

Ben Gurion University

***FIGURE GROUND SEGREGATION IN VIDEO VIA AVERAGING AND COLOR DISTRIBUTION.***

Final project by:

Dror Zenati

## 1. Introduction

*1.1 Motivation:*

Sometimes it's quite important to be able track an object in a given video (tracking drivers in the road, identifying moving objects in night vision video etc.). The course gave us tools for segmenting the figure in a given image. But, what are the approaches for segmenting a figure from a set (>1) of images (I.e. video file)?.

In my project i investigated the approaches for doing so and tried to achieve high quality of figure ground segregation.

*1.2 The Goal:*

As said, the project main goal was to achieve a high quality of figure ground segregation while the figure is considered to be moving in some direction and the ground is static. The algorithm should be generic (applies to all videos that fulfills the assumptions i made).

*1.3 Assumptions:*

There are many variations of videos. Therefore, I made a few assumptions about the given video:

* Background: the background can be either known or unknown. In my work, i didn't make any assumptions about the background and i considered it as *unknown*.
* Camera: the video may be taken from a stationary camera or moving camera. In my work, I restricted it for a *stationary camera*.
* Lighting: the lighting throw the video may be either fixed or varying. I used a technique that allows a difference in the lighting throw the video as long as it varies in a reasonable speed (will be explained later).

## 2. Approach and Method

It can be said that in order to achieve the goal my algorithm is divided into 4 steps:

**Step 1 – Averaging:**

Videos are in fact a set of frames. In order to locate the moving object it was necessary to average the frames. Assuming that the background is stationary, doing so allowed me to locate the object in the next steps.

At the beginning I averaged the entire video frames throw the video into one image that represents the average. But, later I discovered that it is not a good idea since throw all the video each pixel color value is changing due to many parameters such as – lighting, noise, shadows and more. So, i decided to divide the video to a set of adjacent frames and each "local" set of adjacent frames were averaged separately. This technique allowed a changing in lightning throw the video without affecting the segmentation.

Another improvement was to average block of pixels into one value and than average adjacent blocks. I discovered that working in pixel resolution is quite not efficient and not so accurate.

**Step 2 – segregation throw color distribution:**

After averaging i found out that in order to identify the object I can use a technique called "segmentation throw color distribution". The idea is simple, for each block compute the absolute difference between the block values and the corresponding average, for all 3 components (RGB):







Combine those values into one value that represents the difference:



Finally, locate the figure throw thresholding:



**Step 3 – locating object components:**

In this stage i had a sketch of the figure I want to segment but it wasn't accurate enough since there were a lot of noises. So, at this stage i decided that only figures with size bigger than 24\*24 pixels will be considered as object (this parameter is an input to my algorithm and can be changed). For each frame I found the pixels that fulfill this condition.

**Step 4 – "magic wand":**

When I got to this stage i hade, for each frame, the blocks that belongs to the object. But, the frames were divided into blocks so the curves looked not so accurate (like rectangles). Now, i developed (with an open source I found that I adjusted to my needs) a method that takes pixel and find all the pixels in the area that correspond to its color. I gave this tool the pixels i found in the previous stage the output was a binary mask of all the object pixels that were lost in the previous stages. The final output of my algorithm was (for each frame):



In order to present the output of my algorithm i put together all this frames into an .AVI file with the figure segregated from the ground (in black).

## 3. Results

I will show my results step after step throw some examples i chose:

The original movies:

## test1_f29_original.JPGtest3_f42_original.PNGtest2_f44_original.PNG

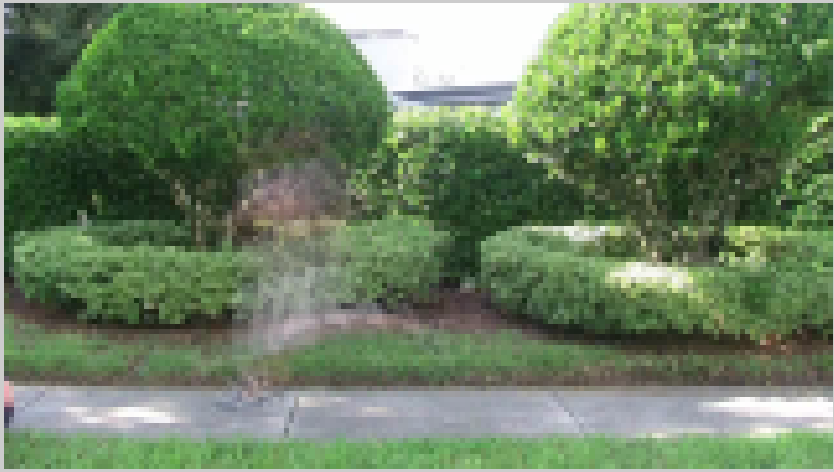
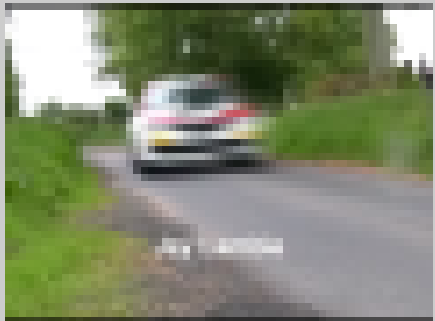
(Frame 25 out of 44)

(Frame 44 out of 61)

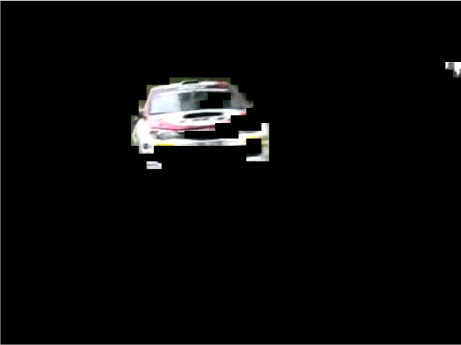
(Frame 42 out of 66)

After Averaging:

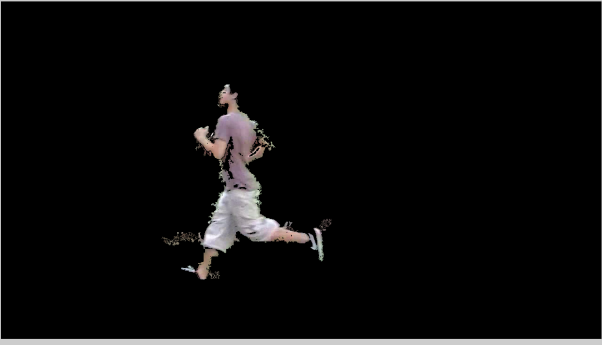
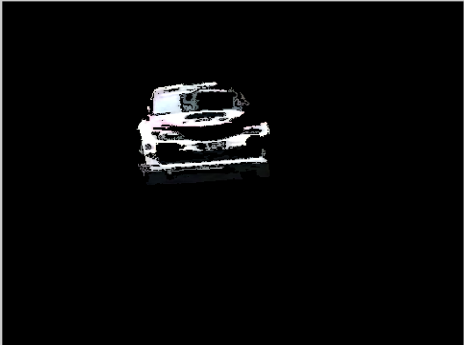
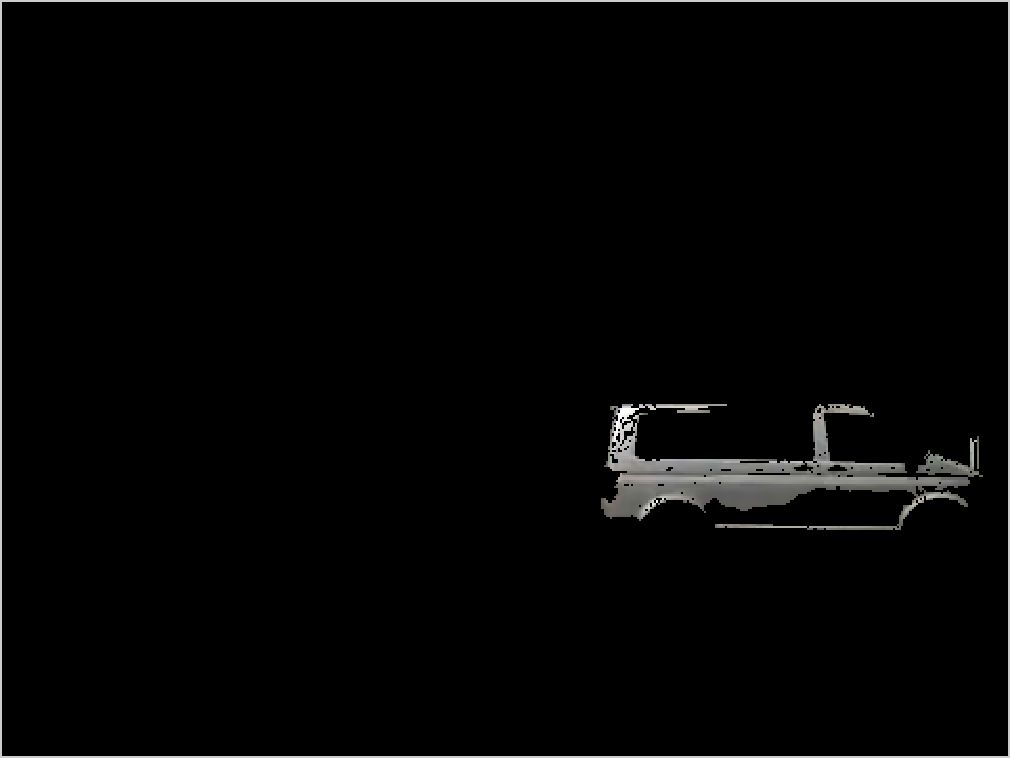




After segregation throw color distribution:



"Magic wand":



It can be seen that more than 90 percent of the first figure is shown (and a good results in the others too). Moreover, there are no noises in the background.

**4. Conclusions**

My goal was achieving good quality segmentation. As we learned in class, segmentation is subjective and not an easy task. My algorithm take a few parameters such as: threshold for the figure ground segregation, block size for averaging, another threshold for the magic wand tool and more. Choosing different values affects directly on the results. Therefore, the quality is a matter of trial and error and can be changed according to needs.

Other things that may affect the quality and are not related to my algorithm are the object *speed*. As faster it moves as it segmented in a higher quality. The object *location* throw the video also affects the quality. In order to achieve better result the object need to change his position throw the video. Otherwise, most of the time he will considered as background. Also, the *size* of the figure and the *color* that composes it may affect the results.

**5. Functionality**

Here are some explanations about the main methods on my project.

*5.1 Magic wand*

function bin\_mask = magicwand(im, ylist, xlist, tolerance)

% MAGICWAND simulates the Photoshop's magic wand tool.

% It allows selection of connected pixels whose colors are

% within a defined tolerance of reference pixels.

%

% SYNTAX

% bin\_mask = magicwand(im, ylist, xlist, tolerance);

%

% INPUT

% im: input image RGB

% ylist: vector of row cordinates (reference pixels)

% xlist: vector of column cordinates (reference pixels)

% tolerance: distance to reference pixels

%

% OUTPUT

% bin\_mask: binary mask of selected regions

For example:

The following code selects the girl's face and applies magic wand on two pixels on her face:

im = imread('test.jpg');

bin\_mask = magicwand(im, [199 217], [318 371], 50);

subplot(1, 2, 1); imshow(im);

subplot(1, 2, 2); imshow(bin\_mask);



*5.2 Average adjacent frames*

function averagedFrames = averageadjacentframes(mov, blockSize, nFramesForAverage)

% AVERAGEADJACETFRAMES apply averaging in a given movie.

% in order to overcome the noise between adjacent images it

% divides the frame into blocks, average each block and then

% averages the block with the corresponding blocks.

%

% SYNTAX

% averagedFrames = averageadjacentframes(mov, blockSize,

% nFramesForAverage)

%

% INPUT

% mov: the movie given in an AVI format.

% blockSize: size of block that the frames is divided to.

% nFramesForAverage: number of frames to average.

%

% OUTPUT

% averagedFrames: array of averaged frames.

For example:

The following code gets a movie and applies averaging on it. The image shows the 8'th item means the average of figures 29-32 (each 4 frames are averaged).

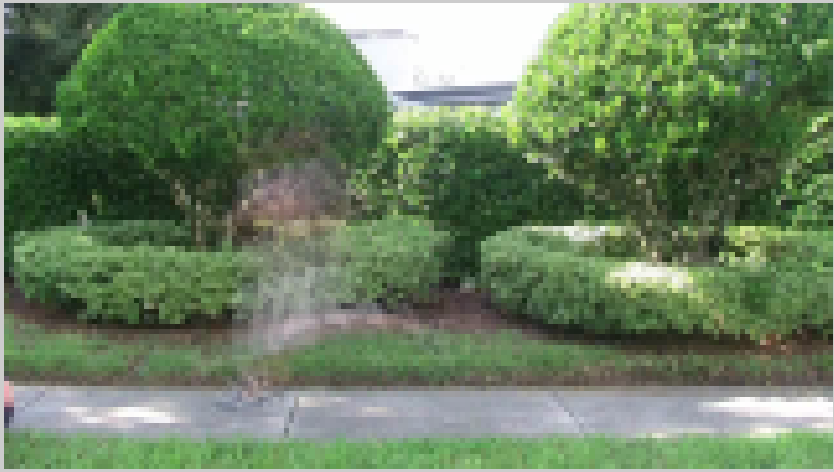
mov = VideoReader('test1.avi');

blockSize = 8;

nFramesForAverage = 4;

averagedFrames = averageadjacentframes(mov, blockSize, ...nFramesForAverage);

imshow(averagedFrames{8});



*5.3 Segregation by color distribution*

function FramesArrayAfterSegragetion = segregationbycolordistribution...

(mov, blockSize, averagedframes,treshold)

% SEGREGATEBYCOLORDISTRIBUTION returns, for each frame,

% the segragated frame. i.e 0 for blocks considered as

% background ans I(x,y) for blocks considered as the figure.

%

% SYNTAX

% framesArrayAfterSegragetion = segregationbycolordistribution...

% (mov, blockSize, averagedFrames,treshold);

%

% INPUT

% mov: the movie given in an AVI format.

% blockSize: size of block that the frames is divided to.

% averagedFrames: frames after averaging.

% treshold: figure if I(x,y) - average(x,y) > treshold

% otherwise - background.

%

% OUTPUT

% FramesArrayAfterSegragetion: segregeted frames.

For example:

The following code returns the segregated frame of frame 29 (the acove example).

mov = VideoReader('test1.avi');

blockSize = 8;

nFramesForAverage = 4;

averagedFrames = averageadjacentframes(mov, blockSize,... nFramesForAverage);

treshold = 0.1;

framesArrayAfterSegragetion = segregationbycolordistribution...

(mov, blockSize, averagedFrames,treshold);



**6. User manual**

1. Open main.

2. Select values for the constant variables:

BLOCK\_SIZE: usually select 4 or 8.

FRAMES\_FOR\_AVERAGE: usually select 2 or 4.

TRESHOLD: 0.1-1 is optimal.

TOLERANCE: 10 – 30.

3. Run it.

**7. References**

[1]. Ohad Ben Shahar, “Perceptual Organization : Segmentation”

<http://www.cs.bgu.ac.il/~ben-shahar/Teaching/Computational-Vision/LectureNotes/ICBV-Lecture-Notes-54-Perceptual-Organization-4-Segmentation-1SPP.pdf>

[2]. Leow Wee Kheng, “Computer vision and pattern recognition”

http://www.comp.nus.edu.sg/~cs4243/lecture/removebg.pps

[3]. Hongazhi Wang and John Oliensis, "Rigid Shape Matching

by Segmentation Averaging"

http://www.cs.stevens.edu/~oliensis/shape.pdf