EECS451 Project Final Report

Vehicle Detection and Classification

Shihong Huang
Department of Civil and Environmental Engineering
University of Michigan
Ann Arbor, Michigan
edhuang@umich.edu

I. PROJECT DESCRIPTION

My graduate research is mainly focused on transportation engineering. So in this project I will use Digital Signal Processing (DSP) tool to achieve a goal that is related to transportation engineering.

The objective of this project is to detect and track vehicles in a video and to classify if these vehicles are long vehicles or regular vehicles. The video will be recorded from a surveillance camera at a fixed perspective. And the camera is aimed at the central part of the intersection. Then the vehicles passing through the intersection will be recorded. MATLAB will be used as the main software in this project.

II. DATA DESCRIPTION

The video was recorded by Minnesota Transportation Observatory with the help from Minnesota Department of Transportation. The video was recorded on April 7, 2014. The intersection of interest was Trunk Highway 13 at Diffley Road in Minnesota.

The size of this one-hour-long video is about 633MB. Fig 1 is a frame taken from the video, showing a vehicle making the left-turn. The actual resolution is 640*480 pixels.

Fig 1 Original data

However, this video is too big for MATLAB to load and executive the DSP commands. To simplify this project, a software called Format Factory is used to intercept the video. Finally the sample video is 50 seconds long and 2.45MB big. The sample video is recorded at the rate of 10 frames per second. And the new video has a resolution of 296*448 pixels. Fig 2 shows a frame of the sample video.

Fig 2 Sample data

III. VEHICLE DETECTION

A. Step #1 Turn the frames into grayscale

The first step is to load the video in MATLAB. The video frames are represented as 3-dimensional matrices. To make later work easier, the sample RGB frames are converted into grayscale. Then, the values in each frame matrix are converted into double.

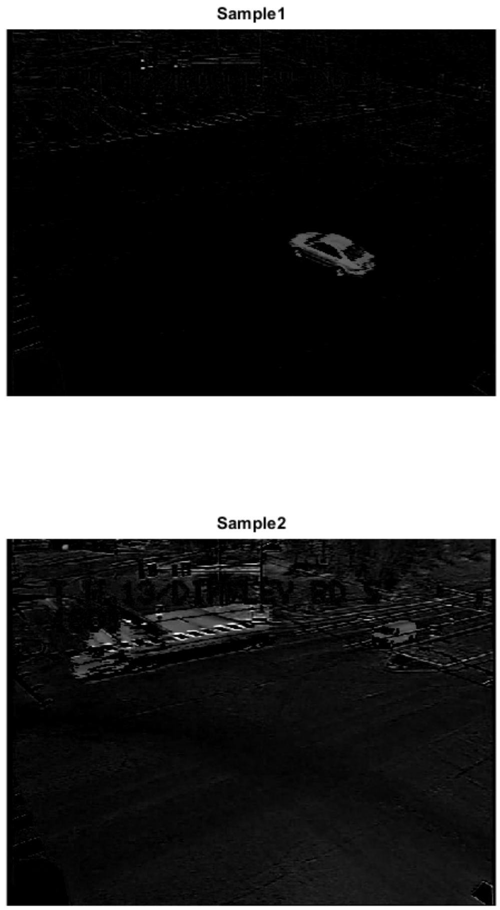
Now each frame in MATLAB is represented as a 2-dimensional matrix and each value in the matrix means the gray level of that particular pixel, with 1 being white and 0 being black.

B. Step #2 Differentiate the background

To detect a vehicle from a frame, it is very necessary to differentiate the background and the moving vehicle. A simple idea is to use background subtraction. To do so, a background frame is arbitrarily chosen from the video when there is no vehicle present. Then each frame is subtracted by the background frame. As shown in frame Sample1 in Fig 3, a moving vehicle can been easily identified from the frame.

However, it turns out that this method does not work for every frame. As shown in Fig 3, frame Sample2 has a lot of noises, which are possibly caused by the vibration of the surveillance camera. And this kind of noise could been seen throughout the video. Using a filter does not help solve this problem. A better way to identify the vehicles is indeed needed.

Fig 3 Samples after subtraction



To avoid this kind of noises, a method called Singular Value Decomposition (SVD) is used to decompose the sample video frames and to find out the background frame. SVD[1] states that an $m * n$ matrix can be decomposed as

$$A_{m*n} = U_{m*m}\Sigma_{m*n}V_{n*n}^T \quad (1)$$

Where

A_{m*n} is an $m * n$ matrix,

U_{m*m} is an $m * m$ unitary matrix,

Σ_{m*n} is an $m * n$ diagonal matrix with non-negative real numbers on the diagonal,

V_{n*n}^T is the transpose of the $n * n$ unitary matrix V_{n*n}.

Then matrix A_{m*n} could be written as a sum of rank 1 matrices[2]

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_n u_n v_n^T \quad (2)$$

Where

σ_i is the ith singular value,

u_i is the ith column of U_{m*m},

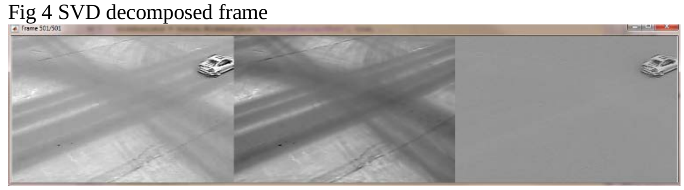
v_i is the ith row of V_{n*n}^T.

Each rank 1 matrix $u_i v_i^T$ is the size of the original matrix and the singular values are ordered $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Each one of these matrices is a mode. The original image could then be reconstructed from just a subset of modes (this is also the technique of image compression).

In this project, in order to implement the SVD tool in MATLAB, all the frames are first converted into one matrix using "reshape" command. Then SVD tool is used to decompose the matrix. Only the first mode, i.e. $\sigma_1 u_1 v_1^T$, is used as the background matrix. And the moving object, i.e. the vehicle, could be detected using background subtraction. The background matrix and moving object matrix are then converted back to frames, again using "reshape" command.

Fig 4 shows the result by using the SVD tool in MATLAB. The left frame is the original grayscale frame. The middle one is the background, i.e. the first mode. And the right frame is the detected vehicle by subtracting the background. Apparently this method works very well. The moving object has been clearly extracted from the original frame. And the vibration of the camera does not affect the results.

Fig 4 SVD decomposed frame



C. *Step #3 2-D fft*

To see what happens in the frequency domain, let's take the fft of the grayscale frame. The 2-D fft is given by

$$X[k,l] = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x[m,n]e^{-j\frac{2\pi}{N}km}\, e^{-j\frac{2\pi}{N}ln} \quad (3)$$

Where

$x[m,n]$is the pixel of the frame.

In MATLAB, command fft2 is used to do the calculation. Besides that, we take the logarithm of the magnitude for perceptual scaling. Fig 5 shows the 2-D fft of a frame when there is no vehicle present and Fig 6 shows when there is a vehicle present. Apparently when there is a vehicle, the magnitude plot shows more "snows".

Fig 5 2-D FFT of a frame when there is no vehicle

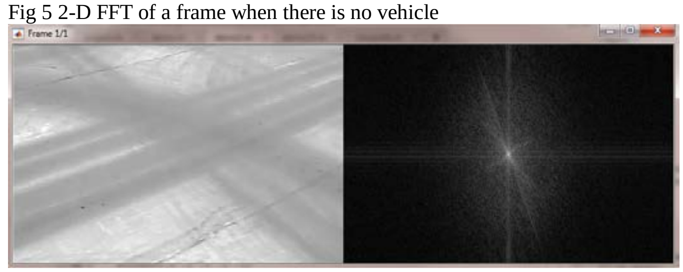


Fig 6 2-D FFT of a frame when there is a vehicle

D. Step #4 Filter

It is better to filter the noises before doing anything. A 2-D Gaussian filter is used in this case.

$$g(m,n) = \frac{1}{2\pi} e^{-m^2-n^2} \tag{4}$$

Then the filter matrix is defined as

$$\begin{bmatrix} g(-2,-2) & g(-2,-1) & g(-2,0) & g(-2,1) & g(-2,2) \\ g(-1,-2) & g(-1,-1) & g(-1,0) & g(-1,1) & g(-1,2) \\ g(0,-2) & g(0,-1) & g(0,0) & g(0,1) & g(0,2) \\ g(1,-2) & g(1,-1) & g(1,0) & g(1,1) & g(1,2) \\ g(2,-2) & g(2,-1) & g(2,0) & g(2,1) & g(2,2) \end{bmatrix}$$

This is a lowpass filter.

To take 2-D convolution, we use

$$y[m,n] = h[m,n] * x[m,n] \tag{5}$$

$$= \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} h[m-k, n-l] x[k,l]$$

In MATLAB, the easy way is to use the command conv2. Fig 7 shows the filtered frame (on the right) comparing to the original frame (on the left). However, we cannot see any obvious difference. Actually the filter is not that helpful in this particular project. A reasonable guess would be that the SVD tool has already filtered some noises.

Fig 7 Filtered frame

E. Step #5 Object detection and classification

To really show that our object of interest has been found, a rectangle is put into the frame to show that a vehicle is indeed being identified. To do so, we first need to turn the right frame in Fig 4 into a binary matrix. A threshold is set so that the each pixel is compared to the median value of the matrix plus or minus the threshold and the pixel is set either black (as the background) or white (as the object).

$$\text{pixel}[m,n] = \begin{cases} 0, & med - t \le \text{frame}[m,n] \le med + t \\ 1, & otherwise \end{cases} \tag{6}$$

Where

med is the median value of the matrix,

t is the threshold.

This process is done for every frame in the video (the median value may be different for each frame, but the threshold is the same). Fig 8 shows the frame when it is converted to binary.

Fig 8 Binary frame

Then, a MATLAB tool called blobAnalysis is used to help detect the object. Basically, Blob Analysis block calculates statistics for labeled regions in a binary image and the block returns quantities such as the centroid, bounding box, label matrix, and blob count[3].

In this project, the attribute BoundingBoxOutputPort is set to be true (it returns the coordinates of blob centroids), the attribute AreaOutputPort is set to be false (we do not need the area of the object), the attribute CentroidOutputPort is set to be false (we do not need the coordinates of blob centroids of the object) and the attribute MinimumBlobArea is set to be 100 (only the object that has an area of over 100 pixels will be considered as the object of interest).

Now that the blobAnalysis has been defined, it could be used to find the object in the binary frame. The information of the identified object is restored in a matrix called bbox in MATLAB. Information includes centroid, length and width of each bounding box. Then black rectangles are inserted into the frames according to bbox of each frame.

This whole process is done for each frame by using a loop function.

Fig 9 Detecting the object

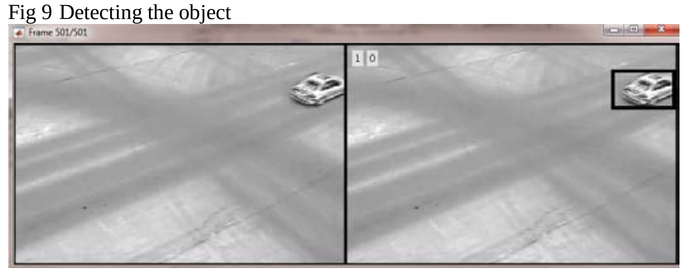


Fig 9 shows the result. The left frame is the original frame and the right frame shows the vehicle being detected.

At the up left corner there are two numbers. The first number means the number of vehicles detected in this frame and the second number means the number of long vehicle. Since we have identified the moving vehicle, the bounding box could somehow reflect the length of that vehicle. By checking the length of the bounding box, a long vehicle could be identified. In this project, a vehicle with its bounding box longer than 150 pixels is defined as a long vehicle. This definition is pretty simple and arbitrary but it works well. To differentiate a long vehicle from a regular vehicle in the frames, a white box instead of a black box is used to identify the vehicle, as shown in Fig 10.

Fig 10 Detecting long vehicle

IV. SUMMARY

This project detects and tracks vehicles in a video that is recorded by a surveillance camera with a fixed perspective. The objective is achieved successfully by using MATLAB. Vehicles that passing through the intersection are all captured and rectangular boxes are inserted into the video frames in order to highlight these vehicles. What's more, long vehicles are identified from regular vehicles by using white boxes. We also count the number of vehicles in each frame.

The in-class tools used in this project are background subtraction, 2-D fft, 2-D convolution and filter. The out-class tools used in this project are SVD and blobAnalysis. Detailed information about the tools and their functions could be found in Table 1.

TABLE 1 TOOLS USED IN THIS PROJECT

	Tool name	Function
In-class tool	Background subtraction	Differeniate background and moving object
	2-D fft	See the spectrum of the frame
	2-D convolution	Filter noises in the frame
	Gauss Filter	Lowpass filter
Out-class tool	SVD	Decompose the video frame, differeniate background and moving object
	blobAnalysis	Identify the moving object, achieve its coordinates and size in the frame

The SVD is really powerful and awesome. It helps on finding the background of each frame. In a lot of movies, the hackers hack into the closed-circuit television system and cover their friends by removing the images of their friends out of the surveillance camera. I believe that's exactly what the SVD tool does!

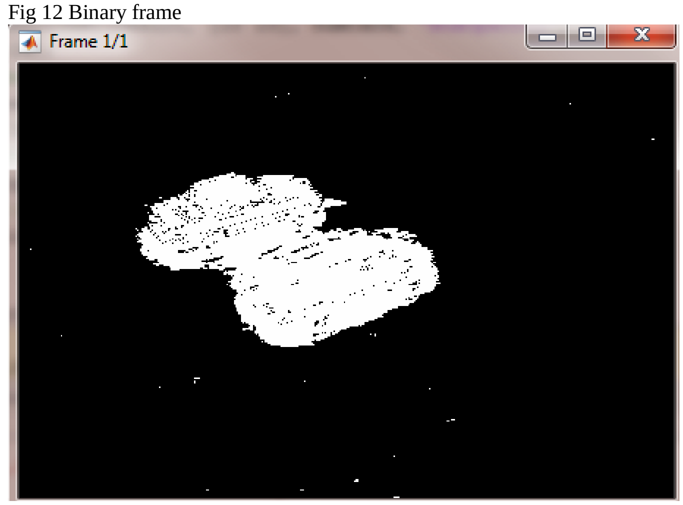
BlobAnalysis is also powerful. It could identify an object from a binary picture. Long vehicle is also identified by checking the length of the bounding box.

However, my algorithm is not perfect. One thing need to be mentioned here is that the SVD tool is computational expensive. For this short sample of video (50 seconds, 501 frames, 296*448 pixels), it takes several minutes to execute the SVD command on my laptop (my laptop is very old, though). As long as we find the background image, we could implement this algorithm to identify vehicles.

Besides that, the blobAnalysis tool sometimes fails to distinguish two objects when they are too close to each other. Fig 11 is an example where two vehicles are identified as one single vehicle. By checking the binary frame shown in Fig 12, we can see the limitation of blobAnalysis.

Fig 11 False identification

Fig 12 Binary frame



But overall, I have found a lot of new functions in MATLAB and MATLAB is indeed a powerful tool to engineering students. To apply what I learnt from class into practice is of great fun!

REFERENCES

[1] http://en.wikipedia.org/wiki/Singular_value_decomposition

[2] M. Spiegelman, "Applications of the SVD," unpublished. http://www.columbia.edu/itc/applied/e3101/SVD_applications.pdf

[3] http://www.mathworks.com