Alperen Bağ 2018400279

CMPE 362 Project 3

Question 1

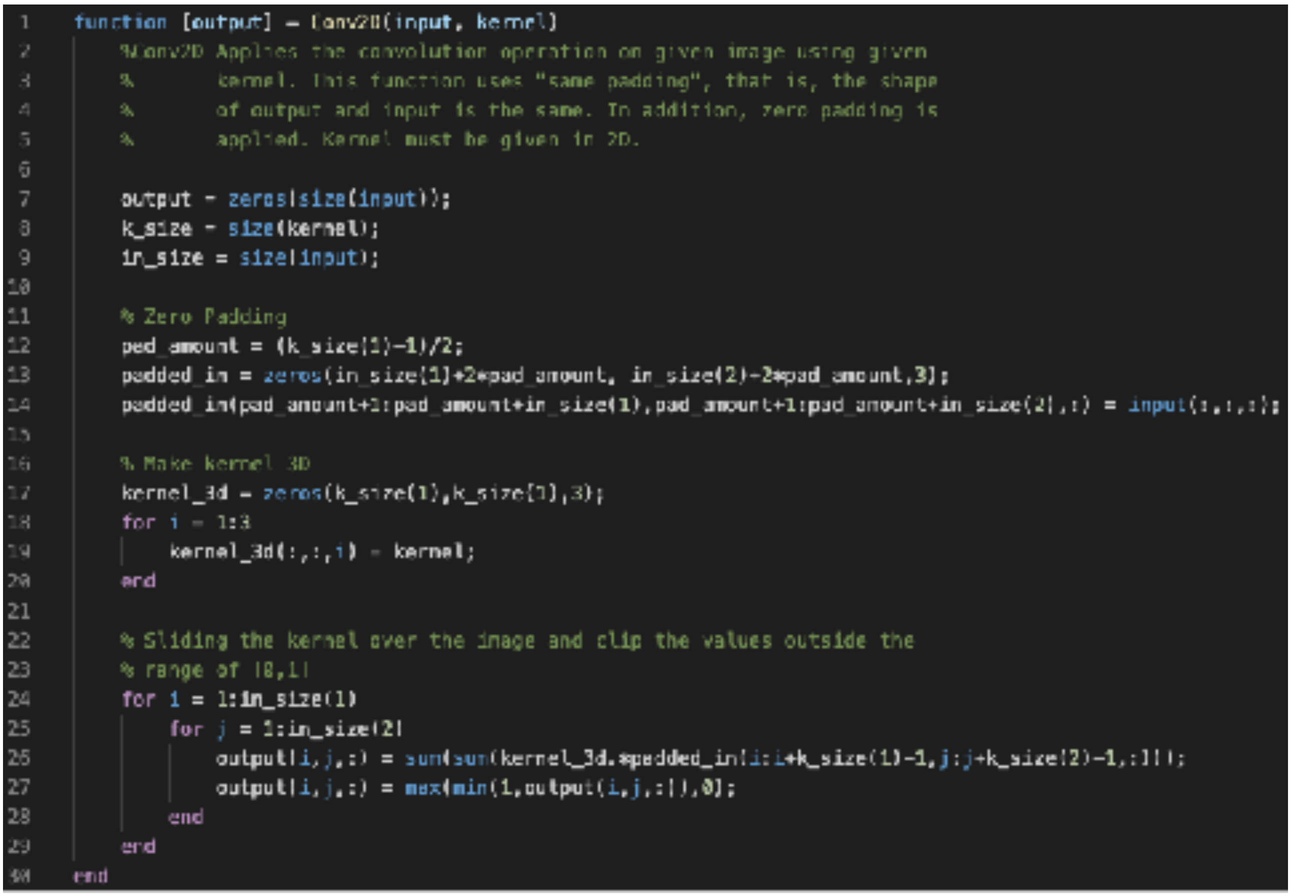
Convolution Operation Implementation

In this project, 2-dimensional convolution operation is implemented as a MATLAB function.

Basically, the function takes an image (matrix) on which convolution will be applied, a kernel which will be used as kernel for the convolution and slid over the input image with a stride of 1 unit.

The 2-D convolution function applies zero padding to input image and keeps the shape of the output image as the same of the input image.

Implementation

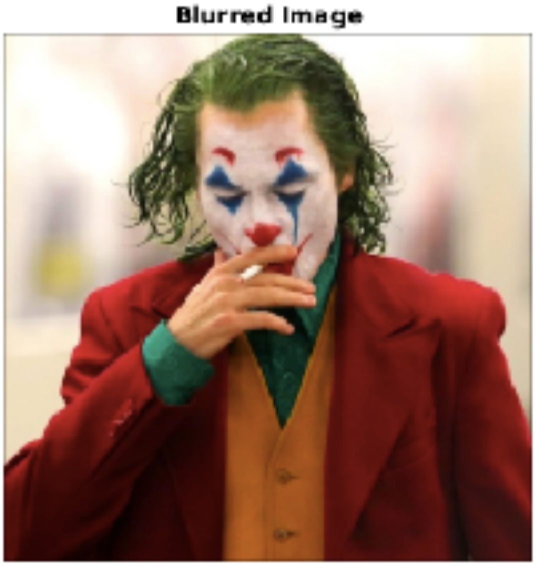


```matlab
function [output] = Conv2D(input, kernel)
    %Conv2D Applies the convolution operation on given image using given
    %       kernel. This function uses "same padding", that is, the shape
    %       of output and input is the same. In addition, zero padding is
    %       applied. Kernel must be given in 2D.

    output = zeros(size(input));
    k_size = size(kernel);
    in_size = size(input);

    % Zero Padding
    pad_amount = (k_size(1)-1)/2;
    padded_in = zeros(in_size(1)+2*pad_amount, in_size(2)+2*pad_amount,3);
    padded_in(pad_amount+1:pad_amount+in_size(1),pad_amount+1:pad_amount+in_size(2),:) = input(:,:,:);

    % Make kernel 3D
    kernel_3d = zeros(k_size(1),k_size(1),3);
    for i = 1:3
        kernel_3d(:,:,i) = kernel;
    end

    % Sliding the kernel over the image and clip the values outside the
    % range of [0,1]
    for i = 1:in_size(1)
        for j = 1:in_size(2)
            output(i,j,:) = sum(sum(kernel_3d.*padded_in(i:i+k_size(1)-1,j:j+k_size(2)-1,:)));
            output(i,j,:) = max(min(1,output(i,j,:)),0);
        end
    end
end
```

Blurring Image

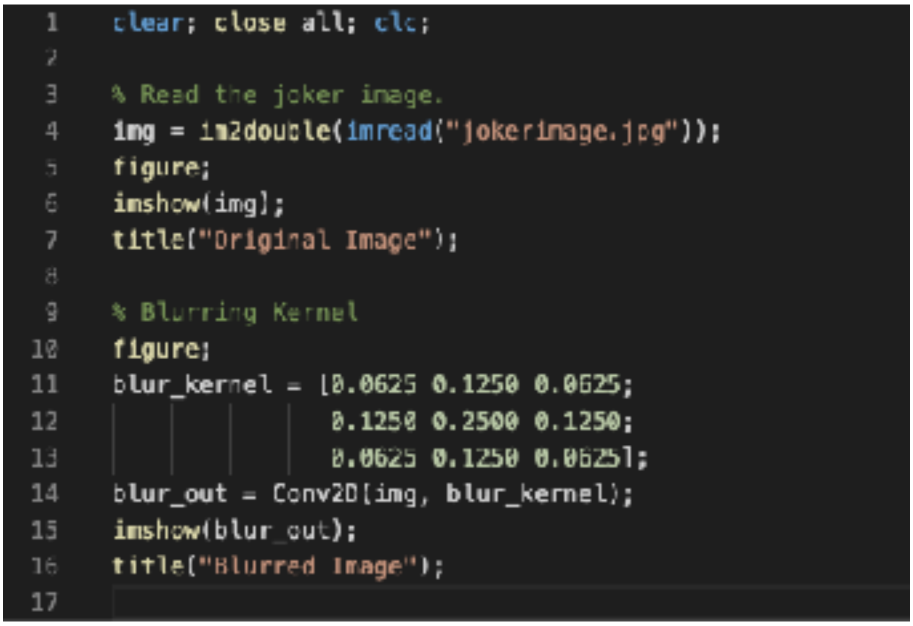
Using the convolution function explained above, an image can be blurred with the right kernel matrix as follows. The blurring kernel can be found in the implementation.

Original Image



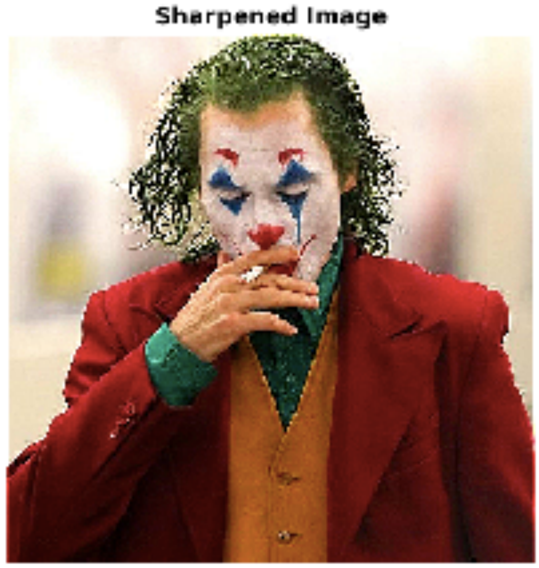
Blurred Image

Implementation

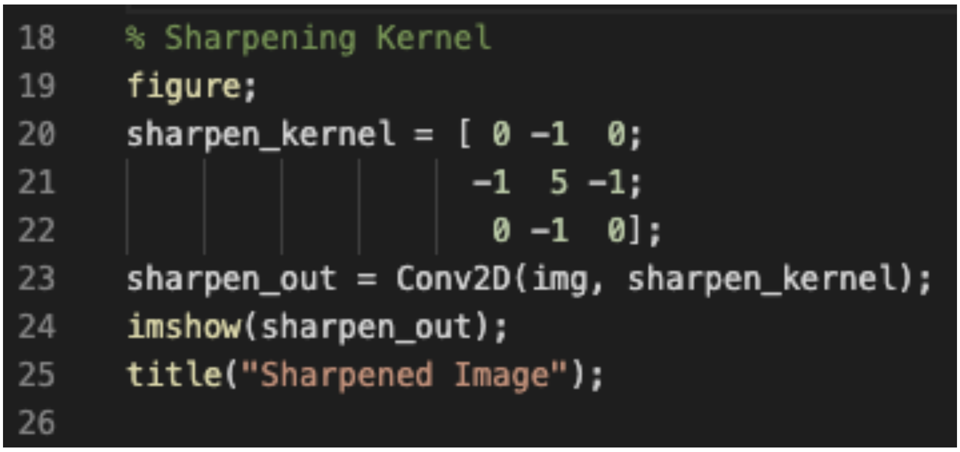


```
clear; close all; clc;

% Read the joker image.
img = im2double(imread("jokerimage.jpg"));
figure;
imshow(img);
title("Original Image");

% Blurring Kernel
figure;
blur_kernel = [0.0625 0.1250 0.0625;
               0.1250 0.2500 0.1250;
               0.0625 0.1250 0.0625];
blur_out = Conv2D(img, blur_kernel);
imshow(blur_out);
title("Blurred Image");

```

Sharpening Image

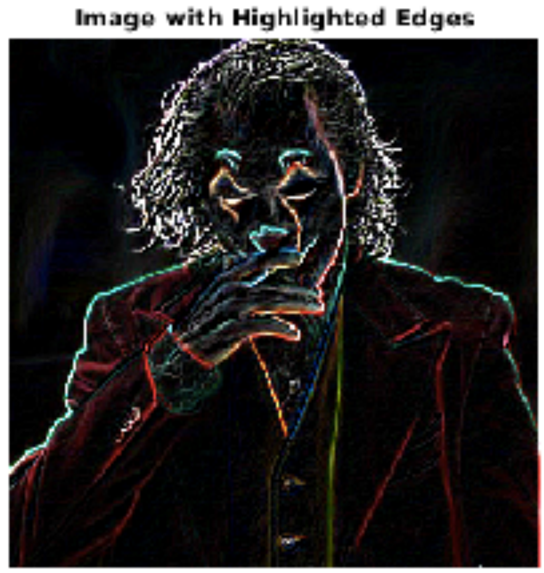


Implementation



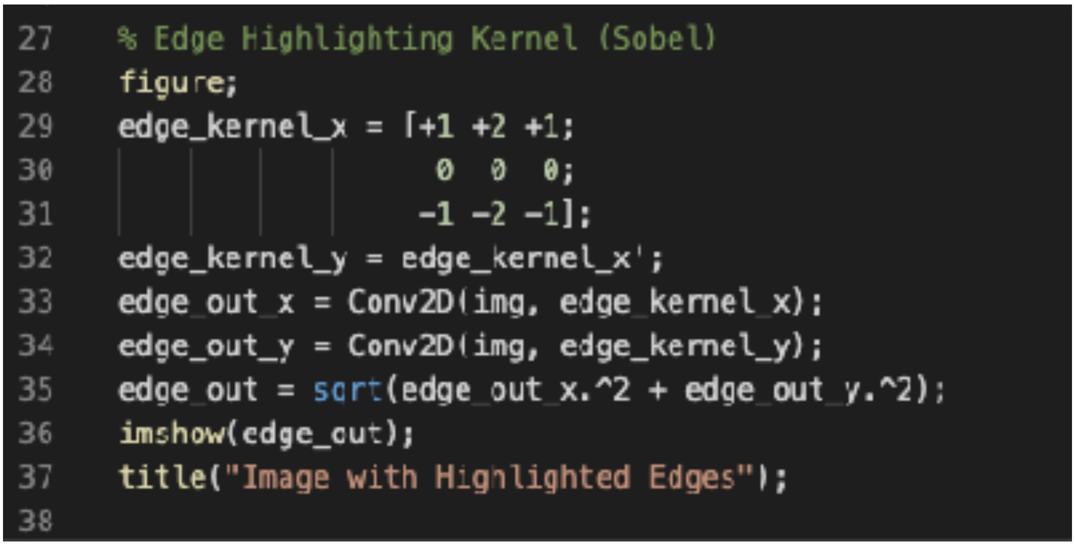
```
% Sharpening Kernel
figure;
sharpen_kernel = [ 0 -1  0;
                  -1  5 -1;
                   0 -1  0];
sharpen_out = Conv2D(img, sharpen_kernel);
imshow(sharpen_out);
title("Sharpened Image");

```

Edge Highlighting



Implementation

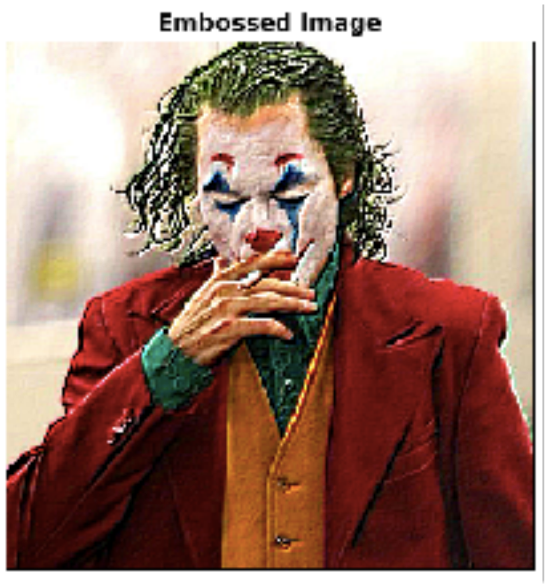
Sobel kernel is utilized to detect edges in both directions.



```
% Edge Highlighting Kernel (Sobel)
figure;
edge_kernel_x = [+1 +2 +1;
                  0  0  0;
                 -1 -2 -1];
edge_kernel_y = edge_kernel_x';
edge_out_x = Conv2D(img, edge_kernel_x);
edge_out_y = Conv2D(img, edge_kernel_y);
edge_out = sqrt(edge_out_x.^2 + edge_out_y.^2);
imshow(edge_out);
title("Image with Highlighted Edges");
```

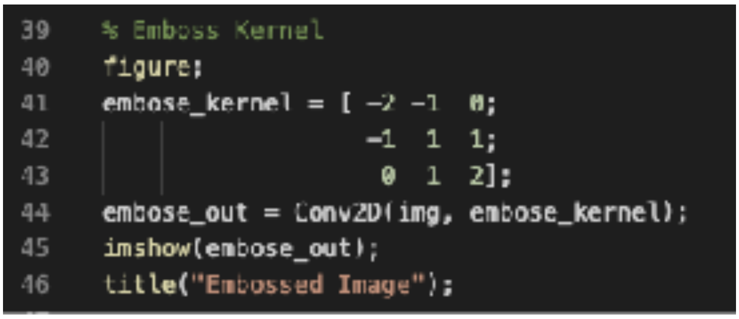
Embossing Image

Original Image



Embossed Image

Implementation



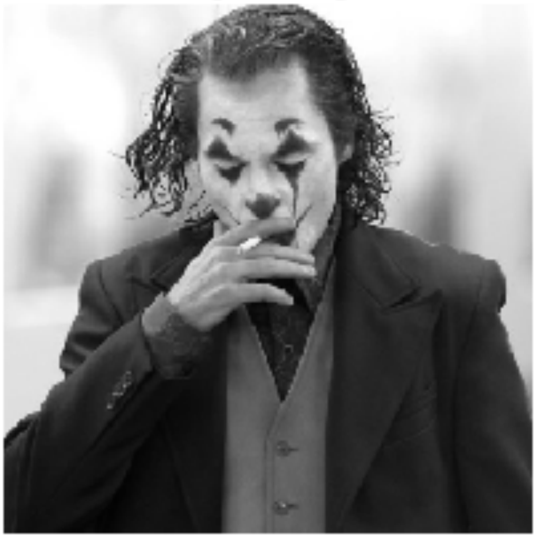
```
% Emboss Kernel
figure;
embose_kernel = [ -2 -1  0;
                  -1  1  1;
                   0  1  2];
embose_out = Conv2D(img, embose_kernel);
imshow(embose_out);
title("Embossed Image");
```

Question 2

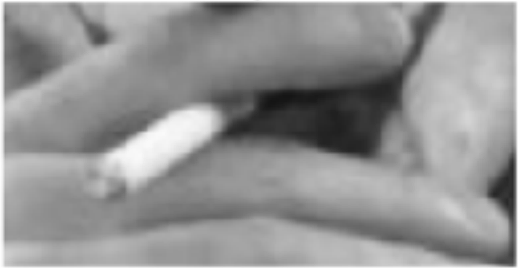
In this question, *Point Feature Matching* algorithm is applied on the Joker image. I used the code given in official MATLAB tutorial for object detection, which was also referred in our assignment file. After finding the coordinates of the box containing the cigarette in the original image using this algorithm, this bounding box is replaced with a flower image which is reshaped to the shape of the bounding box.

Output Images Step-By-Step

Original Image

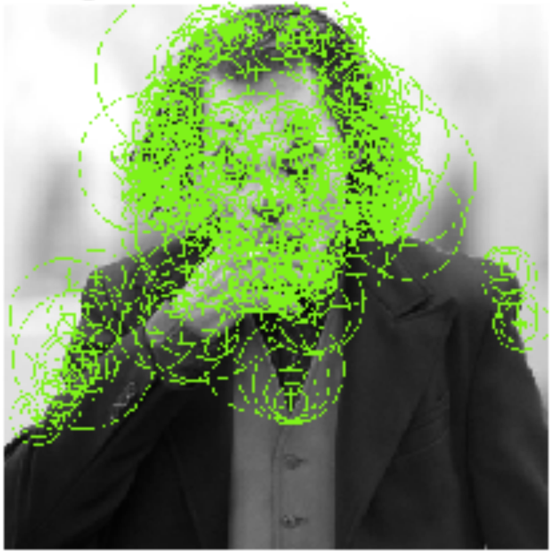


Object

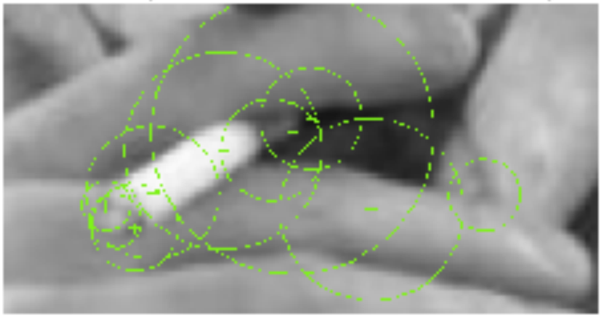


In the first step, feature points are found for the whole image and the object image that is to be detected.

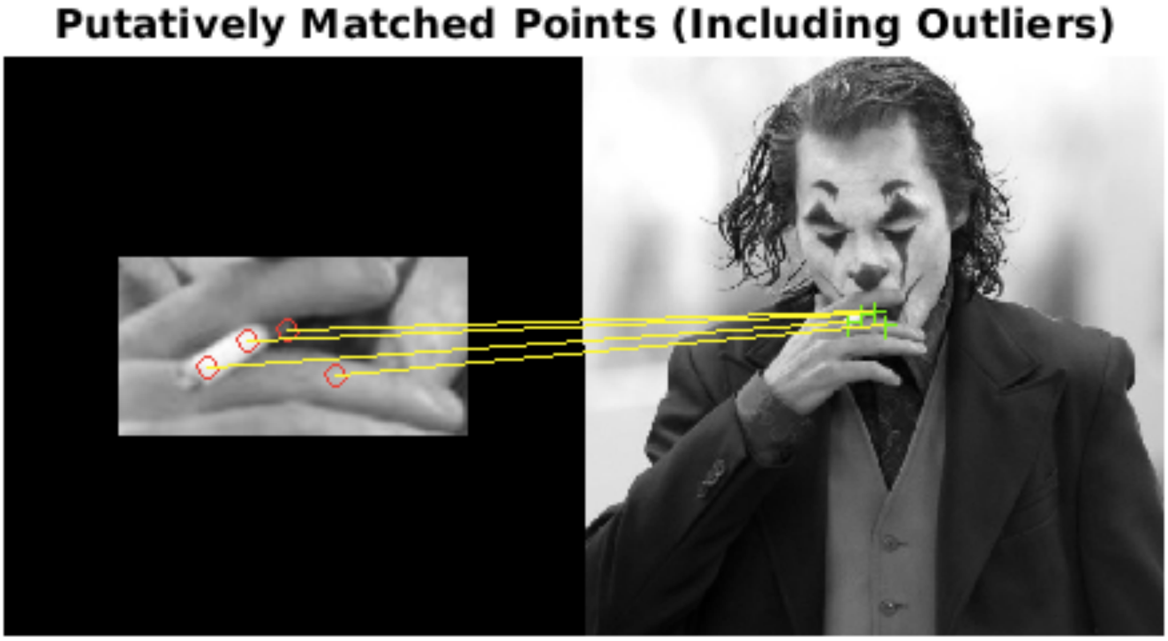
300 Strongest Feature Points from Scene Image



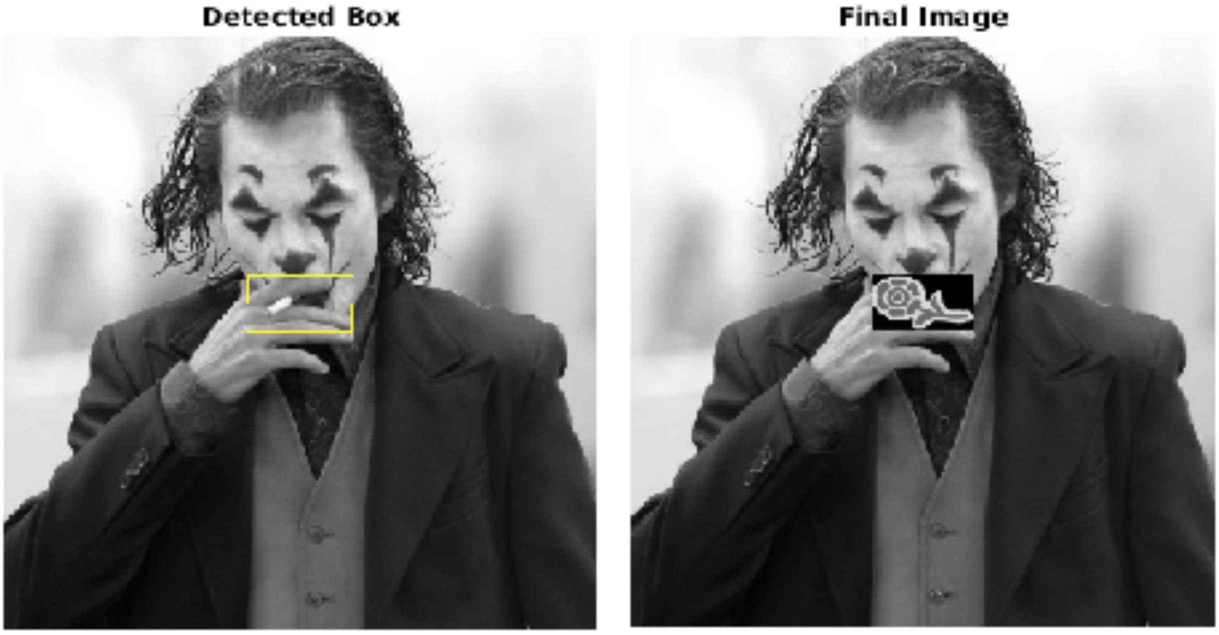
100 Strongest Feature Points from Box Image



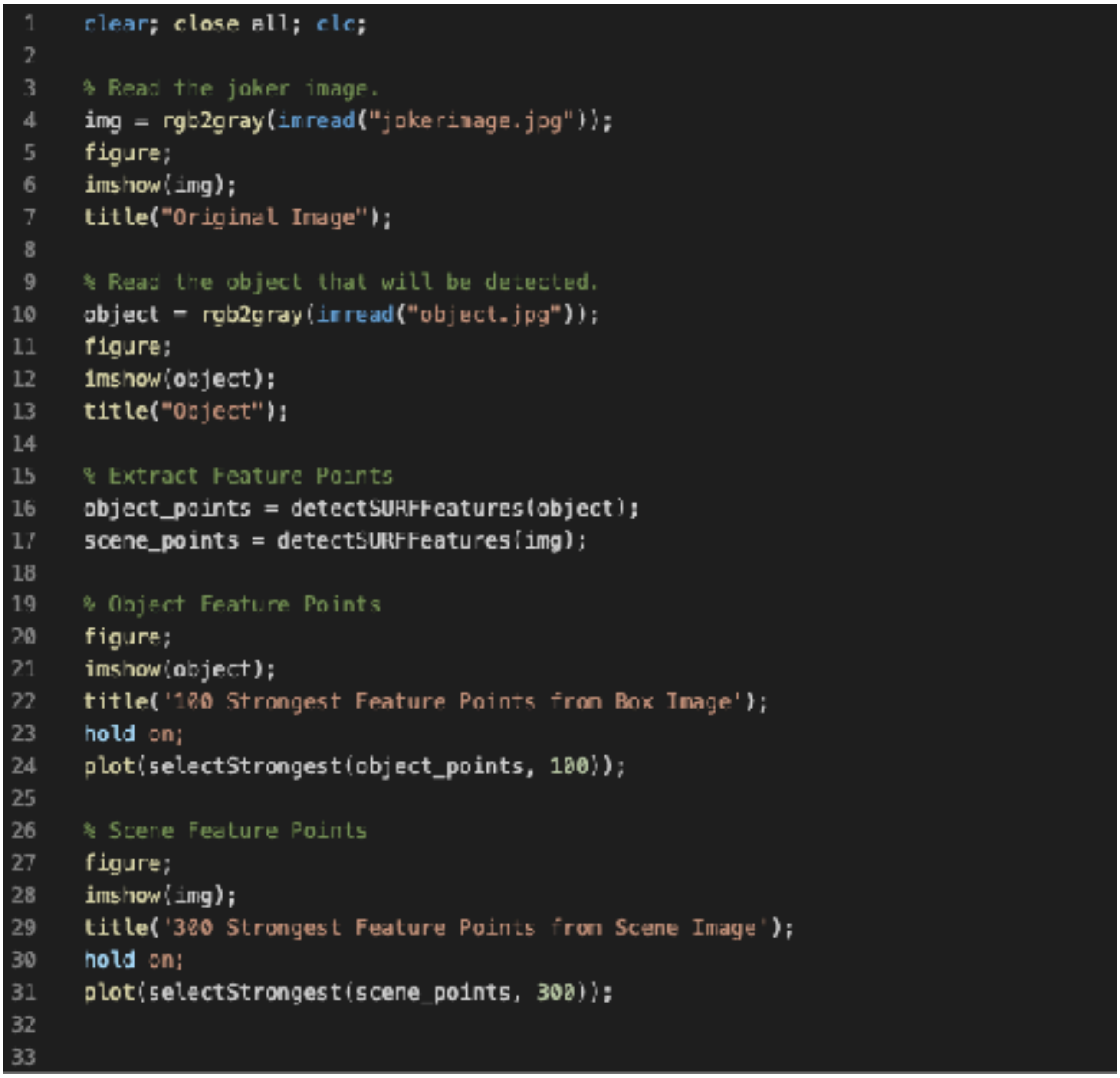
Then, detected points obtained from each image are matched in order to predict the location of the object in the whole image. One should be aware of the probability of getting outlier points. Fortunately, we did not have any outliers, however, outlier points should be also detected in general case.



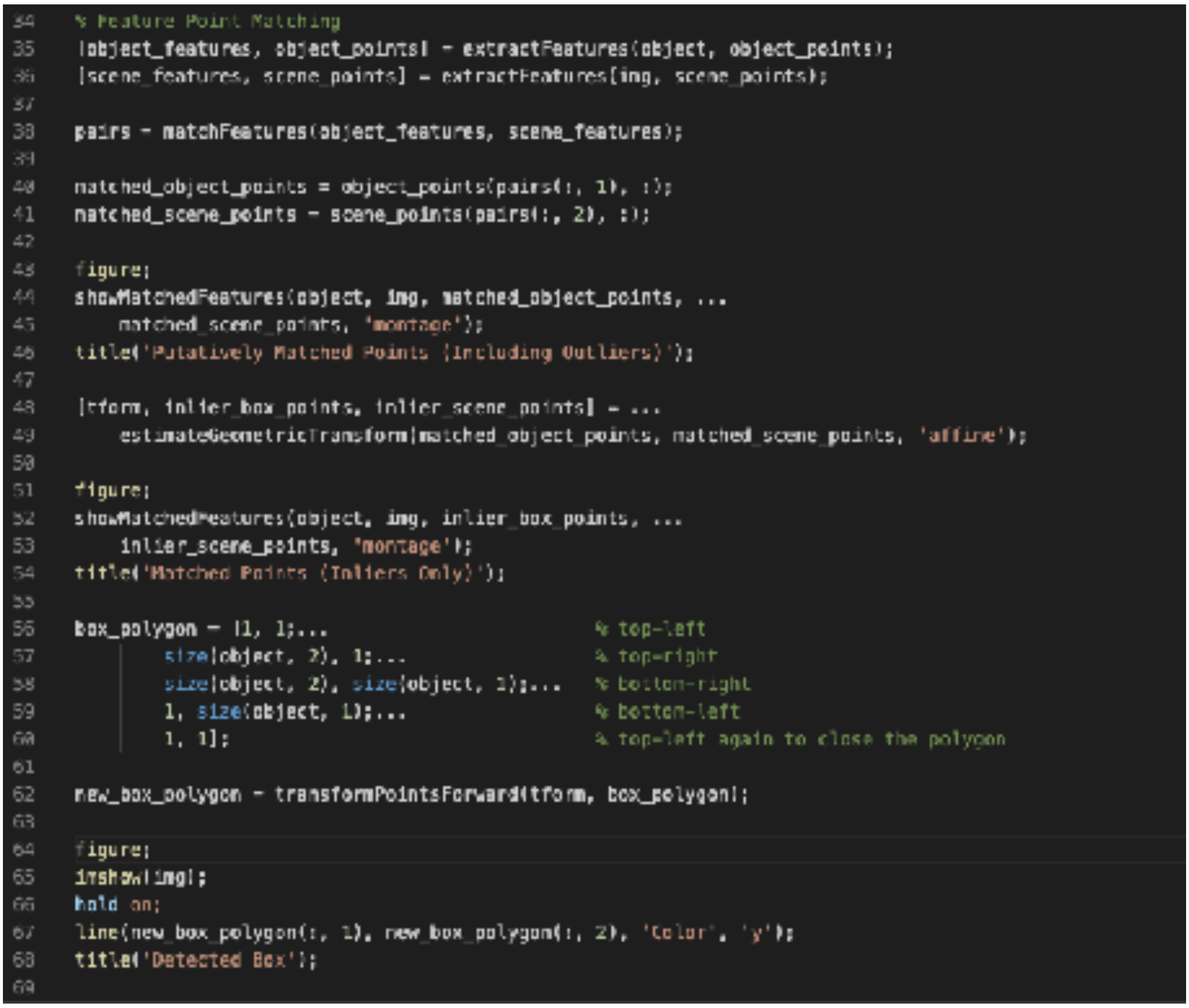
Now that we have the bounding box, we can put the flower image on the cigarette.



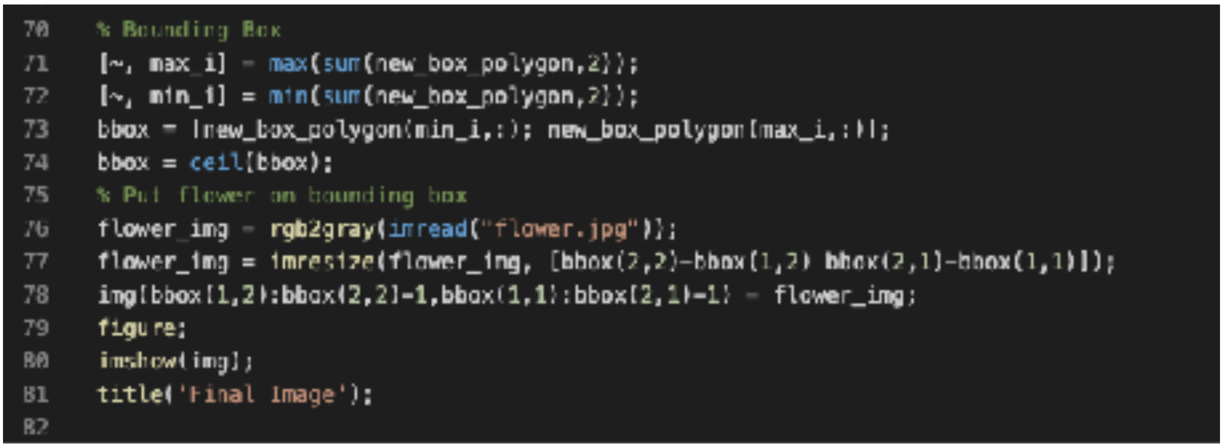
Implementation



```
clear; close all; clc;

% Read the joker image.
img = rgb2gray(imread("jokerimage.jpg"));
figure;
imshow(img);
title("Original Image");

% Read the object that will be detected.
object = rgb2gray(imread("object.jpg"));
figure;
imshow(object);
title("Object");

% Extract Feature Points
object_points = detectSURFFeatures(object);
scene_points = detectSURFFeatures(img);

% Object Feature Points
figure;
imshow(object);
title('100 Strongest Feature Points from Box Image');
hold on;
plot(selectStrongest(object_points, 100));

% Scene Feature Points
figure;
imshow(img);
title('300 Strongest Feature Points from Scene Image');
hold on;
plot(selectStrongest(scene_points, 300));
```



```
% Feature Point Matching
[object_features, object_points] = extractFeatures(object, object_points);
[scene_features, scene_points] = extractFeatures(img, scene_points);

pairs = matchFeatures(object_features, scene_features);

matched_object_points = object_points(pairs(:, 1), :);
matched_scene_points = scene_points(pairs(:, 2), :);

figure;
showMatchedFeatures(object, img, matched_object_points, ...
    matched_scene_points, 'montage');
title('Putatively Matched Points (Including Outliers)');

[tform, inlier_box_points, inlier_scene_points] = ...
    estimateGeometricTransform(matched_object_points, matched_scene_points, 'affine');

figure;
showMatchedFeatures(object, img, inlier_box_points, ...
    inlier_scene_points, 'montage');
title('Matched Points (Inliers Only)');

box_polygon = [1, 1;...                           % top-left
        size(object, 2), 1;...                    % top-right
        size(object, 2), size(object, 1);...      % bottom-right
        1, size(object, 1);...                    % bottom-left
        1, 1];                                    % top-left again to close the polygon

new_box_polygon = transformPointsForward(tform, box_polygon);

figure;
imshow(img);
hold on;
line(new_box_polygon(:, 1), new_box_polygon(:, 2), 'Color', 'y');
title('Detected Box');
```



```
% Bounding Box
[~, max_i] = max(sum(new_box_polygon,2));
[~, min_i] = min(sum(new_box_polygon,2));
bbox = [new_box_polygon(min_i,:); new_box_polygon(max_i,:)];
bbox = ceil(bbox);
% Put flower on bounding box
flower_img = rgb2gray(imread("flower.jpg"));
flower_img = imresize(flower_img, [bbox(2,2)-bbox(1,2) bbox(2,1)-bbox(1,1)]);
img(bbox(1,2):bbox(2,2)-1,bbox(1,1):bbox(2,1)-1) = flower_img;
figure;
imshow(img);
title('Final Image');
```