86735: Computer Vision
Color based segmentation and normalized cross correlation

Due date: Wed 14.15, 21.10.15

Prof. F. Odone, Prof. F. Solari, M. Chessa, N. Noceti

Ernest Skrzypczyk, BSc

Color based segmentation and normalized cross correlation

The goal of the practical laboratory session was to perform segmentation on the test images using components of a color space and template matching with cross correlation.

Segmentation in image processing is a process of dividing the image into parts, called segments, which have similar properties, like belonging to the same object or curve, line. The goal of segmentation is to create a higher level of abstract information, that are easier for further analysis or processing.

When dealing with a sequence of images one of the more useful processing capabilities is to determine the similarities between them, for example like in the laboratory session, determine the position of a moving object in a dynamic scenery. One of the possible techniques is to use template matching, in particular normalized cross correlation is very useful for this task.

In the proposed script for the Matlab environment segmentation is done by using the hue component of the input image and normalized cross correlation for template matching the object. Additionally, since the results of segmentation using the hue component were not satisfactory, an algorithm using hue, saturation and further processing has been added. For blurring during the image processing the Gaussian filter was used.

Decomposing images into components

First the images were loaded and converted to grayscale, after which they were decomposed into red, green and blue color space channels. Next decomposition was into the components of the HSV color space, so hue, saturation and value respectively.

Figure 1: Grayscale image number 1

The figures (1) to (3) show conversion and decomposition for the first image only. The rest of the images can be very easily generated by the enclosed script written for the Matlab computing environment.

The conversion into grayscale is done in Matlab using a weighted sum of the red, green, and blue components according to formula (1):

$$Output(x,y) = Red(x,y) \cdot 0,2989 + Green(x,y) \cdot 0,5870 + Blue(x,y) \cdot 0,1140 \quad (1)$$

The decomposition into each of the RGB channels consists of copying the data from each channel and displaying them in grayscale. The decomposition of the RGB input images into HSV color space is performed by the *rgb2hsv* function, after which the channels of hue, saturation and value are presented in grayscale as with RGB channels.

Figure (2) depicts all channels of the RGB input image as grayscale and in filled form. Since the car in the center is red, its red component is lighter, ergo has a higher value, than the green and blue components.

Figure (3) clearly shows that the hue component is very noisy, in other words the changes are not smooth. Also the range of values is narrow, which indicates a more uniform histogram. On the other hand the saturation channel seems to have a bimodal histogram and the object of interest, namely the red car in

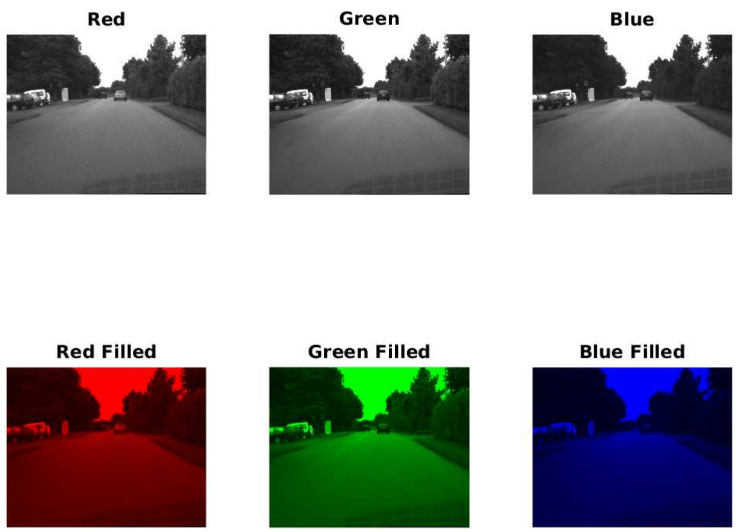

Figure 2: Red, green, blue decomposition of image number 1

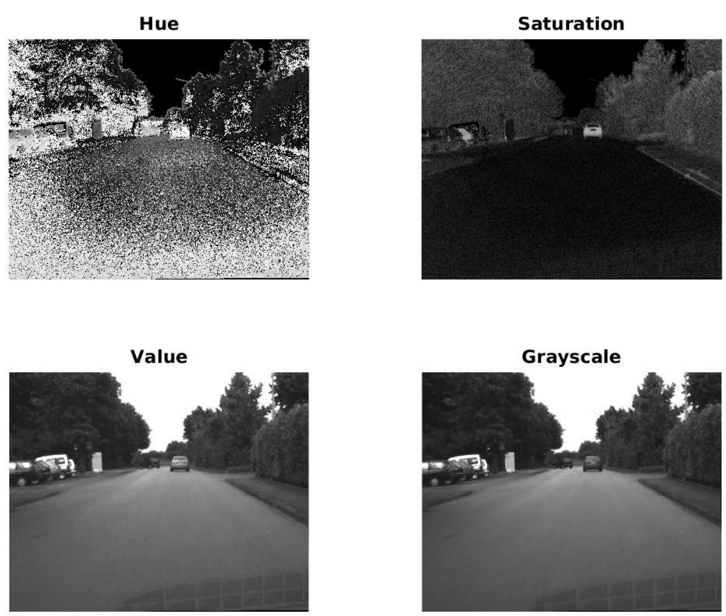

Figure 3: Hue, saturation and value decomposition of image number 1

the center of the image, is distinguishable from its surroundings. This indicates that the saturation channel might be better suited for the task of segmenting the red car.

Hue based segmentation

Next step was to use the hue channel of image 1 and to segment it, in this case with the goal of isolating the red car in the center from the rest of the image. Since the hue channel is not contrast-rich or noise-free, it can be assumed before processing, that the result will not be unambiguous. Depending on the application this might be a critical observation. Since the results shown in figures ($4 \div 7$) are not satisfactory, a great effort of parametrizing the thresholds and regions of interest (ROIs) has been undertaken. This resulted in modifying the thresholding formula from the suggested equation (2) into (3):

$$Threshold = MeanValue \pm 3 \cdot StandardDeviation \tag{2}$$

$$Threshold = p_1 \cdot MeanValue \pm p_2 \cdot StandardDeviation \tag{3}$$

The lower value of those formulas corresponds to threshold minimum, and the greater to threshold maximum. Binarizing the hue channel with those thresholds was supposed to segment the ROI. The lines $48 \div 50$ and 292 in the script (1) correspond to the parametrizing loop. Different regions of interest have been defined in line 46 and set into code in lines $110 \div 201$. The ROI is then used for calculating the mean and standard deviation values, which in turn determine the thresholds.

Figures ($4 \div 7$) clearly show how this technique has problems with noisy channels. The bounding box was supposed to be placed around the car or at least a part of the car corresponding to the region of interest, not around the whole image. The results are very similar, despite different shapes and coordinates of the ROIs. The cross was supposed to be placed on the object, as the centeroid of the recognized area. These poor results were the reason for implementing a custom algorithm of segmentation based on component channels.

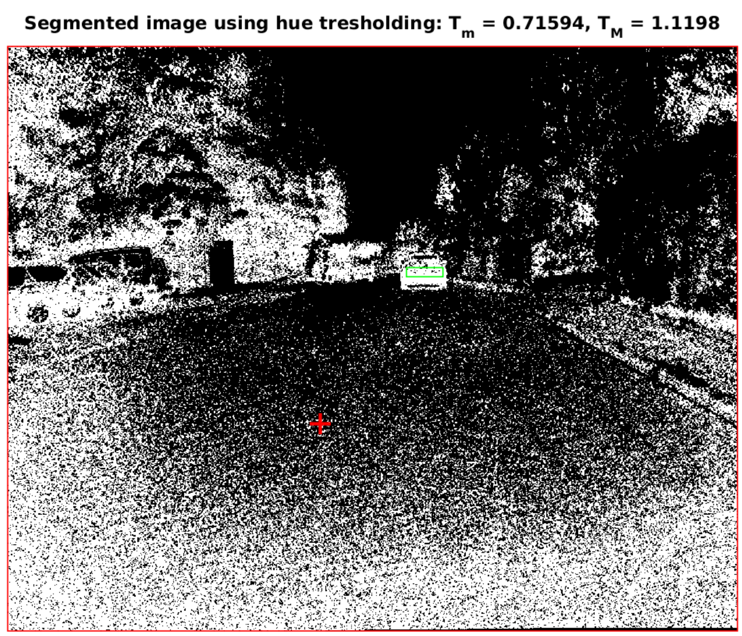

Figure 4: Hue segmentation using object part 1

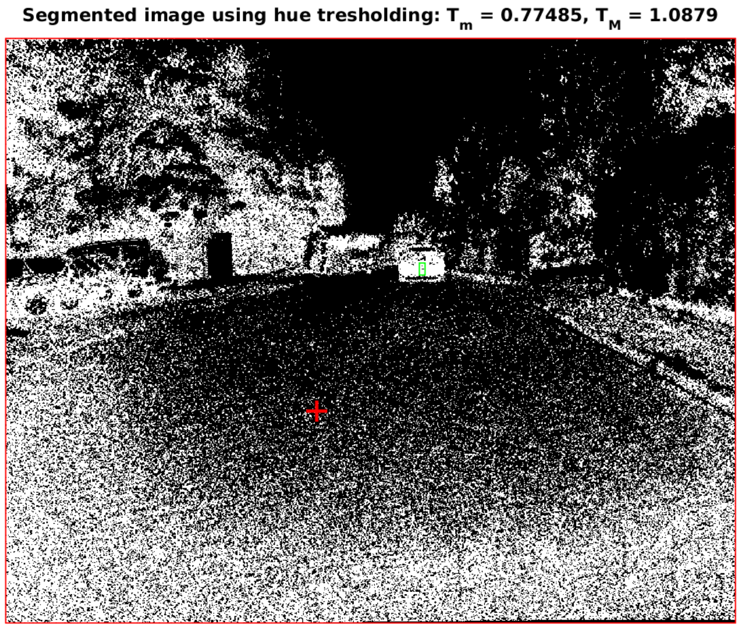

Figure 5: Hue segmentation using object part 2

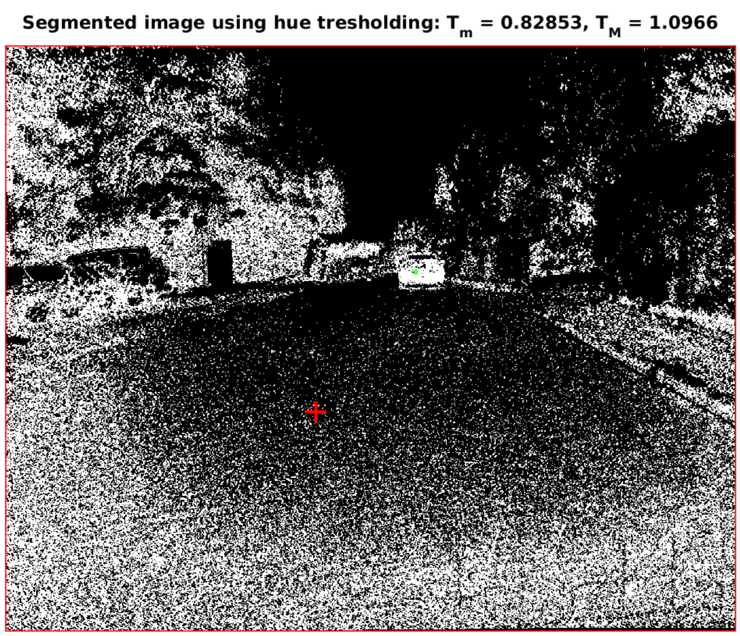

Figure 6: Hue segmentation using object part 4

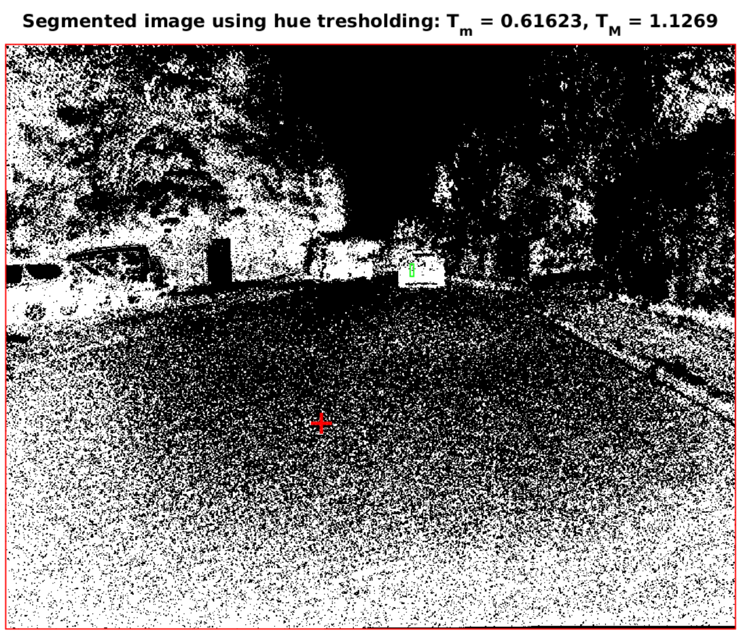

Figure 7: Hue segmentation using object part 6

Segmentation by hue and saturation

Lines $151 \div 201$ of the script are the implementation of the custom code. As stated in the beginning of the script (lines $38 \div 41$), further parametrization can be performed changing the weights $w11 \div w33$. Instead of changing the thresholds, the source image is the weighted mean sum of the hue and saturation components with addition to a processed hue component. The processing included blurring using the Gaussian filter with a radius of 18 pixels and the standard deviation of also 18 pixels as seen in the line 176 of the script. In the code the image was binarized, however the images were not in order to show the influence and weight of certain components. The saturation channel is dominant as expected. The processed hue component helps fill the saturated region of interest.

The results presented in figures $(8 \div 11)$ are quite satisfactory. Only one region of interest, depicted on figure 10 did not result in anticipated segmentation. The ROI was very small and resulted in placement of the centroid on the second red car in the image. Other than that, the algorithm was successful with different shapes and positions of the ROI.

Figure 8: Hue and saturation segmentation using object part 1

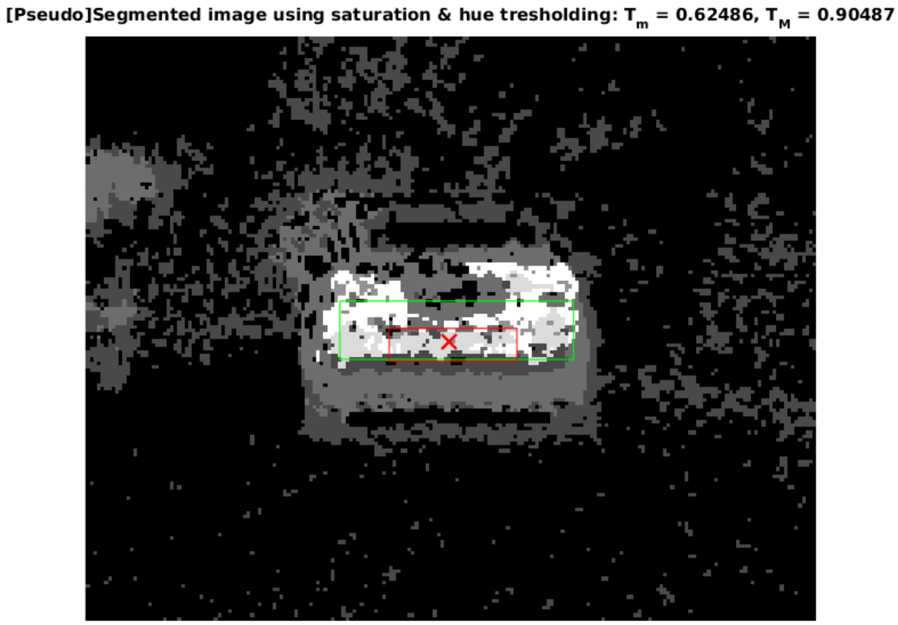

Figure 9: Hue and saturation segmentation using object part 1 - Zoomed in to the ROI

Figure 10: Hue and saturation segmentation using object part 4

Figure 11: Hue and saturation segmentation using object part 5

Template matching using normalized cross correlation

The last part of the script (1) is performing cross correlation between an object in the image, in this case the red car in the center, and the whole image. Because of the way the images are processed the dimensions of the resulting images are increased by the size of the object using for cross correlation. This should be taken into consideration. One possibility is to select the object of interest and then perform matching across all images. This approach has limits, since the object might change so much that a cross correlation will not be successful any more. Therefore updating the selected object should occur with a frequency guaranteeing enough similarities between the object and images. One possible scenario is a curve as seen in this image sequence. If the there is enough distance between the camera car and the target car and the curve has a small enough radius, then the car would be distorted as an object for template matching, thus possibly reducing the success rate significantly. Another approach would be to select the object in every image and then perform template matching, but this serves little purpose.

The figures ($12 \div 14$) show the normalized cross correlation (NCC) for input images using the object from the first image. Figures ($15 \div 17$) show NCC for selected images using the object from the processed image respectively. Next figures ($18 \div 23$) show the comparison for both procedures between the normalized cross correlation and conventional cross correlations without additional operations and with subtraction of the mean and standard deviation values. To display those, additional normalizing was performed.

Clearly the NCC provides the most accurate and readable information about the cross correlation out of those techniques presented. It also provides further benefits when it comes to variations in lighting in the image scenery. The last figure (24) shows the NCC performed on the first image in a three dimensional chart. The red peak corresponds to perfect match.

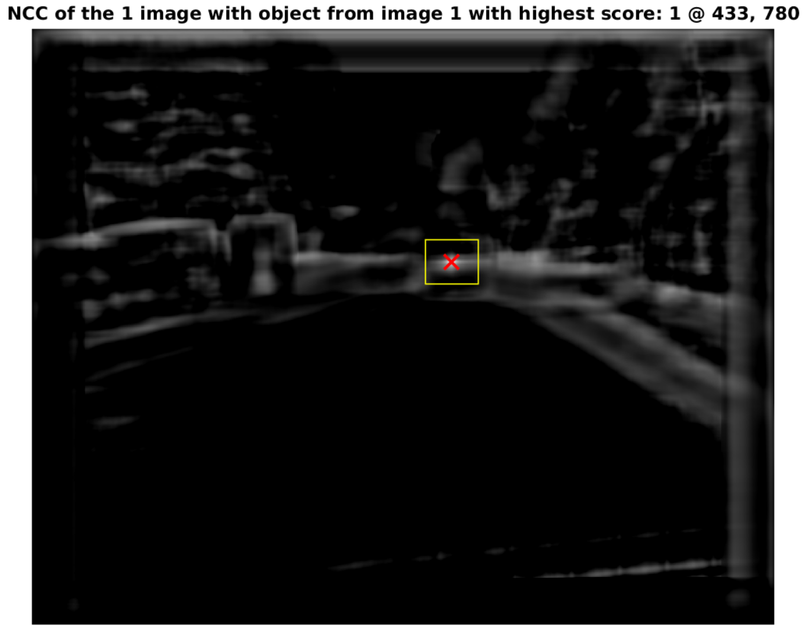

Figure 12: Normalized cross correlation for image 1

Figure 13: Normalized cross correlation for image 2

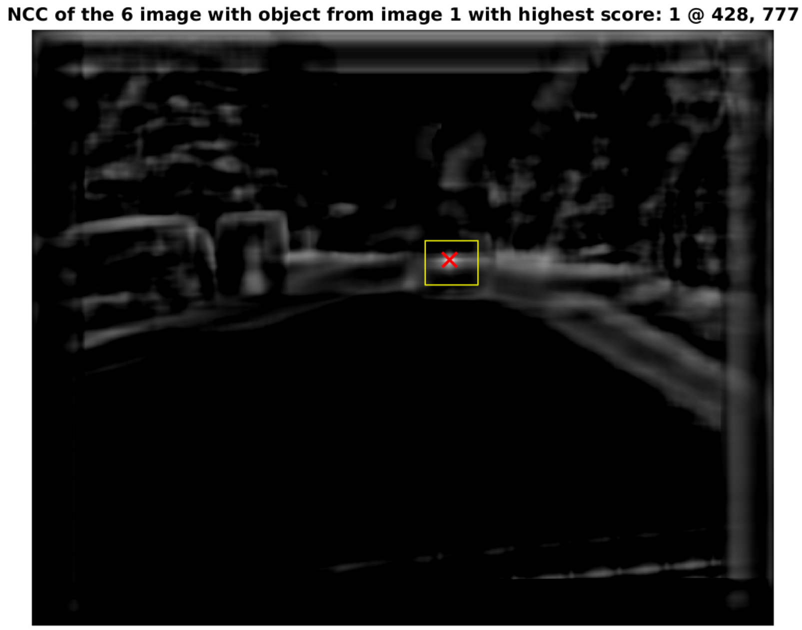

Figure 14: Normalized cross correlation for image 6

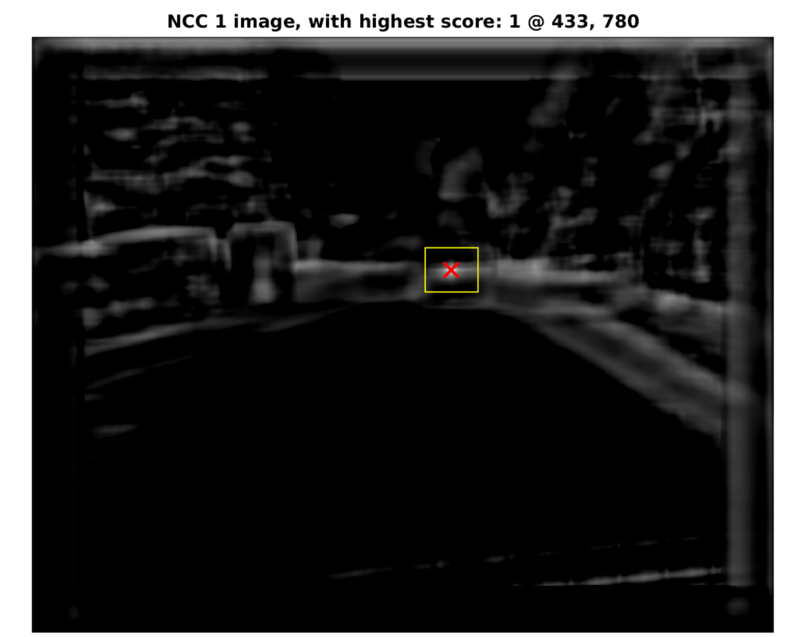

Figure 15: Normalized cross correlation for image 1

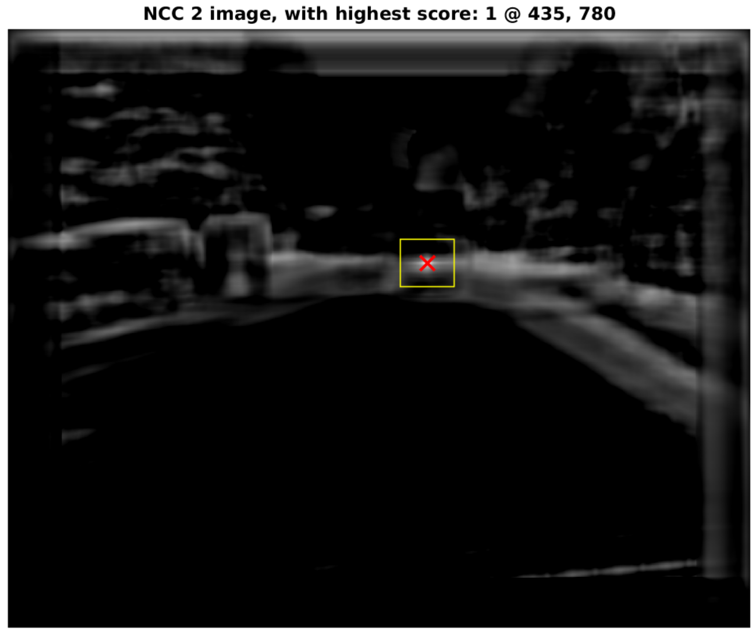

Figure 16: Normalized cross correlation for image 2

Figure 17: Normalized cross correlation for image 6

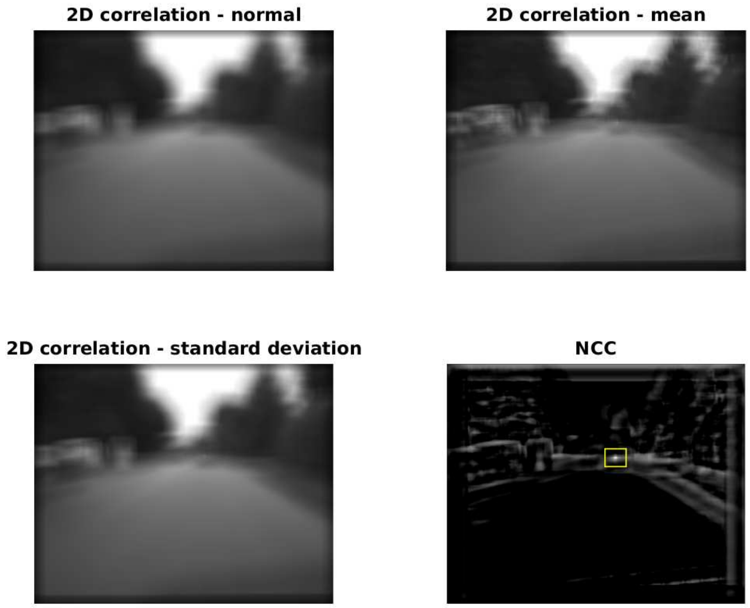

Figure 18: Comparison of cross correlations for image 1 with object 1 as template

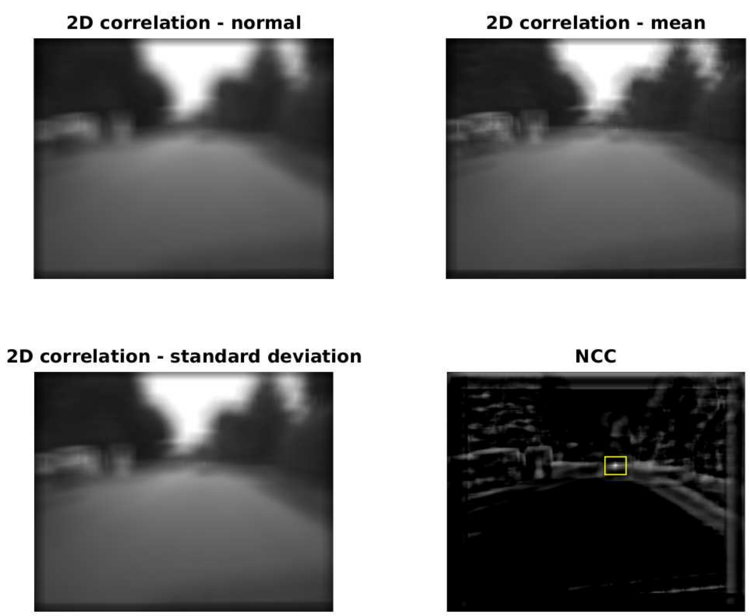

Figure 19: Comparison of cross correlations for image 2 with object 1 as template

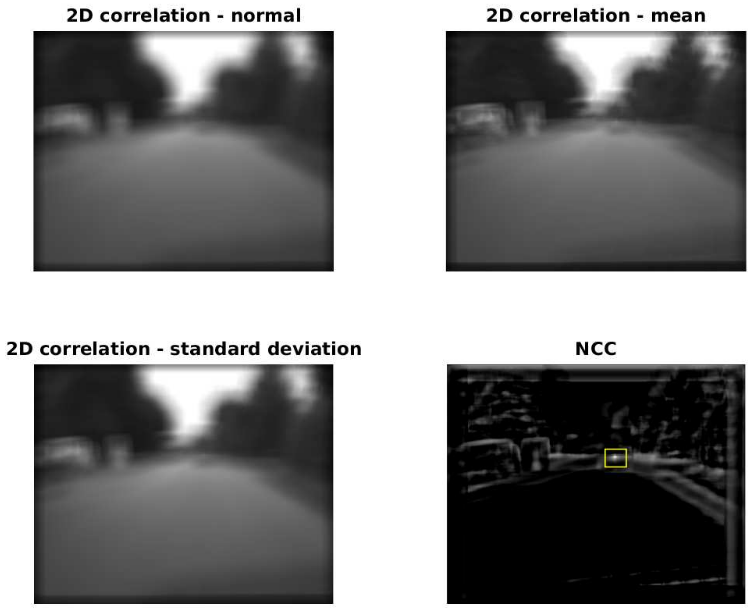

Figure 20: Comparison of cross correlations for image 6 with object 1 as template

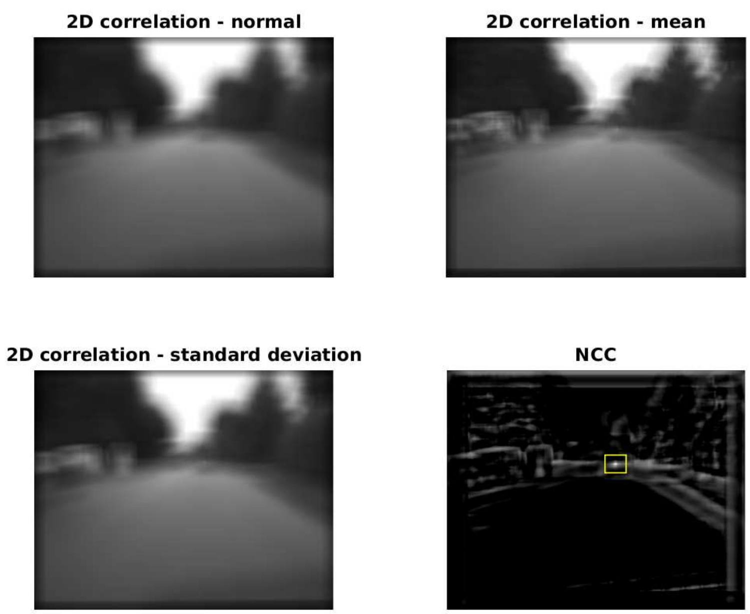

Figure 21: Comparison of cross correlations for image 1

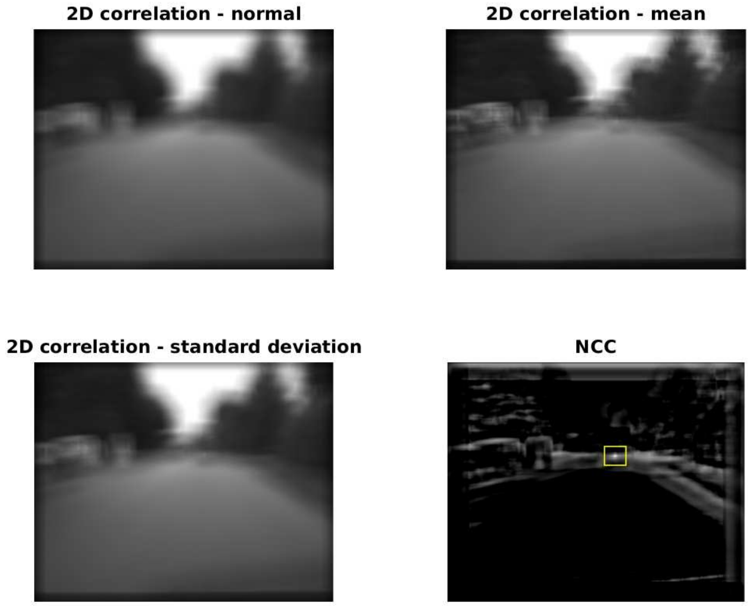

Figure 22: Comparison of cross correlations for image 2

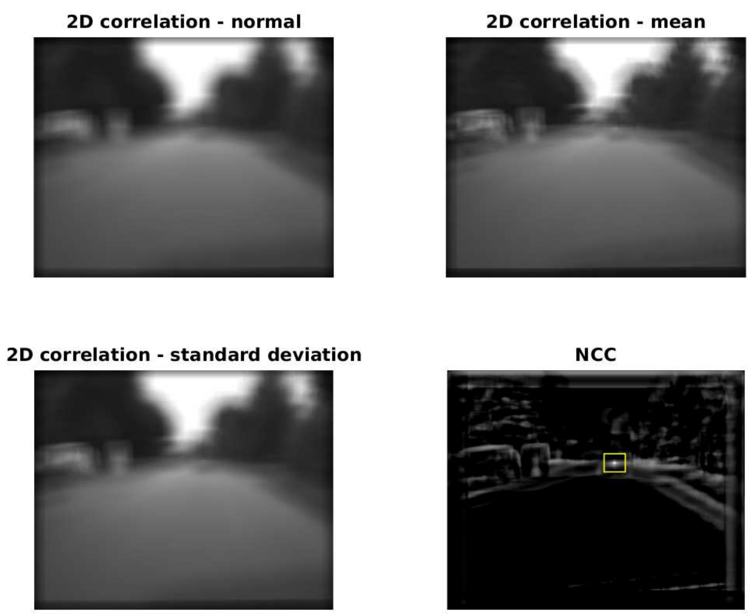

Figure 23: Comparison of cross correlations for image 6

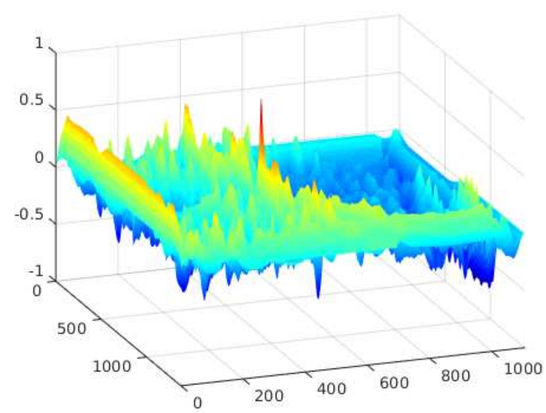

Figure 24: 3 dimensional representation of normalized cross correlations for image 1

Script 1: Color based segmentation and template matching using cross correlation script written for Matlab.

```
%% Ernest Skrzypczyk - 4268738
%% 21.10.2015
%% Computer Vision - L03 - Colour based segmentation and normalized cross correlation
%% Matlab 8.4.0.150421 (R2014b)

%function LAB03

close all; clear all; clc;

%Script parameters
imagesn = 6; %Number of images
imageshow = 1; %Option for showing images
imagerender = 1; %Option for rendering images
imagesave = 0; %Option for saving images
manualroi = 0; %Option for manual input of ROI (region of interest)
singleroincc = 0; %Option for single object used for NCC across all images
singleroinccindex = 1; %Index for single object used for NCC
lowmem = 1; %Option for saving memory through less open figures at a time
imageleave = 1; %Option for waiting after an image has been processed
imageextension = 'pdf'; %Image file format
%For quickly generating images use:
imageshow = 0; imagerender = 1; imagesave = 1; imageextension = 'png'; imageleave = 0;
%For showing images use:
%imageshow = 1; imagerender = 1; imagesave = 0; imageleave = 1;

%Setting basic script options
screensize = get(0,'ScreenSize'); %Screen size
set(groot, 'defaultFigurePosition', [screensize(3)/6, screensize(4)/6, screensize(3)/1.5, screensize(4)/1.5]);
set(groot, 'defaultFigurePaperUnits', 'points', 'defaultFigurePaperSize', [1366 1024])
if imagerender == 1; set(groot, 'defaultFigureVisible', 'on', ' defaultFigureRenderer', 'opengl'); end %painters might be very slow
if imagesave == 1; set(groot, 'defaultFigureRenderer', 'opengl');%painters');
if imageshow == 0; set(groot, 'defaultFigureVisible', 'off'); end; end

%Segmentation weights for the thresholding range (p1 * mean +/- p2 * stddev)
%p1 = 0.95; p2 = 0.5;
p1 = 1; p2 = 0.8;

%Additional weights
w11 = 1; w12 = 1; %Mean value weights
w21 = 1; w22 = 1; %Standard deviation weights
w31 = 1; w32 = 2; w33 = 0.5; %Component images weights

%Coordinate boxes for the whole object (car)
ObjectCoordinatesArray = {[682, 351, 98, 82, 780, 433], [680, 347, 100, 88, 780, 435], [682, 347, 96, 84, 778, 431], [678, 347, 102, 84, 780, 431], [674, 345, 112, 86, 786, 431], [678, 345, 98, 84, 776, 429]};
```

```
%Coordinate boxes for parts of the object
ObjectCoordinatesArrayParts = {[700, 388, 64, 16], [726, 394, 10, 21], [707, 394, 44, 19], [718, 393, 4, 6], [710, 379, 10, 30], [710, 385, 6, 22]};

%% Parametrizing loop
% for p1 = 0.2:0.2:2 %Parametrizing thresholding
%     for p2 = 0.2:0.2:1.0
p1, p2 %Print current parameters
dir=['L03_p1_', num2str(p1), '_p2_', num2str(p2)]; mkdir(dir);

%% Main loop
%Loading images
for i = 1:imagesn
    %Concating filenames / It is possible to use dir('*.png') instead
    filename = ['ur_c_s_03a_01_L_0', num2str(375 + i), '.png'];
    Images{i} = imread(filename, 'png');
    %Images in grayscale
    ImagesGrayscale{i} = rgb2gray(Images{i});
    if imagerender == 1
        figure;
        imshow(ImagesGrayscale{i}); colormap('gray');
        title('Grayscale image');
    end
    if imagesave == 1
        saveas(gcf, [dir, '/', 'L03_01_GS_', num2str(i)], imageextension);
    end

    %Images in seperate RGB channels
    ImagesR{i} = Images{i}(:, :, 1);
    ImagesG{i} = Images{i}(:, :, 2);
    ImagesB{i} = Images{i}(:, :, 3);
    ImagesZ{i} = zeros(size(Images{i}, 1), size(Images{i}, 2));

    %Displaying R-G-B images as grayscale and with colour masks
    if imagerender == 1
        figure;
        title('Decomposition of the image into red, green, blue channels');
        subplot(2, 3, 1); imshow(ImagesR{i}); title('Red');
        subplot(2, 3, 2); imshow(ImagesG{i}); title('Green');
        subplot(2, 3, 3); imshow(ImagesB{i}); title('Blue');
        subplot(2, 3, 4); imshow(cat(3, ImagesR{i}, ImagesZ{i}, ImagesZ{i})); title('Red Filled');
        subplot(2, 3, 5); imshow(cat(3, ImagesZ{i}, ImagesG{i}, ImagesZ{i})); title('Green Filled');
        subplot(2, 3, 6); imshow(cat(3, ImagesZ{i}, ImagesZ{i}, ImagesB{i})); title('Blue Filled');
    end
    if imagesave == 1
        saveas(gcf, [dir, '/', 'L03_02_RGB_', num2str(i)], imageextension);
    end
    %Images in HSV colorspace
    ImagesHSV{i} = rgb2hsv(Images{i}); colormap('gray');
```

```
    ImagesH{i} = ImagesHSV{i}(:,:,1);
    ImagesS{i} = ImagesHSV{i}(:,:,2);
    ImagesV{i} = ImagesHSV{i}(:,:,3);
    %Displaying H-S-V images as grayscale
    if imagerender == 1
        figure;
        title('Decomposition of the image into hue, saturation and value channels');
        subplot(2, 2, 1); imshow(ImagesH{i}); title('Hue'); colormap('gray');
        subplot(2, 2, 2); imshow(ImagesS{i}); title('Saturation'); colormap('gray');
        subplot(2, 2, 3); imshow(ImagesV{i}); title('Value'); colormap('gray');
        subplot(2, 2, 4); imshow(ImagesGrayscale{i}); title('Grayscale');
    end
    if imagesave == 1
        saveas(gcf, [dir, '/', 'L03_03_HSV_', num2str(i)], imageextension);
    end
if lowmem == 1; if imageleave == 1; pause; end; close all; end

%% Object parts loop
for l = 1:length(ObjectCoordinatesArrayParts)
    %% Segmentation by hue
if i == 1 %Run only for the first image
    figure;
    imshow(Images{i});
    if manualroi == 1
        title('Hue & Hue and Saturation - Select the region of interest (ROI)');
        %title('Hue - Select the region of interest (ROI)');
        ObjectCoordinates = int16(getrect); % X, Y, Width, Height;
    else
        title('Hue segmentation - Selected region of interest');
        %ObjectCoordinates = [700, 388, 64, 16]; %Default object coordinates
        ObjectCoordinates = ObjectCoordinatesArrayParts{l}; %Object coordinates taken from the array
    end
    rectangle('Position', ObjectCoordinates, 'EdgeColor', [0, 1, 0]);
    disp(['Selected ROI segmentation by hue [Xmin, Ymin, Width, Height, Xmax, Ymax]: ', num2str(ObjectCoordinates), ' ', num2str(ObjectCoordinates(1) + ObjectCoordinates(3)), ' ', num2str(ObjectCoordinates(2) + ObjectCoordinates(4))]);
    MeanValue = mean2(ImagesH{i}(ObjectCoordinates(2):ObjectCoordinates(2) + ObjectCoordinates(4), ObjectCoordinates(1):ObjectCoordinates(1) + ObjectCoordinates(3), :));
    StandardDeviationValue = std2(ImagesH{i}(ObjectCoordinates(2):ObjectCoordinates(2) + ObjectCoordinates(4), ObjectCoordinates(1):ObjectCoordinates(1) + ObjectCoordinates(3), :));
    disp(['m = ', num2str(MeanValue), ' ; s = ', num2str(StandardDeviationValue)]);
    ThresholdMinimum = p1 * MeanValue - p2 * StandardDeviationValue;
    ThresholdMaximum = p1 * MeanValue + p2 * StandardDeviationValue;
    ImagesSegmentedMask = ImagesHSV{i}(:,:,1) > ThresholdMinimum & ImagesHSV{
```

```
132▲ i}(:,:,1) < ThresholdMaximum;
ImagesSegmentedHue{i} = ImagesZ{i} + ImagesSegmentedMask;
if imagerender == 1
    figure;
    imshow(ImagesSegmentedHue{i}); title({['Segmented image using hue  ▼136
136▲ tresholding: ', 'T_m = ', num2str(ThresholdMinimum), ', T_M = ',  ▼136
136▲ num2str(ThresholdMaximum)]});
    RegionProperties = regionprops(ImagesSegmentedHue{i}, 'Area', ' ▼137
137▲ Centroid', 'BoundingBox');
    hold('on'); rectangle('Position', ObjectCoordinates, 'EdgeColor', [0, ▼138
138▲  1, 0]);
    testmax = 0; maxindex = 1; %Determining the optimal region box
    for k = 1:length(RegionProperties)
        testbox{k} = rectint(RegionProperties(k).BoundingBox,  ▼141
141▲ ObjectCoordinates);
        if testbox{k} > testmax; testmax = testbox{k}; maxindex = k; end
    end
    ULCW = RegionProperties(maxindex).BoundingBox; rectangle('Position',  ▼144
144▲ ULCW, 'EdgeColor', [1, 0, 0]);
    XCenter = floor(RegionProperties(maxindex).Centroid(1)); YCenter =  ▼145
145▲ floor(RegionProperties(maxindex).Centroid(2));
    plot(XCenter, YCenter, '+r', 'LineWidth', 2, 'MarkerSize', 12);
end
if imagesave == 1
    saveas(gcf, [dir, '/', 'L03_04_SEG_', num2str(i), '_l_', num2str(l)], ▼149
149▲  imageextension);
end
%% Segmentation by hue and saturation
%Unnecessary since using the same ROI in hue
%figure;
%imshow(Images{i});
%if manualroi == 1
%    title('H&S&P - Select the region of interest (ROI)');
%ObjectCoordinates = int16(getrect); % X, Y, Width, Height;
%else
%    title('Hue and Saturation custom segmentation');
%    ObjectCoordinates = [700, 388, 64, 16];
%    ObjectCoordinates = ObjectCoordinatesArrayParts{l}; %Object  ▼161
161▲ coordinates taken from the array
%end
rectangle('Position', ObjectCoordinates, 'EdgeColor', [0, 1, 0]);
disp(['Selected ROI custom segmentation [Xmin, Ymin, Width, Height, Xmax, ▼164
164▲  Ymax]: ', num2str(ObjectCoordinates), ' ', num2str(ObjectCoordinates ▼164
164▲ (1) + ObjectCoordinates(3)), ' ', num2str(ObjectCoordinates(2) +  ▼164
164▲ ObjectCoordinates(4))]);
MeanValueH = mean2(ImagesH{i}(ObjectCoordinates(2):ObjectCoordinates(2) + ▼165
165▲  ObjectCoordinates(4), ObjectCoordinates(1):ObjectCoordinates(1) +  ▼165
165▲ ObjectCoordinates(3), :));
MeanValueS = mean2(ImagesS{i}(ObjectCoordinates(2):ObjectCoordinates(2) + ▼166
166▲  ObjectCoordinates(4), ObjectCoordinates(1):ObjectCoordinates(1) +  ▼166
166▲ ObjectCoordinates(3), :));
MeanValue = (w11 * MeanValueH + w12 * MeanValueS) / (w11 + w12);
StandardDeviationValueH = std2(ImagesH{i}(ObjectCoordinates(2): ▼168
```

```
ObjectCoordinates(2) + ObjectCoordinates(4), ObjectCoordinates(1): ObjectCoordinates(1) + ObjectCoordinates(3), :));
StandardDeviationValueS = std2(ImagesS{i}(ObjectCoordinates(2): ObjectCoordinates(2) + ObjectCoordinates(4), ObjectCoordinates(1): ObjectCoordinates(1) + ObjectCoordinates(3), :));
StandardDeviationValue = (w21 * StandardDeviationValueH + w22 * StandardDeviationValueS) / (w21 + w22);
disp(['m = ', num2str(MeanValue), ' ; s = ', num2str( StandardDeviationValue)]);
ThresholdMinimum = p1 * MeanValue - p2 * StandardDeviationValue;
ThresholdMaximum = p1 * MeanValue + p2 * StandardDeviationValue;
ImagesSegmentedMask = ImagesS{i} > ThresholdMinimum & ImagesS{i} < ThresholdMaximum;
ImagesSegmentedSaturation{i} = ImagesZ{i} + ImagesSegmentedMask;
ImagesP{i} = imfilter(ImagesH{1}, fspecial('gaussian', [18 18], 18)) .* ImagesH{1}; %Blurring and multiplying hue component image by itself
ImagesSegmentedMask = ImagesP{i} > ThresholdMinimum & ImagesP{i} < ThresholdMaximum;
ImagesSegmentedProcessed{i} = ImagesZ{i} + ImagesSegmentedMask;
ImagesSegmented{i} = (w31 * ImagesSegmentedHue{i} + w32 * ImagesSegmentedSaturation{i} + w33 * ImagesSegmentedProcessed{i}) / ( w31 + w32 + w33);
%ImagesSegmentedHS{i} = ImagesSegmented{i} > ThresholdMinimum & ImagesSegmented{i} < ThresholdMaximum; %Thresholding is an alternative step
ImagesSegmentedHS{i} = (ImagesSegmented{i} - min(ImagesSegmented{i}(:))) / (max(ImagesSegmented{i}(:)) - min(ImagesSegmented{i}(:)));
if imagerender == 1
    figure;
    imshow(ImagesSegmentedHS{i}); title({['[Pseudo]Segmented image using saturation & hue tresholding: ', 'T_m = ', num2str(ThresholdMinimum), ', T_M = ', num2str(ThresholdMaximum)]});
    RegionProperties = regionprops(ImagesSegmentedHS{i} > ThresholdMinimum & ImagesSegmentedHS{i} < ThresholdMaximum , 'Area', 'Centroid', 'BoundingBox'); %Thresholding here for a binary image imperative for the regionprops function
    hold('on'); rectangle('Position', ObjectCoordinates, 'EdgeColor', [0, 1, 0]);
    testmax = 0;  maxindex = 1; %Determining the optimal region box
    for k = 1:length(RegionProperties)
        testbox{k} = rectint(RegionProperties(k).BoundingBox, ObjectCoordinates);
        if testbox{k} > testmax; testmax = testbox{k}; maxindex = k; end
    end
    ULCW = RegionProperties(maxindex).BoundingBox; rectangle('Position', ULCW, 'EdgeColor', [1, 0, 0]);
    XCenter = floor(RegionProperties(maxindex).Centroid(1)); YCenter = floor(RegionProperties(maxindex).Centroid(2));
    plot(XCenter, YCenter, 'xr', 'LineWidth', 2, 'MarkerSize', 12);
end
if imagesave == 1
    saveas(gcf, [dir, '/', 'L03_05_SEG_', num2str(i), '_1_', num2str(1)], imageextension);
```

```
        end
end;
%%%
end; %End object parts loop %Move this line upwards for comparison of all object parts at the same time
if lowmem == 1; if imageleave == 1; pause; end; close all; end %Move this line upwards to save memory
    %The following should run outside of the parametrizing loop
    %% Templates using normalized cross correlation
    figure;
    imshow(Images{i});
    if manualroi == 1
        title('NCC - Select the region of interest (ROI)');
        ObjectCoordinates = int16(getrect); % X, Y, Width, Height;
    else
        title('Normalized Cross Correlation');
        %ObjectCoordinates = [700, 388, 64, 16];
    if singleroincc == 1
        ObjectCoordinates = ObjectCoordinatesArray{singleroinccindex}(1:4);
    else
        ObjectCoordinates = ObjectCoordinatesArray{i}(1:4);
    end; end
    rectangle('Position', ObjectCoordinates, 'EdgeColor', [0, 1, 0]);
    disp(['Selected ROI normalized cross correlation [Xmin, Ymin, Width, Height, Xmax, Ymax]: ', num2str(ObjectCoordinates), '  ', num2str(ObjectCoordinates(1) + ObjectCoordinates(3)), '  ', num2str(ObjectCoordinates(2) + ObjectCoordinates(4))]);
    if singleroincc == 1
        Object{i} = ImagesGrayscale{singleroinccindex}(ObjectCoordinates(2):ObjectCoordinates(2) + ObjectCoordinates(4), ObjectCoordinates(1):ObjectCoordinates(1) + ObjectCoordinates(3), :);
    else
        Object{i} = ImagesGrayscale{i}(ObjectCoordinates(2):ObjectCoordinates(2) + ObjectCoordinates(4), ObjectCoordinates(1):ObjectCoordinates(1) + ObjectCoordinates(3), :);
    end
    MeanValue = mean2(Object{i});
    StandardDeviationValue = std2(Object{i});
    TwoDimCorrelation0 = filter2(Object{i}, ImagesGrayscale{i});
    TwoDimCorrelation0N = ((TwoDimCorrelation0 - min(TwoDimCorrelation0(:))) ./ (max(TwoDimCorrelation0(:)) - min(TwoDimCorrelation0(:))));
    TwoDimCorrelation1 = filter2(Object{i} - MeanValue, ImagesGrayscale{i});
    TwoDimCorrelation1N = ((TwoDimCorrelation1 - min(TwoDimCorrelation1(:))) ./ (max(TwoDimCorrelation1(:)) - min(TwoDimCorrelation1(:))));
    TwoDimCorrelation2 = filter2(Object{i} - StandardDeviationValue, ImagesGrayscale{i});
    TwoDimCorrelation2N = ((TwoDimCorrelation2 - min(TwoDimCorrelation2(:))) ./ (max(TwoDimCorrelation2(:)) - min(TwoDimCorrelation2(:))));
    disp(['m = ', num2str(MeanValue), ' ; s = ', num2str(StandardDeviationValue)]);
    NormalizedCrossCorrelation = normxcorr2(Object{i}, ImagesGrayscale{i});
    %HighestScore = max(max(NormalizedCrossCorrelation));
    %HighestScore = max(NormalizedCrossCorrelation(:));
```

```
 [HighestScoreCoordinates(1), HighestScoreCoordinates(2), HighestScore] = ▼237
237▲ ind2sub(size(NormalizedCrossCorrelation), find( ▼237
237▲ NormalizedCrossCorrelation==max(NormalizedCrossCorrelation(:))));
 if imagerender == 1
     figure;
     imshow(NormalizedCrossCorrelation);
     if singleroincc == 1;
         title({['NCC of the ', num2str(i), ' image with object from image ▼242
242▲  ', num2str(singleroinccindex) ' with highest score: ', num2str( ▼242
242▲ HighestScore), ' @ ', num2str(HighestScoreCoordinates(1)), ', ',  ▼242
242▲ num2str(HighestScoreCoordinates(2))]});
     else
         title({['NCC ', num2str(i), ' image, with highest score: ',  ▼244
244▲ num2str(HighestScore), ' @ ', num2str(HighestScoreCoordinates(1)), ', ▼244
244▲  ', num2str(HighestScoreCoordinates(2))]});
     end
     ObjectCoordinatesNCC = [ObjectCoordinates(1) + ObjectCoordinates(3) / ▼246
246▲  2, ObjectCoordinates(2) + ObjectCoordinates(4) / 2,  ▼246
246▲ ObjectCoordinates(3), ObjectCoordinates(4)];
     hold('on');
     %rectangle('Position', ObjectCoordinates, 'EdgeColor', [0, 1, 0]); % ▼248
248▲ ROI at original coordinates
     rectangle('Position', ObjectCoordinatesNCC, 'EdgeColor', [1, 1, 0]);  ▼249
249▲ %ROI at corrected coordinates
     plot(HighestScoreCoordinates(2), HighestScoreCoordinates(1), 'xr', ' ▼250
250▲ LineWidth', 2, 'MarkerSize', 12);
 end
 if imagesave == 1
     if singleroincc == 1;
     saveas(gcf, [dir, '/', 'L03_06_NCC_1OC_', num2str(i)], imageextension ▼254
254▲ );
     else
     saveas(gcf, [dir, '/', 'L03_06_NCC_', num2str(i)], imageextension);
     end
 end
 if imagerender == 1
     figure;
     set(groot,  'defaultFigureRenderer', 'opengl'); %Set to opengl for  ▼261
261▲ speeding up the process
     surf(NormalizedCrossCorrelation); shading flat; colormap jet;
     axis([0, size(NormalizedCrossCorrelation, 2), 0, size( ▼263
263▲ NormalizedCrossCorrelation, 1), -1, 1]);
     view(4, -84);
 end
 if imagesave == 1
     set(groot,  'defaultFigureRenderer', 'opengl');%painters');
     saveas(gcf, [dir, '/', 'L03_07_NCC_3D_', num2str(i)], imageextension) ▼268
268▲ ;
 end
 if imagerender == 1
     figure;
     title({['Conventional 2d correlation of the ', num2str(i), ' image' ▼272
272▲ ]});
```

```
        subplot(2, 2, 1); imshow(TwoDimCorrelation0N); title('2D correlation - normal'); colormap('gray');
        subplot(2, 2, 2); imshow(TwoDimCorrelation1N); title('2D correlation - mean'); colormap('gray');
        subplot(2, 2, 3); imshow(TwoDimCorrelation2N); title('2D correlation - standard deviation'); colormap('gray');
        subplot(2, 2, 4); imshow(NormalizedCrossCorrelation); title('NCC'); colormap('gray');
        ObjectCoordinatesNCC = [ObjectCoordinates(1) + ObjectCoordinates(3) / 2, ObjectCoordinates(2) + ObjectCoordinates(4) / 2, ObjectCoordinates(3), ObjectCoordinates(4)];
        hold('on');
        %rectangle('Position', ObjectCoordinates, 'EdgeColor', [0, 1, 0]); % ROI at original coordinates
        rectangle('Position', ObjectCoordinatesNCC, 'EdgeColor', [1, 1, 0]); %ROI at corrected coordinates
    end
    if imagesave == 1
        if singleroincc == 1;
        saveas(gcf, [dir, '/', 'L03_07_CC_1OC_', num2str(i)], imageextension);
        else
        saveas(gcf, [dir, '/', 'L03_07_CC_', num2str(i)], imageextension);
        end
    end
if imageleave == 1; pause; end %Pause between images
    close all;
end %End main loop
%    end; end %End parametrizing loop

%end %Function
```

The above script is commented in a self explanatory way and according to requirements of the laboratory session and beyond. Since the segmentation based on the hue component did not satisfy with its results, a great effort has been undertaken to try improve this simple approach.

Conclusions

The algorithm presented in the script provides segmentation based on the hue component, a custom segmentation based on the weighted hue and saturation components, lastly it incorporates the template matching technique using mainly normalized cross correlation. Since the simple approach of using only the hue component for segmentation of the object of interest had unsatisfying results, an expended custom algorithm has been implemented. The weighted hue and saturation components with additionally processed hue channel, resulted in almost 100% success rate for segmentation of the red car in the center of the image.

One aspect that should be considered here is the quality of the channels information. The hue components was noisy. The saturation channel was more suited for the task of segmenting the red car. However, of course other information could and maybe should have been considered. For example, from the selected region of interest, the most appropriate channel in RGB color space could have been selected as an additional input for the images to be weighted and thresholded. It would be wise to implement a preprocessing algorithm determining, which components to use based on their characteristics. Again, it all depends on the application and requirements. Additional operations like dilatation or closing with appropriate parameters could have been also performed on the given input images.

The template matching, in particular the normalized cross correlation used for this task, turned out to be a very powerful tool for recognizing patterns. However there are some limitations. Since computer vision is heavily used in the industry, especially with mass production, there are several problematic scenarios possible. For example, in car factory a camera could have the task to track identical components. But since they are identical, it could be problematic for the algorithm distinguish multiple elements from each other, or confuse their order if no further unique properties or additional information were given. However NCC is robust enough to match templates across several images of moving objects, which makes it very capable technique for a broad palette of applications.