CSC 449 – Homework 4
Phyo Thiha

Introduction

In this assignment, we implemented background subtraction to track a moving object. We calculate the bounding box of the background subtracted object, and use it as measurement parameters in 2D Kalman filter implementation. We present our findings in the following order: list of files we implemented for this assignment under ‘Relevant Files’; explanation of our implementation in ‘Methods’ section; selected presentation of our results and discussion of our findings in ‘Results and Discussion’ section and finally, a list of resources we used for this assignment.

Relevant Files

approximate_median.m	- background subtraction with median approximation
frame_difference.m	- background subtraction with frame difference approach
kalman.m	- Kalman filter calculation
addnoise.m	- add noise to the original video data
originalbb (dir)	- results of tracking the original object using background subtraction
noiseaddedbb (dir)	- results comparing the tracking when the noise was added vs. the original
justkalmanresult (dir)	- results of tracking with Kalman filter
kalmanwrandom (dir)	- results of tracking with Kalman filter vs. that from background subtraction with random noise added

Methods

We tried two types of background subtraction: 1) frame difference and 2) approximate of median frame difference. Frame difference is arguably the simplest form of background subtraction. The current frame is simply subtracted from the previous frame, and if the difference in pixel values for a given pixel is greater than a threshold, 'T', the pixel is considered part of the foreground.

$$|\text{frame}_i - \text{frame}_{i-1}| > T$$

Intuitively speaking, a major flaw of this method is that for objects with uniformly distributed intensity values (such as the object with uniform color), the interior pixels are interpreted as part of the background. Another problem is that objects must be continuously moving. If an object stays still for more than a frame period (1/fps), it becomes part of the background.

This method does have two major advantages. One obvious advantage is the modest computational cost. Another is that the background model is highly adaptive. Since the background is based solely on the previous frame, it can adapt to changes in the background faster than any other method. That also means that the frame difference method subtracts out extraneous background noise (such as moving trees and objects in the background), much better than the more complex methods such as approximate median and mixture of Gaussians methods.

A challenge with simple frame difference approach is determining the threshold value. We found our threshold--25--empirically through trial and error. If we set the threshold too high, the object is detected only partially in the best case. We will not provide the results from the frame difference due to space constraint of this assignment's the submission folder. However, we encourage the reader to run "frame_difference.m" and observe the results.

The main approach we relied on for this assignment is called "approximate median"[1]. In this approach, the previous N frames of video are buffered, and the background is calculated as the median of buffered frames. Then as in the case of frame difference, the background is subtracted from the current frame and thresholded to determine the foreground pixels.

Median filtering has been shown to be very robust and to have performance comparable to higher complexity methods [2, 3]. However, storing and processing many frames of video--as is often required to track slower moving objects--requires an often prohibitively large amount of memory. A more efficient alternative of median filtering works by approximating the median value of the buffer frames stepwise: if a pixel in the current frame has a value larger than the corresponding background pixel, the background pixel is incremented by 1. Similarly, if the current pixel is less than the background pixel, the background is decremented by one. In this way, the background eventually **converges** to an estimate where half the input pixels are greater than the background, and half are less than the background—approximately the median.

We shot multiple test videos in our apartment using two of the side walls as the background (see figure 1) and a soccer ball as the foreground object. We roll the ball across the room to make sure the movement is across both X- and Y- coordinates of the image frame. We took the videos under two different lighting conditions: under the room light and the natural light. Again, due to the file size constraint, we will only provide the results under the natural light, but we will discuss the differences we found in the two conditions in the next section. The video were shot at resolution (640x480), 30 frames per second. Because background subtraction algorithms typically process lower resolution grayscale video, we converted the video to grayscale.

For the final part of the assignment, we implemented the Kalman filter to track the object and compare the results with that of the foreground object. We decided to track two variables-- **the X- and Y-coordinates of the upper left corner of the bounding box**. We used the state prediction and update formula described in the class handout as below.

We assume that the object is moving at constant velocity our states, and that the states (x_i) and measurements (y_i) are normally distributed in pixel as follows.

$$x_i \sim N(d_i\, x_{i-1}, \sigma_{di})$$
$$y_i \sim N(m_i\, x_i, \sigma_{mi})$$

where we empirically obtained that $x_0^- = 622$px (X-coordinate with '0px' from the left margin) and $\sigma_0^- = 5$px (for the reason we will explain later).

Our prediction equation is:

$$\overline{x}_i^{\,-} = d_i \overline{x}_{i-1}^{+}$$

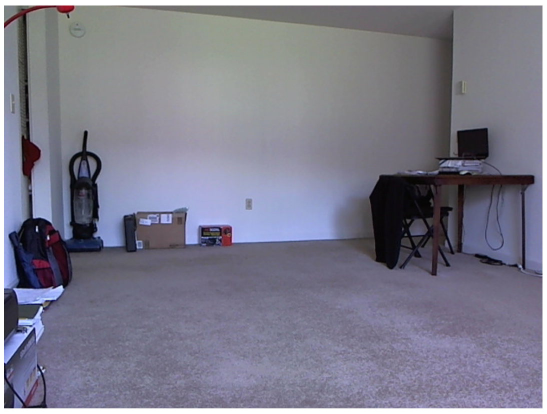
$$\sigma_i^- = \sqrt{\sigma_{di}^2 + (d_i \sigma_{i-1}^+)^2}$$

Our state correction equation is:

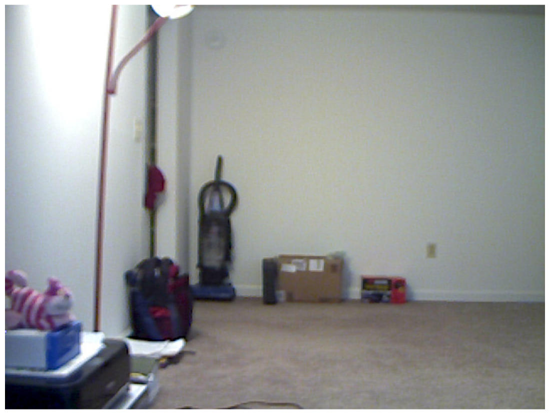
$$x_i^+ = \left(\frac{\overline{x}_i^{\,-} \sigma_{mi}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{mi}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_{mi}^2 (\sigma_i^-)^2}{(\sigma_{mi}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$

We calculate the X- and Y-coordinates--the predicted and corrected states--of the ball in separate steps using the above equations.



a)



b)

Figure 1. Background of the room taken under a) natural light and b) room light

Results and Discussion

A few sample frames from "approximate median" background subtraction are shown in figure 2. We noticed that the median filtering does a better job at separating the entire object from the background than simple frame difference approach (please run the Matlab code for the latter's results). This is because the more slowly adapting background incorporates a longer history of the visual scene, achieving about the same result as if we had buffered and processed *N* frames.

When we experimented the video taken with the camera held in our hands, the background subtraction does not work well. This is due to the fact that the background is no longer static and our approach assumes that the background is relatively static to the moving object in the foreground. We also observed that we need to adjust the threshold value, T, for different lighting condition.

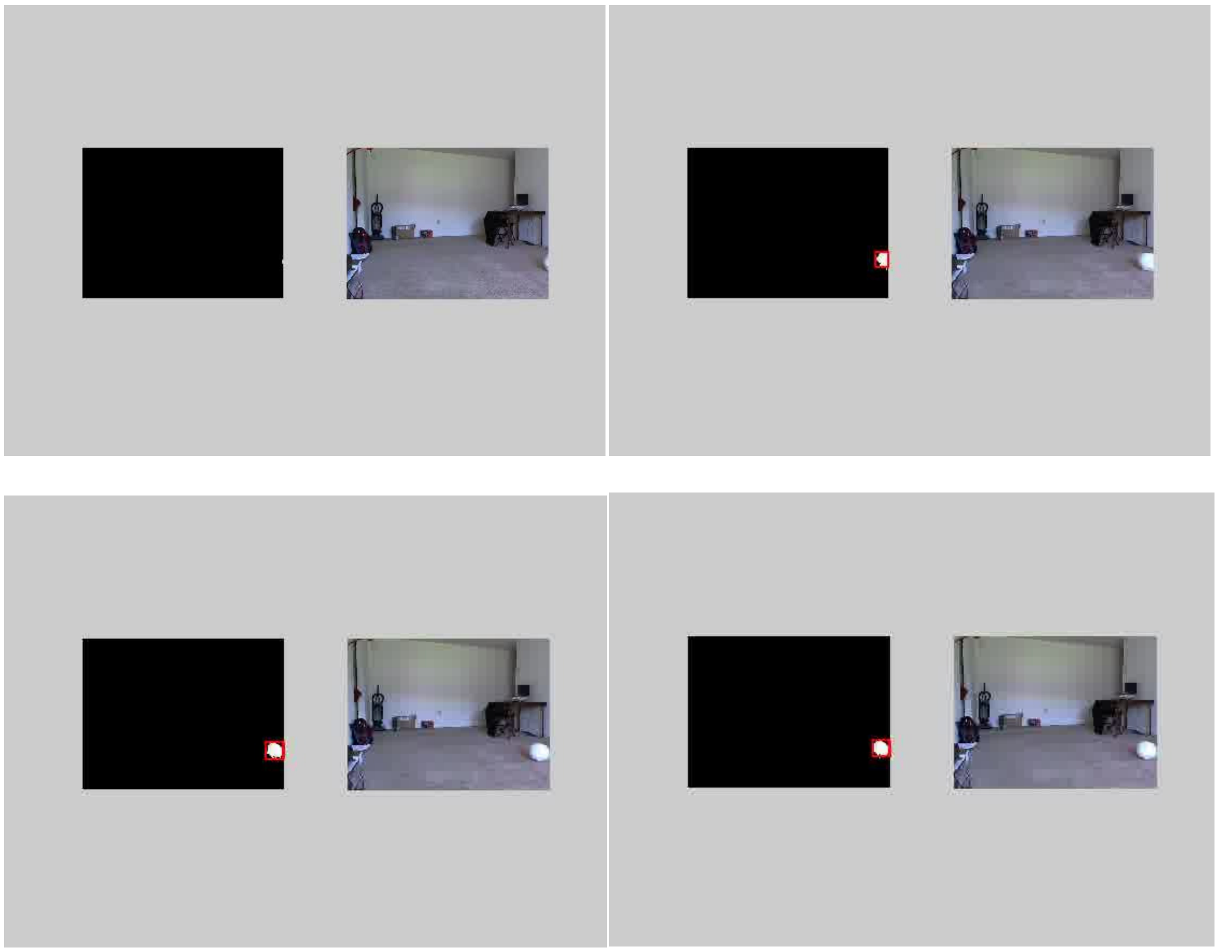


Figure 2. The first few sample frames of background subtraction with median approximation under natural room light. We observed that the ball is almost perfectly tracked by the algorithm.

Our initial attempts at background subtraction were hampered by the fact that the soccer ball has stripes on its surface, thereby causing its body to be divided into two to three separate entities. This complicated our effort to set the ground truth measurement values for Kalman filter tracking. As a result, we wrapped the ball in a sheet of paper towel to prevent the stripes from appearing in the video.

Once we have the background subtraction working, we took the entering position of the ball in the image frame for our initial starting position (622px and 337px for X- and Y-coordinates, respectively) as initial measurement values in Kalman tracking. However, since our background subtraction does a near perfect job of tracking the ball, we were not able to observe the noise in the tracking. In our first attempt to solve this, we added random Gaussian and salt and pepper noise to each image frame; however, this approach did not affect the tracking accuracy of the background subtraction significantly. Therefore, we decided to shift the top left corner of the bounding box result (from background subtraction) by a random value of normal distribution with 5px mean and 5px covariance. In other words, we shifted the bounding box by a random pixel values between ±5 pixels to artificially generate noise so that we can prove that Kalman filter can correct these anomalies/error in the measurement accordingly.

We have provided the results from original background subtraction (with approximate median filtering) in the folder named "/originalbb" and that with the noise added under "/noiseaddedbb". We observed that adding noise causes the bounding box to be jittery around the ball, and therefore, achieved the desired effect of noise being introduced to the system. In the folder titled "/kalmanwithrandom", we provide the comparison of results from Kalman tracking (with yellow bounding box) and that of background subtraction with noise added (with green bounding box). We observed that the Kalman tracking significantly dampened the noise we artificially introduced earlier. Finally, we provide just the Kalman filtering results in the folder named, "/justkalmanresult" and the bounding box color is kept yellow.

We also tried with significantly high noise values. For example, we shifted the bounding box's coordinates by as much as ±30px, and found that Kalman tracking withstood the noise level of up to 20px. After that threshold, the bounding box by Kalman filter seemed to follow the random movement of the noisy bounding box much more eagerly, and therefore, did not enclose the object entirely in its frame most of the time.

Conclusion

While the approximate median filtering has major flaws such as having to find the optimum threshold values, it does a better job of capturing the entire body of the foreground object than the simple frame difference approach. However, this significantly increased accuracy came with the price of more computation being necessary for approximating the median value. We also found that approximate median approach was robust enough to handle changing light levels thanks to taking the median of a few recent frames rather than depending on the most recent frame for background subtraction. As for the Kalman filter, we observed that it works reasonably well under limited noise level, provided that we model the behavior of the states correctly (we were lucky to have Professor Jiebo directing us which formula to use).

Appendix

[1] N. McFarlane, C. Schofield, “Segmentation and tracking of piglets in images”, British Machine Vision and Applications, pages 187-193, 1995.

[2] C. Kamath. Background subtraction for detection of moving objects August 8, 2005. April 2, 2012. URL: https://computation.llnl.gov/casc/sapphire/background/background.html

[3] S. Cheung and C. Kamath “Robust Techniques for Background Subtraction in Urban Traffic Video”. EURASIP Journal on Applied Signal Processing, Volume 14, pp 1-11, 2005. UCRL-JRNL-201916.