Lab 3

Hanz Cuevas Velásquez, Bob Fisher
Advanced Vision
School of Informatics, University of Edinburgh

Week 4, 2018

This lab will focus on change detection using a simple background subtraction technique applied to a video.

We need to go inside the folder where we downloaded the lab material using the address bar of Matlab or using the terminal command `cd` in the Matlab command window. For this lab, we can observe that the `Images` folder contains 170 images which are the frames of the recording of a highway. The images were taken and modified from the 2016 IEEE Scene Background Modeling Contest (SBMC 2016) dataset[1].

The first step we will do is to load the images and store them in a cell. A cell can help us to store matrices in a single variable. We will also show the video of the highway:

```
path = 'Images/';
a = dir([path '*.png']);
im_cell = {};

for i=1:length(a)
    Img = im2double(imread([path 'highway_' num2str(i) '.png']));
    imshow(Img)
    im_cell{i} = Img;
end
```

In this lab we will see how a simple change detection method can help us to segment the cars that are on the highway.

1 Background subtraction

Background subtraction is a widely used technique for detecting moving objects in a video. This technique compares a main image, background or also called static image with the frames of a video to spot any change in the intensity of the pixels. If one of the frames has a different value than the background image, we can consider that the frame has a foreground object or moving object. For the lab, we will use the following method of background subtraction:

$$moving\_object = abs(frame - background) \tag{1}$$

Where the background will be the first image in our cell and the frame will be the rest, as seen in Figure 1 a) and b). Your task is to transform the RGB image into gray scale and apply the background subtraction formula.

```
% We will use the gray scale of our colour image
background = %(Code: transform the first image in the cell into gray scale)
for i=2:length(im_cell)
    b_s = %(Code: use the background subtraction formula) Note: don't forget to
        transform the rest of the images in the cell to gray scale.
    %We threshold the image to create a mask.
```

[1] http://scenebackgroundmodeling.net/

```
    I=b_s;
    I(I>0.3)=1;
    I(I<=0.3)=0;
end
subplot(2,2,1), imshow(background);
subplot(2,2,2), imshow(rgb2gray(im_cell{end}));
subplot(2,2,3), imshow(b_s);
subplot(2,2,4), imshow(I);
```

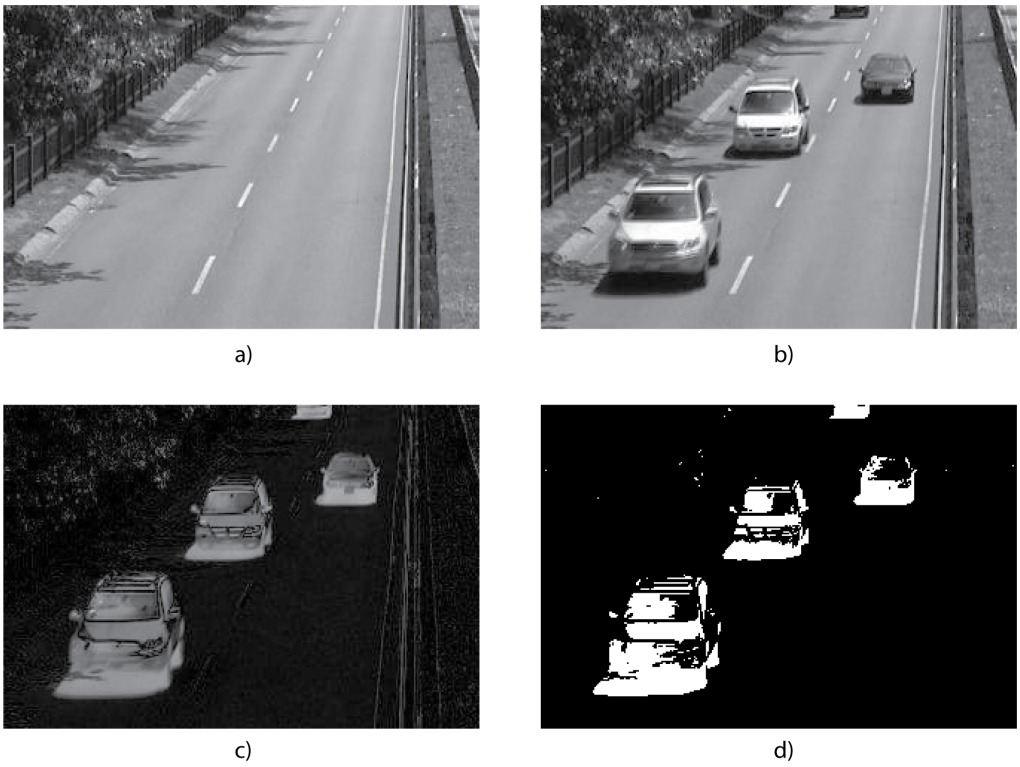

Figure 1: Background subtraction method a) Background, b) Frame 170, c) Background subtracted, d) Mask after intensity threshold.

As we can observe in figure 1 c). The values of the pixels are high if there is a foreground object and is low otherwise. However, we can also observe that this technique is really sensitive to the change of illumination in the scene and small motions like the one the tree does in our video (see the white spots in the background subtracted image were the tree is).

2 Cleaning the mask

As stated in the previous paragraph, our mask has some noise created by different factors; we have to clean them. One simple but powerful tool to do so are the morphological operations we learned last week. However, we have to think about which methods we will use. In our case. First, we eliminate the noise using the `bwareaopen` command. This eliminates all the closed areas less than a threshold. We observe that the cars are not perfectly segmented after subtracting the background. This is because some values of the pixels of the cars are equal to the values of the pixels of the background. In some cases, this makes the top of the car be completely separated from the car. To join them together, we will use dilation to expand the foreground part of the mask and `imclose` to join the foreground segments that are close to one another according to a threshold. Finally, we will fill the holes that exist inside a foreground area. The task for this section is the following:

eliminate the noise in the mask, dilate the image, join the foreground segments, and finally fill the holes. Keep using your previous code.

```
background = %Code from section 1
%%Add from here
SE=strel('square',5); %Use it for the dilation
SE1=strel('rectangle',[10, 5]); %Use it for the imclose operation
%%Add to here
for i=2:length(im_cell)
    b_s = %Code from section 1;
    I=b_s;
    I(I>0.3)=1;
    I(I<=0.3)=0;
    %(Code: Start writing your code here, you only need to use the variable I (the
        mask) as input)
end
subplot(2,2,1), imshow(background);
subplot(2,2,2), imshow(im_cell{end});
subplot(2,2,3), imshow(b_s);
subplot(2,2,4), imshow(I);
```

If we compare the first mask we created, figure 1 d), and the new cleaned mask, figure 2 b), we will see that it improved considerably.

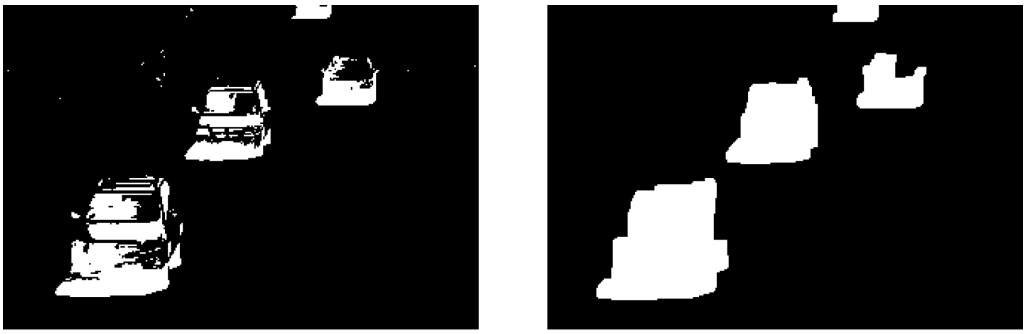

Figure 2: a) Background subtracted, b) Background subtraction method after refinement.

3 Drawing bounding boxes on frames

The final step is to create a bounding box around our cars similar to the one we did in the previous lab. Here, we will write a function that embeds all the data we need for the bounding box and add it at the end of our for loop to show the images. Your task is to complete the function `show_cars`, you have to find the positions of the bounding boxes for the cars in each frame of the highway video. Once completing the code, you should be able to visualize all the frames with bounding boxes on the cars similar to the one shown in figure 3

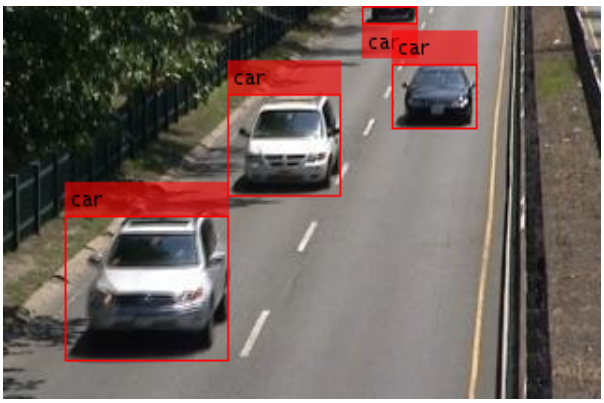

Figure 3: Final image.

```
function show_cars(I, im_cell, tbf)
    %I: The mask that segments the cars.
    %im_cell: an RGB image (frame of the video).
    %tbf: time between frames.
    lbl = bwlabel(I);
    position=[];
    label_str={};
    for j=1:max(max(lbl))
        %(Code: find the position of the bounding boxes)
        position = %Concatenate the positions of each car
        label_str{j} = ['car'];
    end
    if isempty(label_str)==0
        rgb=insertObjectAnnotation(im_cell, 'rectangle', position, label_str, '
            Color', 'red', 'FontSize', 11);
        imshow(rgb);
    else
        imshow(im_cell)
    end
    pause(tbf)
end
```

```
background = %Code from section 1
SE=strel('square',5); %Use it for the dilation
SE1=strel('rectangle',[10, 5]); %Use it for the imclose operation
for i=2:length(im_cell)
    b_s = %Code from section 1
    I=b_s;
    I(I>0.3)=1;
    I(I<=0.3)=0;
    %Code from section 2

    %add the function created above
    tbf = 0.01; %time between frames. You can modify it to pass each frame slower
    show_cars(I, im_cell{i}, tbf)
    %%Add to here
end

%Delete the subplots
%subplot(2,2,1), imshow(background);
%subplot(2,2,2), imshow(im_cell{end});
%subplot(2,2,3), imshow(b_s);
%subplot(2,2,4), imshow(I);
```