CmpE 362 Introduction to Signals for Computer Engineers

Spring 2020

Homework 3

Emilcan ARICAN
2016400231

Solution 1

Functions

apply_effect: First separates the color channels. Then takes the size of the given matrices. Adds padding to the image. Applies convolution to image and kernel matrix. Finally merges the color channels.

my_conv2: Takes the sizes of the kernel matrix and the image. Creates and empty array in order to fill with the calculated values. Iterates over the image and calculates the weighted sums and saves them into the new_img matrix.

add_padding: Takes the size of the kernel matrix. Calculated the needed gap size. extends the matrix by that many zeros.

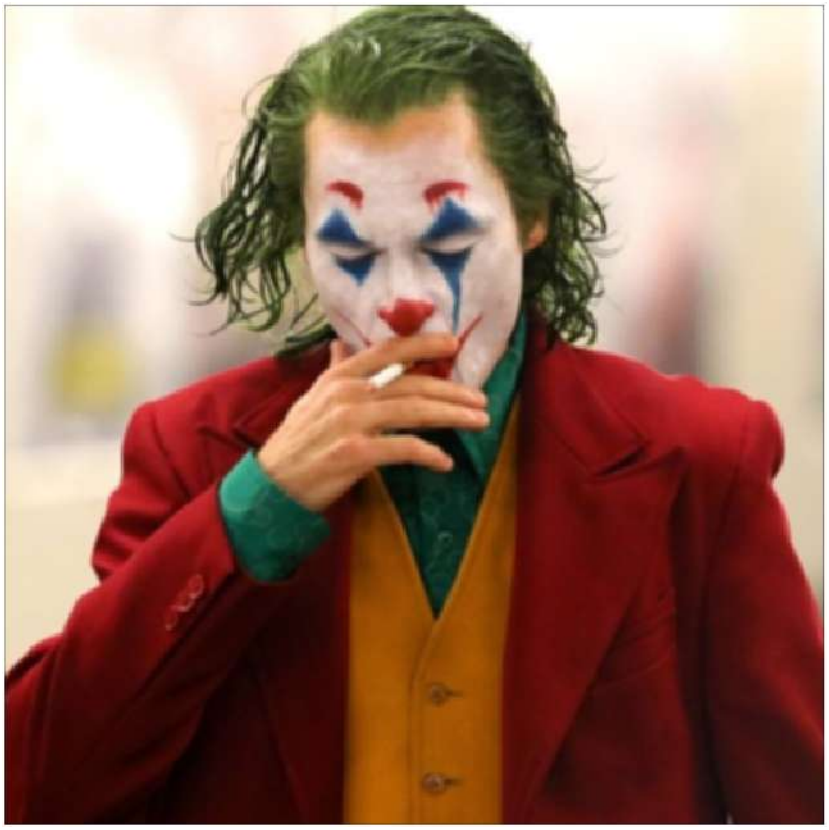
```
function img_final = apply_effect(img, kernel)
    %separate color channels
    imgR = img(:,:,1);
    imgG = img(:,:,2);
    imgB = img(:,:,3);

    %take kernel size
    K = size(kernel);

    %add padding to img
    imgR_p = add_padding(imgR,K);
    imgG_p = add_padding(imgG,K);
    imgB_p = add_padding(imgB,K);

    %apply convolution
    img_finalR = my_conv2(double(kernel),double(imgR_p));
    img_finalG = my_conv2(double(kernel),double(imgG_p));
    img_finalB = my_conv2(double(kernel),double(imgB_p));

    %merge color channels
    img_final = cat(3, img_finalR, img_finalG, img_finalB);
end

function result = my_conv2(kernel, img)
    %take size of the matrixes
    I = size(img);
    K = size(kernel);
    new_img = zeros(512,512);
    %iterate over image
    for i = 1:(I(1)-K(1)+1)
        for j = 1:(I(2)-K(2)+1)
            %calculate the pixel value for the current location
            pixel_value = 0;
            for a = 1:K(1)
                for b = 1:K(2)
                    pixel_value = pixel_value + kernel(a,b)*img(i+a-1,j+b-1);
                end
            end
            %set the pixel value
            new_img(i,j) = pixel_value;
        end
    end
    result = new_img;
end

function img_with_padding = add_padding(img, K)
    % take size of img
    img_size = size(img);

    %calculate padding gap
    gap = (K(1)-1)/2;

    %add zero padding
    img_with_padding = [zeros(gap,img_size(1)+(gap*2)); [zeros(img_size(2),gap), img,
        zeros(img_size(2),gap)]; zeros(gap,img_size(1)+(gap*2))];
end
```

Original

I have read the image from the file.

```
img = imread("img/jokerimage.png");
figure('Name','Original');
imshow(uint8(img));
```

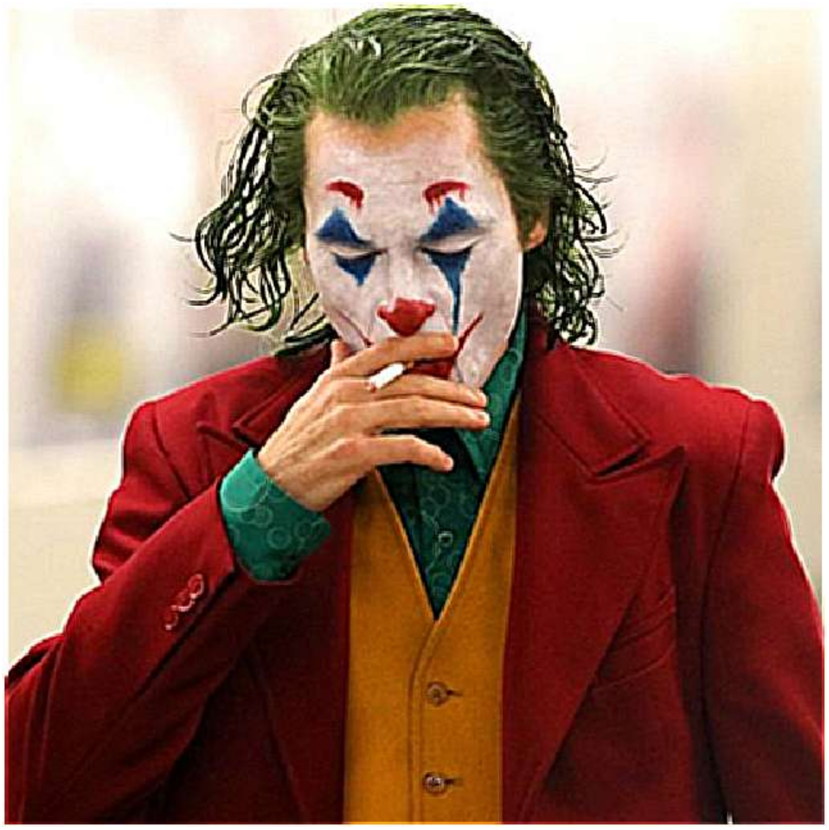
Blurred



Applied the blur filter and displayed the result. It takes the average of the adjacent pixels so that creates a blurry result.

```
%blur kernel
blur = [
    1 1 1;
    1 1 1;
    1 1 1]/9;

%apply blur filter
imgBlur = apply_effect(img, blur);
imgBlur = uint8(imgBlur);

%display result
figure('Name','Blurred');
imshow(imgBlur);
```

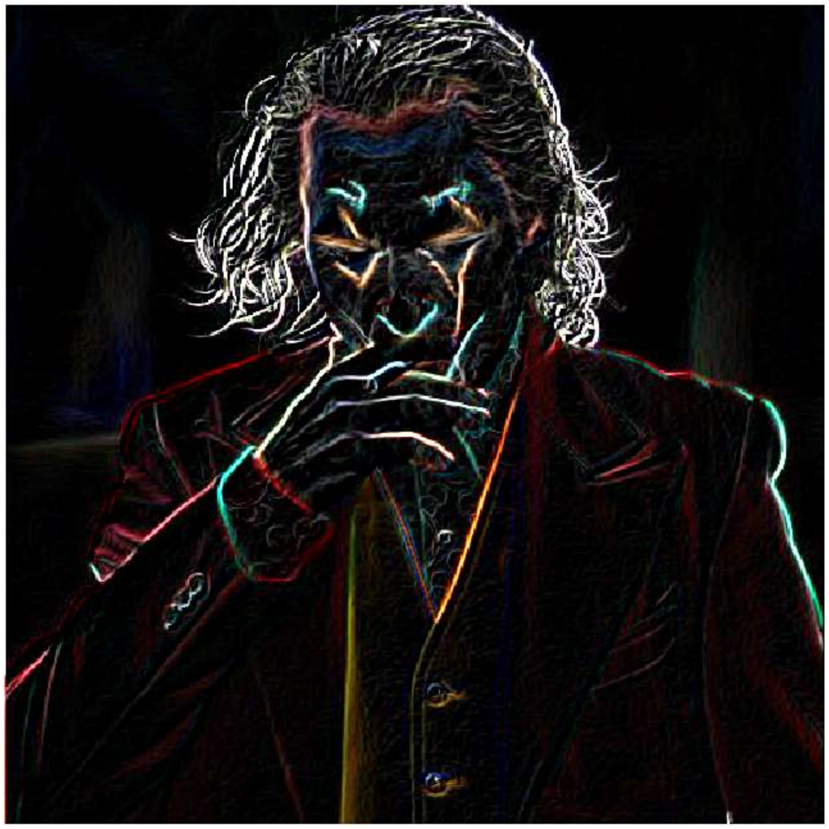
Sharpened



Applied the sharpness filter and displayed the result. It empowers the current value and subtracts the adjacent ones and that creates a sharp result.

```
%sharpness kernel
sharp = [
    -1 -1 -1;
    -1 9 -1;
    -1 -1 -1];

%apply sharpness filter
imgSharp = apply_effect(imgBlur, sharp);
imgSharp = uint8(imgSharp);

%display result
figure('Name','Sharpened');
imshow(imgSharp);
```

Edge Detection

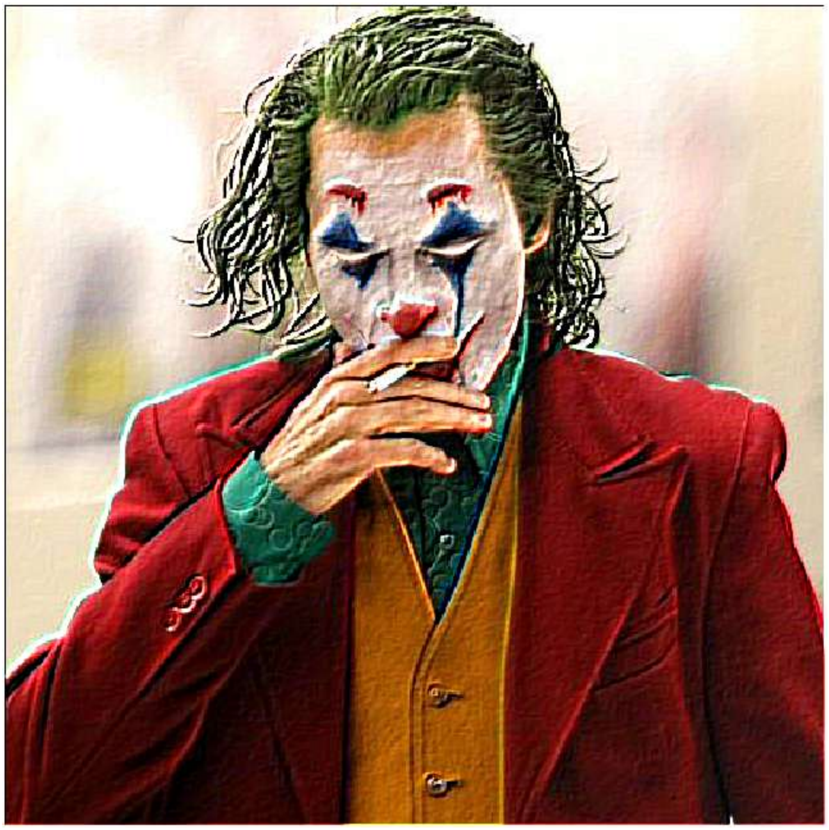


Applied the bottom sobel filter and the right sobel filter. Then merged their results by using $a^2 = b^2 + c^2$ and displayed the result. It removes one side of the pixel from the other so if there is a significant color change that means we detected an edge. First applies vertically then horizontally. In order to reach final result I have used Pythagoras's theorem.

```
%sobel kernels kernel
top_sobel = [1 2 1;0 0 0;-1 -2 -1];
bottom_sobel = [-1 -2 -1;0 0 0;1 2 1];
left_sobel = [1 0 -1;2 0 -2;1 0 -1];
right_sobel = [-1 0 1;-2 0 2;-1 0 1];

%apply bottom sobel filter
imgBottomSobel = apply_effect(img, bottom_sobel);
imgBottomSobel(imgBottomSobel<0) = 0;

%apply rignt sobel filter
imgRightSobel = apply_effect(img, right_sobel);
imgRightSobel(imgRightSobel<0) = 0;

%calculate final result
imgEdge = (imgBottomSobel.^2 + imgRightSobel.^2).^(1/2);
imgEdge = uint8(imgEdge);
```

```
%display result
figure('Name','Edge Detection');
imshow(imgEdge);
```

Embossed

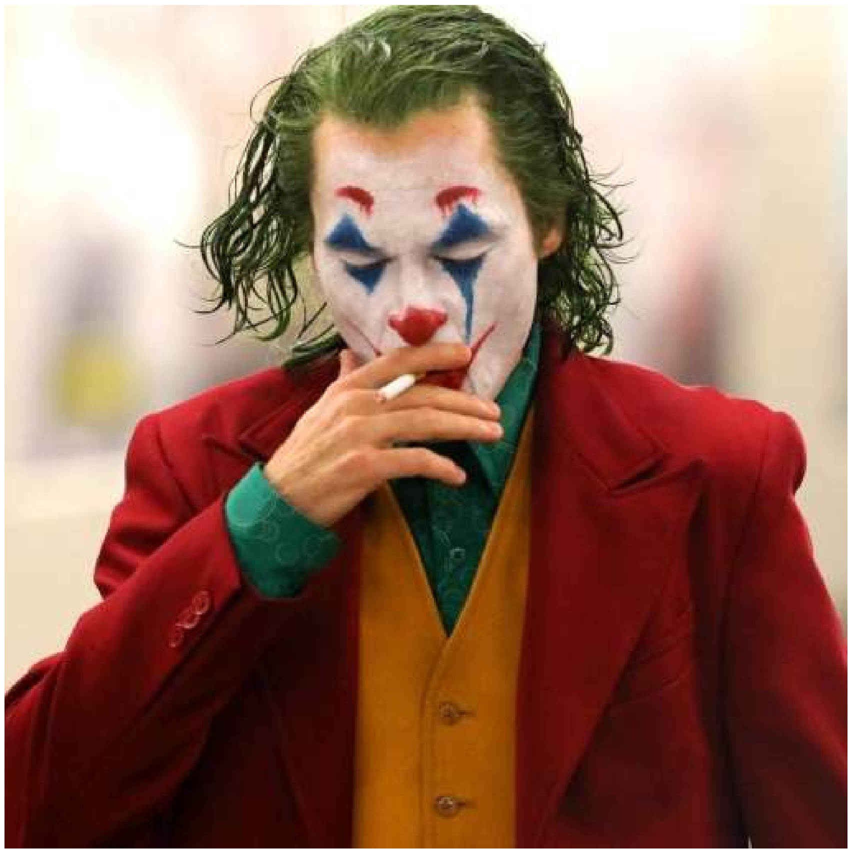


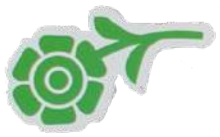
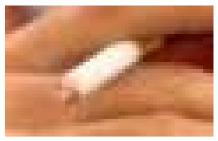
Applied the embossing filter and displayed the result. It enhanced the some left-top edges.

```
%embossing kernel
emboss = [
    2 1 0;
    1 1 -1;
    0 -1 -2];

%apply embossing filter
imgEmboss = apply_effect(img, emboss);
imgEmboss = uint8(imgEmboss);

%display result
figure('Name','Embossed');
imshow(imgEmboss);
```

Solution 2 (Bonus)

Read Image

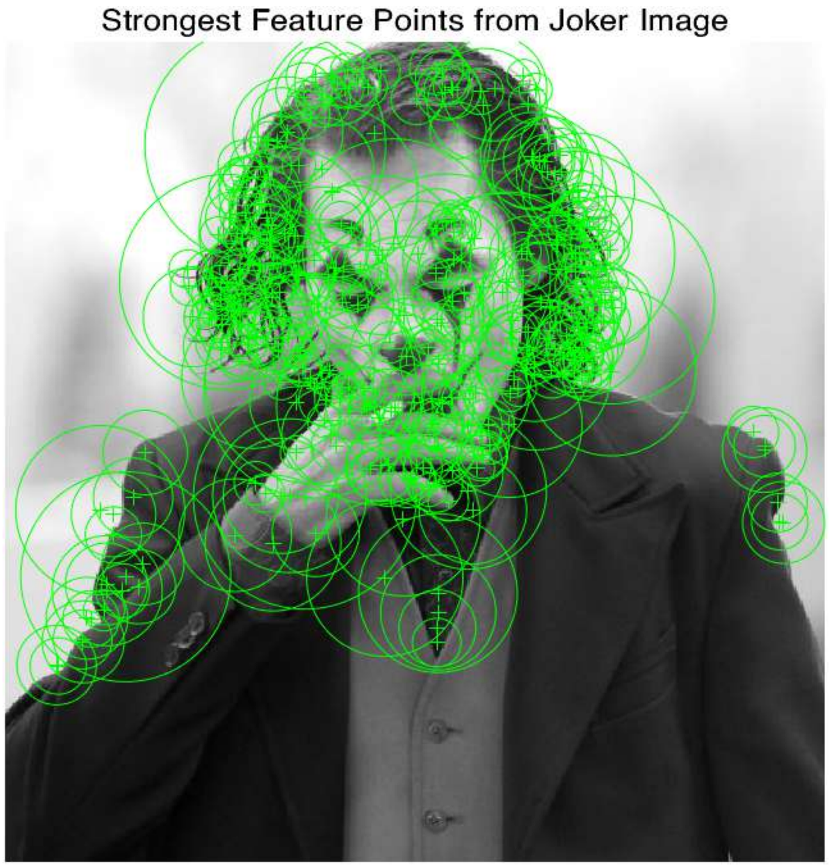


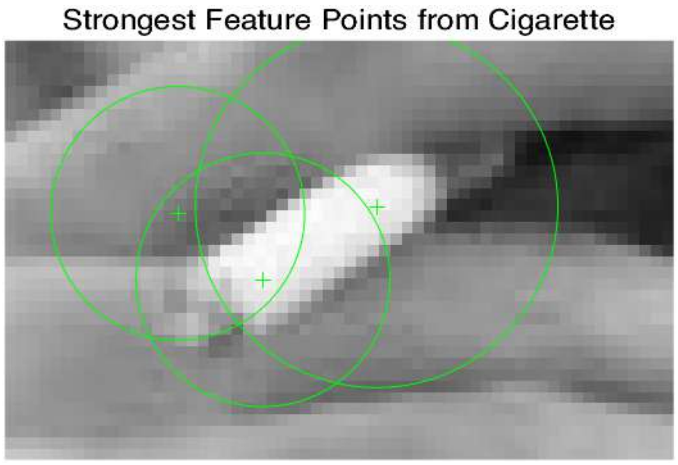



I have read the cigarette, joker and flower images.

```
%% Read Image
img = imread("img/jokerimage.png");
cigarette = imread("img/cigarette.png");
flower = imread("img/flower.png");
```

Detect Feature Points





Detected feature points of the joker image and cigratte by using detectSURFFeatures function. Then results are displayed.

```
%% Detect Feature Points
img = rgb2gray(img);
```

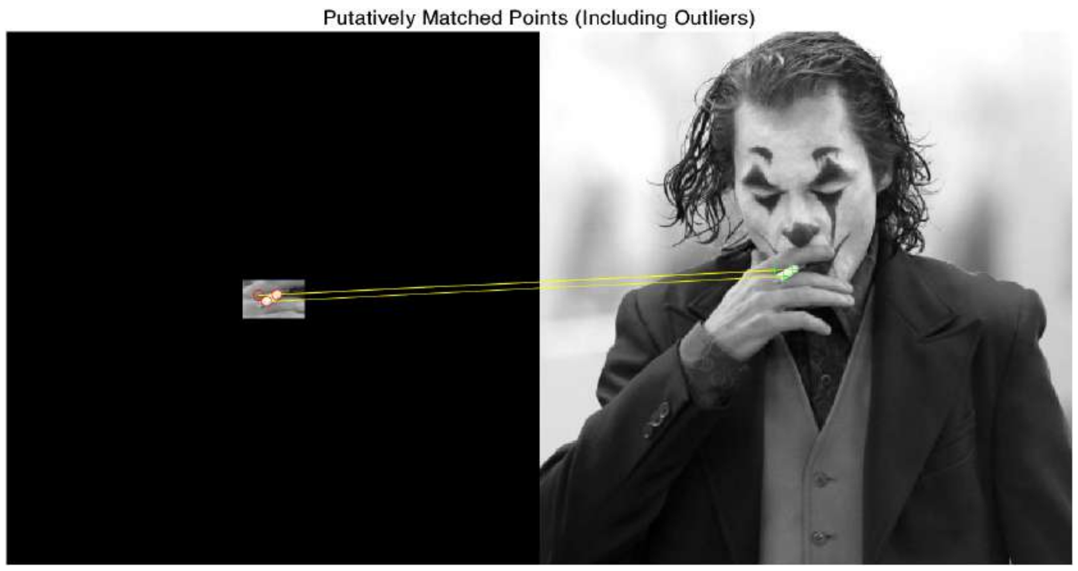
```
cigarette = rgb2gray(cigarette);

cigarettePoints = detectSURFFeatures(cigarette);
imgPoints = detectSURFFeatures(img);

%display results
figure;
imshow(cigarette);
title('Strongest Feature Points from Cigarette');
hold on;
plot(cigarettePoints);

figure;
imshow(img);
title('Strongest Feature Points from Joker Image');
hold on;
plot(imgPoints);
```

Extract Feature Descriptors

Feature descriptors are exracted from the feature points that are calculated at "Detect Feature Points" part.

```
%% Extract Feature Descriptors
[cigaretteFeatures, cigarettePoints] = extractFeatures(cigarette, cigarettePoints);
[imgFeatures, imgPoints] = extractFeatures(img, imgPoints);
```

Find Putative Point Matches

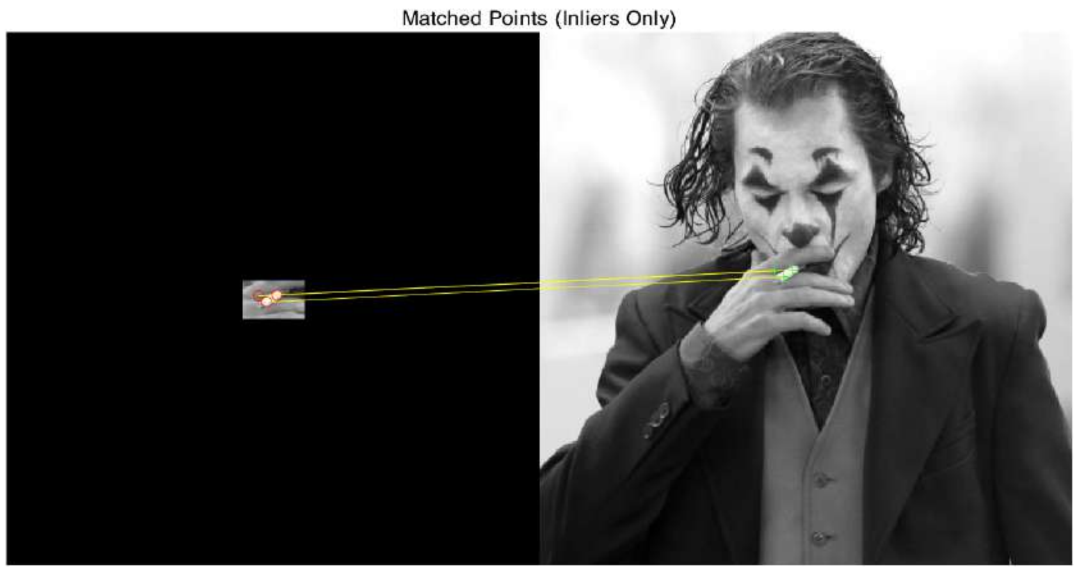


Matched points are founded including outliers. But in this case there are no outlier matches.

```
%% Find Putative Point Matches
```

```
cigarettePairs = matchFeatures(cigaretteFeatures, imgFeatures);

matchedCigarettePoints = cigarettePoints(cigarettePairs(:, 1), :);
matchedImgPoints = imgPoints(cigarettePairs(:, 2), :);

%display results
figure;
showMatchedFeatures(cigarette, img, matchedCigarettePoints, ...
    matchedImgPoints, 'montage');
title('Putatively Matched Points (Including Outliers)');
```

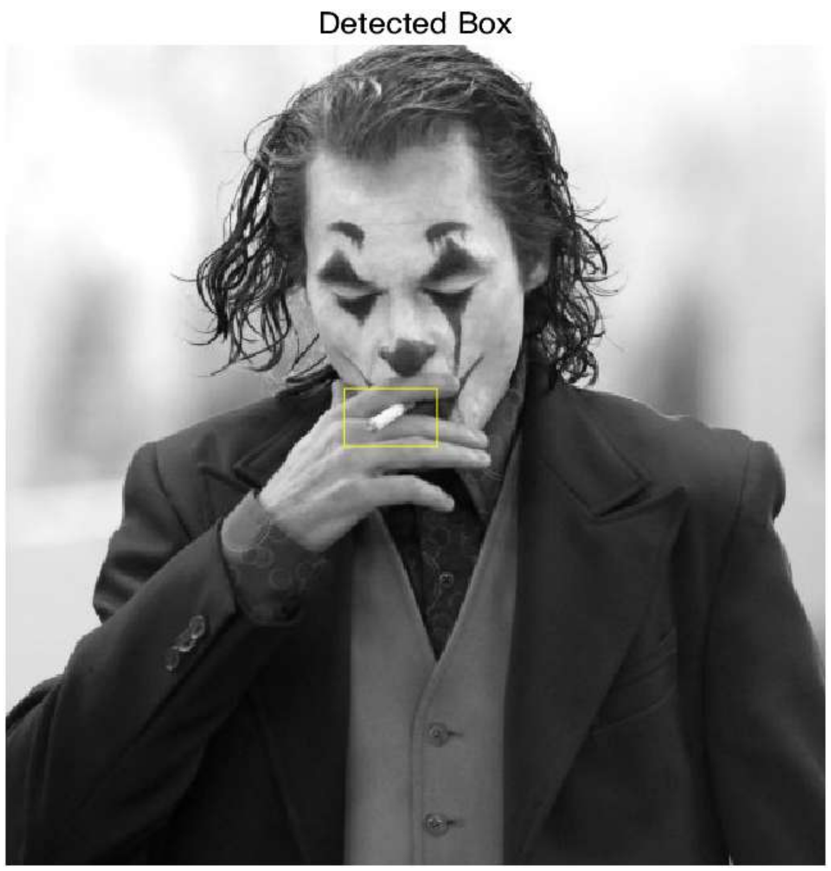
Locate the Object in the Scene Using Putative Matches



Since there are no outlier feature point result of "Find Putative Point Matches" section and this section is identical. So we found the object that we are looking for.

```
%% Locate the Object in the Scene Using Putative Matches
[tform, inlierCigarettePoints, inlierImgPoints] = ...
    estimateGeometricTransform(matchedCigarettePoints, matchedImgPoints, 'affine');

%display results
figure;
showMatchedFeatures(cigarette, img, inlierCigarettePoints, ...
    inlierImgPoints, 'montage');
title('Matched Points (Inliers Only)');
```

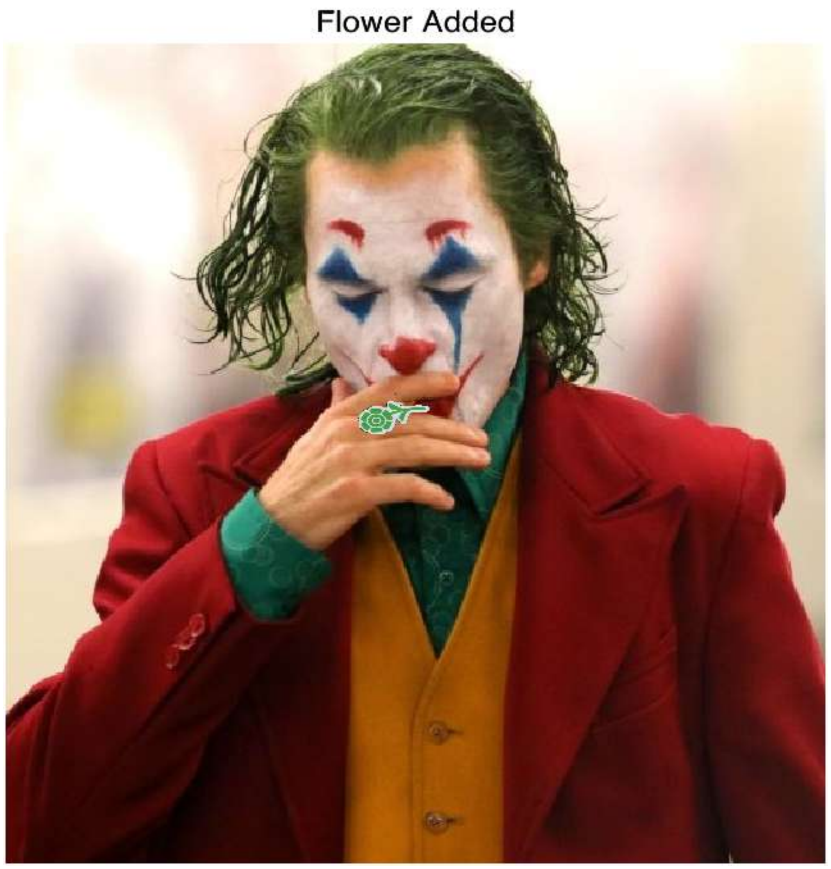
Put the Cigarette in a Rectangle



In order to highlight the detection I put the detected area in a box.

```
%% Put the Cigarette in a Rectangle
cigarettePolygon = [1, 1;...                          % top-left
        size(cigarette, 2), 1;...               % top-right
        size(cigarette, 2), size(cigarette, 1);... % bottom-right
        1, size(cigarette, 1);...               % bottom-left
        1, 1];                  % top-left again to close the polygon

newCigarettePolygon = transformPointsForward(tform, cigarettePolygon);

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

Place The Flower Censor



Finally I placed the flower on top of the detected rectangle.

```
%% Place The Flower Censor
img = imread("img/jokerimage.png");

%display results
figure;
imshow(img);
hold on;
mask = double( any(flower, 3) );
image([min(newCigarettePolygon(:,1)) max(newCigarettePolygon(:,1))],
    [min(newCigarettePolygon(:,2)) max(newCigarettePolygon(:,2))],flower,'AlphaData',
    mask);
title('Flower Added');
```