puzzle_reference.jpg'));

% Binarize pieces image
imgPiecesBin = double(imgPiecesGray < 0.99);
figure(1); clf; set(gcf, 'Color', 'w');
imshow(imgPiecesBin);

% Resize reference image to match area of pieces
piecesArea = sum(imgPiecesBin(:));
referenceArea = size(imgRef,1) * size(imgRef,2);
imgRef = imresize(imgRef, sqrt(piecesArea/referenceArea), 'bicubic');
[refH, refW, channels] = size(imgRef);

% Extract edge map
imgPiecesEdge = edge(imgPiecesBin, 'canny');
imgPiecesEdgeDil = imdilate(imgPiecesEdge, ones(3,3));
figure(2); clf; set(gcf, 'Color', 'w');
imshow(imgPiecesEdge);

% Perform region labeling
[imgLabels, numLabels] = bwlable(imgPiecesBin);
figure(3); clf; set(gcf, 'Color', 'w');
imshow(imgPieces); hold on;
figure(4); clf; set(gcf, 'Color', 'w');
imshow(imgRef); hold on;
for nLabel = 1:numLabels
    fprintf('Matching piece %d/%d\n', nLabel, numLabels);
    [y,x] = find(imgLabels == nLabel);
    minX = min(x); maxX = max(x);
    minY = min(y); maxY = max(y);
    centerX = 0.5*(minX + maxX);
    centerY = 0.5*(minY + maxY);
    pieceW = maxX - minX + 1;
    pieceH = maxY - minY + 1;
    piece = imgPieces(minY:maxY, minX:maxX);
    pieceBin = imgPiecesBin(minY:maxY, minX:maxX);
    pieceEdge = imgPiecesEdgeDil(minY:maxY, minX:maxX);
    % Find best matching location in reference image
    minError = inf;
    for leftX = 1:refW-pieceW+1
        for topY = 1:refH-pieceH+1
            imgRefCrop = imgRef(topY:topY+pieceH-1, leftX:leftX+pieceW-1, :);
            diff = sum(abs(piece - imgRefCrop), 3);
            diff = sum(sum(pieceBin .* diff));
            if diff < minError
                minError = diff;
                minErrorLeftX = leftX;
                minErrorTopY = topY;
            end
        end
    end
    % Annotate pieces image
    figure(3);
    label = sprintf('%d', nLabel);
    put_label(centerX-7, centerY, label);
    % Annotate reference image
    figure(4);
[yEdge, xEdge] = find(pieceEdge == 1);
put_points(xEdge+minErrorLeftX, yEdge+minErrorTopY);
put_label(minErrorLeftX+pieceW/2-7, minErrorTopY+pieceH/2, label);
pause(0.5);
end % nLabel

function put_label(x, y, label)
for dx = -2:2
    for dy = -2:2
        h = text(x+dx, y+dy, label);
        set(h, 'Color', 'k', 'FontSize', 20);
    end % dy
end % dx
h = text(x, y, label);
set(h, 'Color', 'y', 'FontSize', 20);
end

function put_points(xVec, yVec)
    h = plot(xVec, yVec, 'yo');
    set(h, 'MarkerSize', 1, 'MarkerFaceColor', 'y');
2. Eigenfaces and Fisherfaces for Recognition of Glasses

Part A:
Below, we show the mean face computed from all of the training samples.

Part B:
Below, we show the top 10 eigenfaces computed from the training images using the Sirovich and Kirby method. The first 6 eigenfaces appear to model the effects of large illumination variations across a face. Eigenfaces 7-10 begin to capture more of the subtle variations in local facial regions. In particular, eigenface 10 accentuates regions around the eyes and will be used in a later part of this problem for classifying between non-glasses and glasses.

The negatives of these 10 Eigenfaces are also valid solutions and are shown below.

Part C:
As mentioned in Part B, the 10th eigenface places higher weights around the regions of the eyes compared to some other facial regions, so we think it might be useful for distinguishing between
non-glasses and glasses. Below, we plot the histogram of the 10\textsuperscript{th} eigenface coefficients for non-glasses and glasses images in the training set. A threshold chosen by the MAP principle would be $T = 0.75$ and achieves an overall error rate of 25.5 percent over the training set.

Classification results using this threshold on the testing set are reported in the table below.

Part D:
In a 100-dimensional eigenface space, we perform Fisher Linear Discriminant Analysis with the two classes of non-glasses and glasses. The image below shows the resulting fisherface. This fisherface places higher weights around the borders of the glasses and lower weights in other facial regions.

The negative of this fisherface is also a valid solution and is shown below.
Below, we plot the histogram of the fisherface coefficients for non-glasses and glasses images in the training set. The two classes are now better separated than for the 10th Eigenface. A threshold of $T = 0.45$ is chosen by the MAP principle and achieves an overall error rate of 0 percent over the training set.

Classification results using this threshold on the testing set are reported below and contrasted with the results for the 10th eigenface.

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Non-Glass Images Classified Correctly</th>
<th>Number of Glass Images Classified Correctly</th>
<th>Overall Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fisherface</td>
<td>120 / 128</td>
<td>40 / 52</td>
<td>11.11 %</td>
</tr>
<tr>
<td>10th Eigenface</td>
<td>97 / 128</td>
<td>28 / 52</td>
<td>30.56 %</td>
</tr>
</tbody>
</table>

As can be observed from the classification results, the fisherface obtains a substantially lower overall error rate compared to the 10th eigenface. The fisherface uses a more discriminative linear subspace that is specially designed to separate the two glasses, while the eigenfaces were not designed explicitly for this classification task.

MATLAB Code:

```matlab
% EE368/CS232
% Homework 5
% Problem: Glasses Recognition
% Script by David Chen, Huizhong Chen
clc; clear;
height = 100; width = 88; numPixels = width*height;
trainFolders = {'non-glasses-training', 'glasses-training'};
trainIndices{1} = 0:152; trainIndices{2} = 0:66;
testFolders = {'non-glasses-testing', 'glasses-testing'};
testIndices{1} = 153:280; testIndices{2} = 67:118;
```
%% Collect training samples for PCA

imageMeanVector = 0;
for nClass = 1:2
    trainImages{nClass} = [];
    for idx = 1:length(trainIndices{nClass})
        fileName = sprintf('%s/%03d.jpg', trainFolders{nClass}, idx);
        img = im2double(imread(fileName));
        imgVector = img(:);
        trainImages{nClass}(:,end+1) = imgVector;
        imageMeanVector = imageMeanVector + imgVector;
    end
end
numSamples = length(trainIndices{1}) + length(trainIndices{2});
imageMeanVector = imageMeanVector / numSamples;
imageMean = reshape(imageMeanVector, [height width]);
figure(1); clf;
imshow(imageMean); title('Mean Face');
imwrite(imageMean, 'Mean_Face.png');

S = [];
for nClass = 1:2
    for nSample = 1:size(trainImages{nClass},2)
        imgVector = trainImages{nClass}(:,nSample) - imageMeanVector;
        S = [S imgVector];
    end
end

%% Estimate PCA eigenvectors by Sirovich-Kirby method

[V, D] = eig(S' * S);
PCAEigenvectors = V(:,end:-1:1); % Still need to calculate
for n = 1:size(PCAEigenvectors,2)
    PCAEigenvectors(:,n) = PCAEigenvectors(:,n) / norm(PCAEigenvectors(:,n));
end
numPCAEigenvectorsToKeep = 100;
PCAMatrix = PCAEigenvectors(:,1:numPCAEigenvectorsToKeep);'.
figure(2); clf;
for n = 1:10
    eigenface = reshape(PCAEigenvectors(:,n), [height width]);
    eigenface = (eigenface-min(eigenface(:)))/(max(eigenface(:))-min(eigenface(:)));
    subplot(2,5,n);
    imshow(eigenface); title(sprintf('Eigenface %d', n));
imwrite(eigenface, sprintf('Eigenface_%02d.png', n));
end

%% Collect training samples for LDA

PCAMeanVector = 0;
for nClass = 1:2
    classPCAMeanVector = 0;
    classNumSamples(nClass) = 0;
    classPCASamples(nClass) = [];
    for nSample = 1:size(trainImages{nClass},2)
        imgVector = trainImages{nClass}(:,nSample) - imageMeanVector;
        PCAVector = PCAMatrix * imgVector;
        classPCASamples{nClass}(:,end+1) = PCAVector;
        classPCAMeanVector = classPCAMeanVector + PCAVector;
        classNumSamples(nClass) = classNumSamples(nClass) + 1;
    end
    classPCAMeanVector = classPCAMeanVector / classNumSamples(nClass);
end
PCAMeanVector = PCAMeanVector / sum(classNumSamples); % should be approximately 0
RB = 0; RW = 0;
for nClass = 1:2
    deltaVector = classPCAMeanVectors(:,nClass) - PCAMeanVector;
    RB = RB + classNumSamples(nClass) * (deltaVector * deltaVector.');
end
for nSample = 1:size(classPCASamples{nClass},2)
deltaVector = classPCASamples{nClass}(:, nSample) - classPCAMeanVectors(:, nClass);
RW = RW + (deltaVector * deltaVector.');
end % nSample
end % nClass
figure(3); clf;
negativeMeanFace = reshape(PCAMatrix.' * classPCAMeanVectors(:,1), [height width]);
negativeMeanFace = (negativeMeanFace - min(negativeMeanFace(:)))/(max(negativeMeanFace(:)) - min(negativeMeanFace(:)));
positiveMeanFace = (positiveMeanFace - min(positiveMeanFace(:)))/(max(positiveMeanFace(:)) - min(positiveMeanFace(:))); subplot(1,2,1); imshow(negativeMeanFace); title('Mean Face, Class 1');
subplot(1,2,2); imshow(positiveMeanFace); title('Mean Face, Class 2');
imwrite(negativeMeanFace, 'Mean_Face_Class_1.png');
imwrite(positiveMeanFace, 'Mean_Face_Class_2.png');

%% Estimate LDA eigenvector
rankBetween = rank(RB);
[V,D] = eigs(RB, RW);
LDAEigenvectors = V(:,1:rankBetween);
figure(4); clf;
for i = 1:size(LDAEigenvectors,2)
  LDAEigenvectors(:,i) = LDAEigenvectors(:,i) / norm(LDAEigenvectors(:,i));
  fishervector = LDAEigenvectors(:,1);
  reconstruction = PCAVector * fishervector;
  reconstruction = reshape(reconstruction, [height width]);
  reconstruction = (reconstruction - min(reconstruction(:)))/(max(reconstruction(:)) - min(reconstruction(:)));
  imwrite(reconstruction, sprintf('Fisherface_%02d.jpg', i));
  imshow(reconstruction); title('Fisherface');
end

%% Create MAP detector with Fisherface
for nClass = 1:2
  LDACoefficients{nClass} = {};
  for nSample = 1:size(trainImages{nClass},2)
    imgVector = trainImages{nClass}(:,nSample) - imageMeanVector;
    PCAVector = PCAMatrix * imgVector;
    LDACoefficients{nClass}(end+1) = dot(PCAVector, LDAEigenvectors(:,1));
  end % nSample
  end % nClass
if mean(LDACoefficients{1}) > mean(LDACoefficients{2})
  LDACoefficients{1} = -LDACoefficients{1};
  LDACoefficients{2} = -LDACoefficients{2};
end
histBins = linspace(-5, 5, 50);
histNegative = hist(LDACoefficients{1}, histBins);
histPositive = hist(LDACoefficients{2}, histBins);
thresholdFisherface = 0.45;
figure(5); clf;
h = plot(histBins, histNegative, 'r-', ...
  histBins, histPositive, 'b-', ...
  thresholdFisherface*ones(1,100), linspace(0,60), 'k--'); grid on;
set(h, 'LineWidth', 2);
set(gca, 'FontSize', 14);
xlabel('Transform Coefficient'); ylabel('Probability'); title('Fisherface');
legend('Non-glasses', 'Glasses', 'Location', 'NW');
h = text(thresholdFisherface+0.01, 35, sprintf('T = %.2f', thresholdFisherface));
set(h, 'FontSize', 14);

%% Create MAP detector with Eigenface
chooseEigenface = 10;
for nClass = 1:2
  PCACoefficients{nClass} = {};
  for nSample = 1:size(trainImages{nClass},2)
    imgVector = trainImages{nClass}(:,nSample) - imageMeanVector;
    PCAVector = PCAMatrix * imgVector;
    PCACoefficients{nClass}(end+1) = PCAVector(chooseEigenface);
  end % nSample
  end % nClass
if mean(PCACoefficients{1}) > mean(PCACoefficients{2})
  PCACoefficients{1} = -PCACoefficients{1};
  PCACoefficients{2} = -PCACoefficients{2};
end
histBins = linspace(-5, 5, 50);
histNegative = hist(PCACoefficients{1}, histBins);
histPositive = hist(PCACoefficients{2}, histBins);
thresholdEigenface = 0.45;
figure(6); clf;
h = plot(histBins, histNegative, 'r-', ...
  histBins, histPositive, 'b-', ...
  thresholdEigenface*ones(1,100), linspace(0,60), 'k--'); grid on;
set(h, 'LineWidth', 2);
set(gca, 'FontSize', 14);
xlabel('Coefficient'); ylabel('Probability'); title('Eigenface');
legend('Non-glasses', 'Glasses', 'Location', 'NW');
h = text(thresholdEigenface+0.01, 35, sprintf('T = %.2f', thresholdEigenface));
set(h, 'FontSize', 14);
end % nSample
end % nClass
if mean(PCACoefficients{1}) > mean(PCACoefficients{2})
    PCACoefficients{1} = -PCACoefficients{1};
    PCACoefficients{2} = -PCACoefficients{2};
end
histBins = linspace(-5, 5, 50);
histNegative = hist(PCACoefficients{1}, histBins);
histPositive = hist(PCACoefficients{2}, histBins);
thresholdEigenface = 0.75;
figure(6); clf;
h = plot(histBins, histNegative, 'r-', ...
    histBins, histPositive, 'b-', ...)
    thresholdEigenface*ones(1,100), linspace(0,60), 'k--'); grid on;
set(h, 'LineWidth', 2);
set(gca, 'FontSize', 14);
xlabel('Transform Coefficient'); ylabel('Probability'); title('Eigenface');
legend('Non-glasses', 'Glasses', 'Location', 'NW');
h = text(thresholdEigenface+0.01, 35, sprintf('T = %.2f', thresholdEigenface));
set(h, 'FontSize', 14);

%% Measure testing accuracy with Fisherface
for nClass = 1:2
    LDACoefficients{nClass} = [];
    for idx = testIndices{nClass}
        fileName = sprintf('%s/%03d.jpg', testFolders{nClass}, idx);
        img = im2double(imread(fileName));
        imgVector = img(:) - imageMeanVector;
        PCAVector = PCAMatrix * imgVector;
        LDACoefficients{nClass}(end+1) = dot(PCAVector, LDAEigenvectors(:,1));
    end
end
if mean(LDACoefficients{1}) > mean(LDACoefficients{2})
    LDACoefficients{1} = -LDACoefficients{1};
    LDACoefficients{2} = -LDACoefficients{2};
end
for nClass = 1:2
    if nClass == 1
        numRight(nClass) = numel(find(LDACoefficients{nClass} < thresholdFisherface));
    else
        numRight(nClass) = numel(find(LDACoefficients{nClass} > thresholdFisherface));
    end
    disp(sprintf('Class %d: %d/%d correct', nClass, numRight(nClass), length(testIndices{nClass}))); % nClass
end

%% Measure testing accuracy with Eigenface
for nClass = 1:2
    PCACoefficients{nClass} = [];
    for idx = testIndices{nClass}
        fileName = sprintf('%s/%03d.jpg', testFolders{nClass}, idx);
        img = im2double(imread(fileName));
        imgVector = img(:) - imageMeanVector;
        PCAVector = PCAMatrix * imgVector;
        PCACoefficients{nClass}(end+1) = PCAVector(chooseEigenface);
    end
end
if mean(PCACoefficients{1}) > mean(PCACoefficients{2})
    PCACoefficients{1} = -PCACoefficients{1};
    PCACoefficients{2} = -PCACoefficients{2};
end
for nClass = 1:2
    if nClass == 1
        numRight(nClass) = numel(find(PCACoefficients{nClass} < thresholdEigenface));
    else
        numRight(nClass) = numel(find(PCACoefficients{nClass} > thresholdEigenface));
    end
    disp(sprintf('Class %d: %d/%d correct', nClass, numRight(nClass), length(testIndices{nClass}))); % nClass
end
end % nClass
3. Frequency Responses of Edge Templates

Part A:
The central difference operator has the following impulse responses:

\[ h_{\text{hor}}[x,y] = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad h_{\text{ver}}[x,y] = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \]

The frequency responses of the central difference impulse responses are

\[ H_{\text{hor}}(e^{j\omega_x}, e^{j\omega_y}) = e^{-j\omega_x} - e^{j\omega_x} = -2j \cdot \sin(\omega_x) \]

\[ H_{\text{ver}}(e^{j\omega_x}, e^{j\omega_y}) = -2j \cdot \sin(\omega_y) \]

Below, we plot the magnitudes of the frequency responses of the central difference operator. From these plots, we see that the highest magnitude response occurs at \( \omega_x = \pm \pi / 2 \) for the horizontal filter and at \( \omega_y = \pm \pi / 2 \) for the vertical filter. Frequencies which are completely suppressed are \( \omega_x = 0 \) and \( \omega_x = \pm \pi \) for the horizontal filter and \( \omega_y = 0 \) and \( \omega_y = \pm \pi \) for the vertical filter.

![Frequency Responses for Central Difference](image)

The Prewitt operator has the following separable impulse responses:

\[ h_{\text{hor}}[x,y] = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad h_{\text{ver}}[x,y] = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \]

Hence, the frequency responses are the products of 1-d frequency responses, including the central difference’s frequency responses that have been previously derived:

\[ H_{\text{hor}}(e^{j\omega_x}, e^{j\omega_y}) = -2j \cdot \sin(\omega_x) \cdot (1 + 2\cos(\omega_y)) \]

\[ H_{\text{ver}}(e^{j\omega_x}, e^{j\omega_y}) = -2j \cdot \sin(\omega_y) \cdot (1 + 2\cos(\omega_x)) \]

12
Below, we plot the magnitudes of the frequency responses of the Prewitt operator. In the high-pass direction, the behavior is very similar to that of the central difference. But now in the orthogonal low-pass direction, there is a severe attenuation of the high-frequency components. The highest magnitude responses occur at \((\omega_x, \omega_y) = (\pi / 2, 0)\) and \((\omega_x, \omega_y) = (-\pi / 2, 0)\) for the horizontal filter and at \((\omega_x, \omega_y) = (0, \pi / 2)\) and \((\omega_x, \omega_y) = (0, -\pi / 2)\) for the vertical filter.

The Sobel operator also has separable impulse responses:

\[
\begin{align*}
    h_{\text{hor}}[x,y] &= \left[ \begin{array}{cc} -1 & 0 \\ 0 & 1 \end{array} \right] \ast \left[ \begin{array}{c} 1 \\ 2 \\ 1 \end{array} \right] \\
    h_{\text{ver}}[x,y] &= \left[ \begin{array}{c} -1 \\ 0 \\ 1 \end{array} \right] \ast \left[ \begin{array}{ccc} 1 & 2 & 1 \end{array} \right]
\end{align*}
\]

The frequency responses of the Sobel impulse responses are

\[
\begin{align*}
    H_{\text{hor}}(e^{j\omega_x}, e^{j\omega_y}) &= -2j \cdot \sin(\omega_x) \cdot (2 \cos(\omega_y)) \\
    H_{\text{ver}}(e^{j\omega_x}, e^{j\omega_y}) &= -2j \cdot \sin(\omega_y) \cdot (2 \cos(\omega_x))
\end{align*}
\]

Below, we plot the magnitude of the frequency responses of the Sobel operator. The behavior is similar to that of the Prewitt operator. One difference is that the frequency components at \(\omega_x = \pm \pi\) and at \(\omega_y = \pm \pi\) are fully suppressed in the low-pass direction for the Sobel operator, while these frequency components are only partially attenuated for the Prewitt operator.
Finally, the Roberts operator has the following impulse responses:

\[ h_1[x, y] = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad h_2[x, y] = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]

The frequency responses of the Roberts impulse responses are

\[ H_1(e^{j\omega_x}, e^{j\omega_y}) = e^{-j\omega_x} - e^{-j\omega_y} \]
\[ H_2(e^{j\omega_x}, e^{j\omega_y}) = 1 - e^{j\omega_x-j\omega_y} \]

Below, we plot the magnitudes of the frequency responses of the Roberts operator. The high-pass directions now occur in the two diagonal directions. Otherwise, the behavior is qualitatively similar to that of the central difference operator.
The Gaussian filter provides a strong attenuation of the high-frequency components, on top of any existing attenuation of these high-frequency components that was already provided by the edge template in Part A. When combined with the Gaussian filter, the shapes of the frequency responses of the central difference, Prewitt, and Sobel operators become much more similar to each other than in Part A.
MATLAB Code:

```matlab
% EE368/CS232
% Homework 5
% Problem: Edge Templates
% Script by David Chen, Huizhong Chen

clc; clear all; close all;

%% Part A
% Frequency response of central difference
[wx,wy] = meshgrid(linspace(-pi,pi,200), linspace(-pi,pi,200));
Hx_central = -2*1i*sin(wx);
Hy_central = -2*1i*sin(wy);
figure(1); clf; set(gcf, 'Position', [50 50 600 300], 'Color', 'w');
subplot(1,2,1); mesh(wx/pi, wy/pi, abs(Hx_central)); grid on;
axis([-1 1 -1 1 0 2]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_h_o_r |');
subplot(1,2,2); mesh(wx/pi, wy/pi, abs(Hy_central)); grid on;
axis([-1 1 -1 1 0 2]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_v_e_r |');

% Frequency response of Prewitt
Hx_Prewitt = -2*1i*sin(wx) .* (1 + 2*cos(wy));
Hy_Prewitt = -2*1i*sin(wy) .* (1 + 2*cos(wx));
figure(2); clf; set(gcf, 'Position', [100 100 600 300], 'Color', 'w');
subplot(1,2,1); mesh(wx/pi, wy/pi, abs(Hx_Prewitt)); grid on;
axis([-1 1 -1 1 0 6]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_h_o_r |');
subplot(1,2,2); mesh(wx/pi, wy/pi, abs(Hy_Prewitt)); grid on;
axis([-1 1 -1 1 0 6]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_v_e_r |');

% Frequency response of Sobel
Hx_Sobel = -2*1i*sin(wx) .* (2 + 2*cos(wy));
Hy_Sobel = -2*1i*sin(wy) .* (2 + 2*cos(wx));
figure(3); clf; set(gcf, 'Position', [150 150 600 300], 'Color', 'w');
subplot(1,2,1); mesh(wx/pi, wy/pi, abs(Hx_Sobel)); grid on;
axis([-1 -1 1 0 8]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_h_o_r |');
subplot(1,2,2); mesh(wx/pi, wy/pi, abs(Hy_Sobel)); grid on;
axis([-1 -1 1 0 8]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_v_e_r |');

% Frequency response of Roberts
Hx_Roberts = exp(-1i*wx) - exp(-1i*wy);
Hy_Roberts = 1 - exp(-1i*wx).*exp(-1i*wy);
figure(4); clf; set(gcf, 'Position', [200 200 600 300], 'Color', 'w');
```
subplot(1,2,1); mesh(wx/pi, wy/pi, abs(Hx_Roberts)); grid on;
axis([-1 1 -1 1 0 2]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_1 |');
subplot(1,2,2); mesh(wx/pi, wy/pi, abs(Hy_Roberts)); grid on;
axis([-1 1 -1 1 0 2]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_2 |');

%% Part B

% Frequency response of Gaussian
sigma = 1;
g = fspecial('gaussian', [6*sigma+1 6*sigma+1], sigma);
G = freqz2(g, 200, 200);
figure(5); clf; set(gcf, 'Position', [250 250 300 300], 'Color', 'w');
mesh(wx/pi, wy/pi, abs(G)); grid on;
axis([-1 1 -1 1 0 1]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| G |');

% Frequency response of central difference x Gaussian
figure(1); clf; set(gcf, 'Position', [50 50 600 300], 'Color', 'w');
subplot(1,2,1); mesh(wx/pi, wy/pi, abs(Hx_central .* G)); grid on;
axis([-1 1 -1 1 0 2]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_{h_0}G |');
subplot(1,2,2); mesh(wx/pi, wy/pi, abs(Hy_central .* G)); grid on;
axis([-1 1 -1 1 0 2]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_{v_e}G |');

% Frequency response of Prewitt x Gaussian
figure(2); clf; set(gcf, 'Position', [100 100 600 300], 'Color', 'w');
subplot(1,2,1); mesh(wx/pi, wy/pi, abs(Hx_Prewitt .* G)); grid on;
axis([-1 1 -1 1 0 6]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_{h_0}G |');
subplot(1,2,2); mesh(wx/pi, wy/pi, abs(Hy_Prewitt .* G)); grid on;
axis([-1 1 -1 1 0 6]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_{v_e}G |');

% Frequency response of Sobel
figure(3); clf; set(gcf, 'Position', [150 150 600 300], 'Color', 'w');
subplot(1,2,1); mesh(wx/pi, wy/pi, abs(Hx_Sobel .* G)); grid on;
axis([-1 1 -1 1 0 8]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_{h_0}G |');
subplot(1,2,2); mesh(wx/pi, wy/pi, abs(Hy_Sobel .* G)); grid on;
axis([-1 1 -1 1 0 8]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_{v_e}G |');

% Frequency response of Roberts
figure(4); clf; set(gcf, 'Position', [200 200 600 300], 'Color', 'w');
subplot(1,2,1); mesh(wx/pi, wy/pi, abs(Hx_Roberts .* G)); grid on;
axis([-1 1 -1 1 0 2]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_1 |');
subplot(1,2,2); mesh(wx/pi, wy/pi, abs(Hy_Roberts .* G)); grid on;
axis([-1 1 -1 1 0 2]); set(gca, 'FontSize', 14);
xlabel('\omega_x / \pi'); ylabel('\omega_y / \pi'); zlabel('| H_2 |');
4. Document Image Restoration

This is one possible solution. Below, we show a binarized result with Otsu’s method.

From this binarized image, we calculate (i) a Hough transform and (ii) a histogram of orientations for the tallest peaks in the Hough transform. Both results are shown below. The two spikes in the orientation histogram correspond to the long lines in the table.
Below, we show the result of rotating a negative version of the document image by 2 degrees counterclockwise, corresponding to the 2-degree deviation from vertical/horizontal for the spikes in the orientation histogram.

<table>
<thead>
<tr>
<th>CONTRACTOR</th>
<th>LICENSE</th>
<th>CERTIFICATE OF INSURANCE</th>
<th>FULL YEAR FEE</th>
<th>HALF YEAR FEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Contractor/Peace</td>
<td>X</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Excavator/Concrete/Masonry</td>
<td>X</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Carpenter</td>
<td>X</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Plumber/Linear Sprinkler</td>
<td>Illinois Dept. of Public Health Registration</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Saver</td>
<td>X</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Electrician</td>
<td>* Fiber License</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Communications Contractor</td>
<td>X</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>HVAC</td>
<td>X</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Roofer</td>
<td>Roofing License issued by State of Illinois</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Iron or Steel</td>
<td>X</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Fire Protection/Sprinkler</td>
<td>Sprinkler License</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Fire Protection/Alarm</td>
<td>Alarm License</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Paving</td>
<td>X</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
<tr>
<td>Elevator</td>
<td>Elevator Co. License</td>
<td>X</td>
<td>$75</td>
<td>$77.50</td>
</tr>
</tbody>
</table>

To repair the broken horizontal and vertical lines, we can use a combination of flat erosion and closing. First, to repair the broken vertical lines, we perform flat erosion with a structuring element that is 35-pixel vertical line, which removes the horizontal lines and text characters but keeps the vertical lines. The erosion result is shown below.
To this erosion result, we apply flat closing (dilation followed by erosion) with a structuring element that is a 201-pixel vertical line, which is selected to be long enough to close all the vertical gaps. The closing result is shown below.

We compute the pixel-wise maximum of this closing result and the negative document image to obtain the following result. All the broken vertical lines of the table have been successfully repaired without affecting the appearance of other features (e.g., text) in the image.

<table>
<thead>
<tr>
<th>CONTRACTOR</th>
<th>LICENSE</th>
<th>CERTIFICATE OF INSURANCE</th>
<th>FULL YEAR FEE</th>
<th>HALF YEAR FEE</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Contractor/Fence</td>
<td>X</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Excavator/Concrete/Masonry</td>
<td>X</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Carpenter</td>
<td>X</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Plumber/Lawn Sprinkler</td>
<td>Illinois Dept. of Public Health Registration</td>
<td>$75</td>
<td>$37.50</td>
<td></td>
</tr>
<tr>
<td>Searer</td>
<td>X</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Electrician</td>
<td>* Elect. License</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Communications Contractor</td>
<td>X</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>HVAC</td>
<td>X</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Roofer</td>
<td>Roofing License issued by State of Illinois</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Iron or Steel</td>
<td>X</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Fire Protection/Sprinkler</td>
<td>Sprinkler License</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Fire Protection/Alarm</td>
<td>Alarm License</td>
<td></td>
<td>$0</td>
<td>$0</td>
</tr>
<tr>
<td>Painting</td>
<td>X</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
<tr>
<td>Elevator</td>
<td>Elevator Co. License</td>
<td></td>
<td>$75</td>
<td>$37.50</td>
</tr>
</tbody>
</table>
To repair the broken horizontal lines, we first perform flat erosion with a structuring element that is a 49-pixel horizontal line, which removes the vertical lines and text characters but keeps the horizontal lines. The erosion result is shown below.

![Erosion Result]

To this erosion result, we apply flat closing with a structuring element that is a 301-pixel horizontal line, which is selected to be long enough to close all the horizontal gaps. The closing result is shown below.

![Closing Result]
Taking the pixel-wise maximum of the closing result and the negative image where the vertical lines had already been repaired, we obtain the following image. Now, all the horizontal lines are also repaired.

Inverting the image intensities, we get the fully repaired document image shown below.

MATLAB Code:

```
% EE368/CS232
% Homework 5
% Problem: Document Restoration
% Script by David Chen, Huizhong Chen

clc; clear all; close all;

% Load image
```
img = im2double(imread('hw5_insurance_form.jpg'));
[height, width] = size(img);
figure(1); clf; warning off; imshow(img); warning on;

% Binarize image
otsuLevel = graythresh(uint8(255*img))
imgBin = img < otsuLevel;
figure(2); clf; warning off; imshow(imgBin); warning on;

% Calculate Hough transform
thetaVec = -90 : 0.5 : 89.5;
[H, theta, rho] = hough(imgBin, 'Theta', thetaVec);
numPeaks = 40;
thresh = 0.4*max(H(:));
peaks = houghpeaks(H, numPeaks, 'Threshold', thresh);
thetaHist = hist(theta(peaks(:,2)), thetaVec);
figure(3); clf;
subplot(2,1,1);
warning off; imshow(imadjust(mat2gray(H)), 'XData', theta, 'YData', rho, ... 'InitialMagnification', 'fit'); axis on; axis normal; hold on; warning on;
plot(theta(peaks(:,2)), rho(peaks(:,1)), 'ys');
xlabel('
\theta (\text{degrees})'); ylabel('
\rho');
subplot(2,1,2);
bar(thetaVec, thetaHist); grid on;
axis([min(thetaVec) max(thetaVec) 0 max(thetaHist)+2]);
xlabel('
\theta (\text{degrees})'); ylabel('Number of Peaks');
[maxHist, maxIdx] = max(thetaHist);
maxTheta = thetaVec(maxIdx);
rotTheta = 90 + maxTheta;

% Rotate image to upright orientation
imgRotNeg = imrotate(1-img, rotTheta, 'bilinear', 'crop');
imgRot = 1 - imgRotNeg;
figure(4); clf; warning off; imshow(imgRotNeg); warning on;

% Erode with vertical SE, then close with vertical SE
SE = strel('arbitrary', ones(35,1));
imgRotNegErode = imerode(imgRotNeg, SE);
figure(5); clf; warning off; imshow(imgRotNegErode); warning on;
SE = strel('arbitrary', ones(201,1));
imgRotNegErodeClose = imclose(imgRotNegErode, SE);
figure(6); clf; warning off; imshow(imgRotNegErodeClose); warning on;
[SE, SE] = strel('arbitrary', ones(1,49));
figur

% Erode with horizontal SE, then close with horizontal SE
SE = strel('arbitrary', ones(35,1));
imgRotNegErode = imerode(imgRotNeg, SE);
figure(8); clf; warning off; imshow(imgRotNegErode); warning on;
SE = strel('arbitrary', ones(1,301));
imgRotNegErodeClose = imclose(imgRotNegErode, SE);
figure(9); clf; warning off; imshow(imgRotNegErodeClose); warning on;
imgRotNeg = max(imgRotNeg, imgRotNegErodeClose);
figure(10); clf; warning off; imshow(imgRotNeg); warning on;

% Show final result
imgRot = 1 - imgRotNeg;
figure(11); clf; warning off; imshow(imgRot); warning on;