Matlab Project #4 – Background Subtraction

Introduction

In this project we'll implement the Background Subtraction (using Temporal Median) and use it for real-time detection of moving objects in video acquired by the webcam.

Note: before starting to work on this Project, it is recommended to answer questions in Lab 3, Parts 1 and 2.

Description of the project

- *Preliminary work: Webcam setup*

Use the command **doc** in Matlab command window. Then select:
Image Acquisition Toolbox→Getting Started
and read the information in the section **Tutorials.**
Also, use the command **doc** in Matlab command window. Then select:
Image Acquisition Toolbox→Examples
and read the different examples.

You have to learn how to grab video from your webcam and to process this video frame by frame.

- *Background Subtraction*

In lecture we studied the following algorithm for Background Subtraction:

$$D = (|I - B| > \theta),$$

where I is the input image (video frame), B is the background image, θ is a threshold (positive scalar constant) and D is the foreground binary mask ($D = 1$ at the "foreground pixels" of I and $D = 0$ at the "background pixels" of I).

- *Background*

The question arises how to estimate the background image B. We understand that the background is "something that doesn't change in the movie", that is, "the pixels that remain constant over time", that is, "the pixels that are similar to their temporal mean or temporal median". Usually, people prefer to work with temporal median and not temporal mean (because mean induces blurring while median preserves sharpness).

So let's define the background image B as follows:

$$B = \boldsymbol{MEDIAN}(I),$$

where the operator $\boldsymbol{MEDIAN}$ denotes per-pixel median in 1D temporal window of K frames (K is odd integer, default value: $K = 31$).

That is, we stack the frames one over another, and for each pixel in the "current frame" we "pierce" the stack of frames with the "needle" of length K, then compute the median of the pixels that we have "pierced", and this median will be the value of background for this pixel for the "current" frame. For the "next" frame we have to recompute the median for each pixel.

- *Fast recursive computation of median*

In principle, median can be computed "by definition": for every frame i, for every pixel P take the vector of $K-1$ "previous" and one "current" values of this pixel, sort these values, take the "medial" pixel and write it in the "result" frame i at the coordinate of the pixel P. This "brute-force" approach is correct, but computationally inefficient. (Recall that we acquire the video from the webcam and want to process this video "immediately", computing the real-time result after a small "initialization delay".) If we want to compute in real time the "running" temporal median for all frames and all pixels within each frame, we have to use fast recursive method. Indeed, if we move from the "current" frame to the "next" frame, then for each pixel the "K-vector" is changed by only two values: the "oldest" pixel is removed and the "newest" pixel is inserted. So in order to compute the "next" median we need to "smartly update" the "current" median (and not to sort the "next" "K-vector" "from scratch").

- *"Smart update" of the median*

Let's solve an auxiliary problem - let's start with the "spatial" 2D window of size $M \times N$ (and not with "temporal" 1D window of size K as needed). Recall that in Lab 2, Part 1, Question 8 we described an algorithm for computation of median from the histogram. So let's start with the fast computation of the histogram. In Lab 4, Part 2, Question 2 we described an algorithm of recursive computation of histogram in running window. Also, note that if we just remove the "oldest" column and insert the "newest" column, we can treat this as a set of operations "remove one pixel & insert one pixel". After one such operation the median does not jump too much, it almost stays in place and can be computed from the histogram using only two variables: mdn (the value of median in the "current window") and $ltmdn$ (the number of pixels in the "current" window with values less than median). The full details of recursive update of median in a running window of size $M \times N$ (using recursive update of the histogram) are provided in [1]. (Note: In the paper [1] the window width is $window.xsize$ and the window height is $window.ysize$, so $M = window.ysize$ and $N = window.xsize$.)

Now let's return to our case. We need a simpler algorithm than described above, since we need a 1D temporal median instead of the 2D spatial median.

Questions

1. At first, before starting to work with video, we'll work with computation of running median of a 1D vector. Provide the recursive algorithm (based on [1] simplified to the 1D case) for computation of median of a 1D vector in a running window of length K.
2. Implement this recursive algorithm in Matlab. (Note: you have to provide the "user-defined" "vector size" K.)

3. Before starting to compute running median, we have to read K values. Based on Lab 4, Part 1, Question 7, implement in Matlab the improved “super-fast” method that will start to compute running median after reading of $(K+1)/2$ values.
4. Now we go from computation of one running median in 1D vector to computation of “a set of running medians” (a running median for each pixel in a frame) in a video acquired by the webcam. For each frame (starting from frame #$(K+1)/2$) compute the background B of the video acquired by the webcam (using the “super-fast” method described above). The result of this program is the “**Background movie**”. (Note for the case of video processing: in the “super-fast” method we read $(K+1)/2$ frames and then start to output the values of medians.) (Tip: you may find useful to start working on this program with the “usual” video file. When this program will work, change the input from the video file to the “live” video acquired by the webcam.)
5. For each frame (starting from frame #$(K+1)/2$) compute the foreground binary mask D of the video acquired by the webcam. (Note: you have to provide the “user-defined” threshold θ.) The result of this program is the “**Foreground mask movie**”.
6. Did you manage to perform computations in real-time?

Deliverables

Provide the descriptions of the required algorithms, the code of the required programs and several “representative” frames from “Input movie” (from the webcam), “Background movie” and “Foreground mask movie”.

Important note – optional simplification

If you don’t succeed to make a working “real-time” project with the webcam and fast (recursive) median, you can do the simplified “non-real-time” project with the video file and the “brute-force” median algorithm. (Note: of course, you have to answer the questions regarding the “real-time” project.) The maximal grade of the simplified “non-real-time” project is **90**.

Submission

You have to submit the Project Report (with the code included in printed form) at 28/05/2013 at the lab (from 13:00 until 14:00) or on the Video Processing lesson (at the breaks between the lectures).

References

[1] T. S. Huang, G. J. Yang, and G. Y. Tang, “A fast two-dimensional median filtering algorithm,” *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 27, no. 1, pp. 13–18, Feb. 1979, available at:

http://www.uio.no/studier/emner/matnat/ifi/INF2310/v12/undervisningsmateriale/artikler/Huang-etal-median.pdf