Vehicle Visual Tracking

Min Jae You, Young Woo Kim, Sangil Lee, Sam Gao You, Hyunjeong Tae

Introduction

Object tracking in computer vision is the process of developing unique identifications based on the initial set of identified objects and tracking them as they move throughout the frames of a video. The goal for our project was to utilize this technology to analyze a video file provided as an input to accurately identify and track automobiles that are in movement. For the purpose of this project, we decided to apply this technology, specifically with the Kalman filter.

The Kalman filter is used to learn the motion of the selected object, in our case, automobiles, assuming a Gaussian model. As more filters are passed, we accumulate more filters and more reliable predictions using the Kalman filter.

The main motivation behind choosing this specific project is because of its potential application to intelligent traffic management and efficient infrastructural improvement. With a highly accurate vehicle tracking system, one can collect data and analyze them to answer questions such as the ones listed below.

- How many vehicles are present on a given day?
- Which highway should be expanded to reduce traffic congestion?
- Which types of vehicles pass through a given highway?
- How many car accidents occur on a specific highway and at which time of the day do they most frequently occur?

Answering these questions will not only reduce traffic congestion and number of accidents, but also increase the efficiency in urban planning, thus reducing the monetary costs and increasing citizen satisfaction.

The task at hand, as mentioned above, is quite simple. We will segment a video into a series of frames and treat each frame as an independent image and apply a Kalman filter to predict the movement of the object we're tracking. We decided that we should be able to complete this task, because throughout the EECS 442 course, we have applied different types of filters on 2D images in our homework assignments. Since we already have successfully completed such tasks in the past, we decided that we should be capable enough to perform a similar task on the video segments. The use of computer vision is vital to this project that we chose to work on, because we are trying to automate tasks that require a human visual system. If we don't use computer vision, then we must acquire human labor that would have to continuously watch each video filmed by countless number of security cameras on the roads, which are recording twenty-four hours a day. This would be a great waste of labor, as the software that we are working on would immensely shorten the amount of time and reduce resource costs.

One of the related pieces of work includes a report *Object Tracking in a Video Sequence* by Young Min Kim. His project was very similar to what our project wanted to accomplish. The program that he wrote tracks objects that are randomly chosen by a user using scale-invariant feature transform (SIFT) and Kalman filter. Although we didn't utilize SIFT in our project, this report had a detailed explanation of why and how the Kalman filter works along with diagrams and equations that were straightforward to understand, thus giving us a strong mathematical understanding of this tool before we started implementing our project. On a similar note, we also watched a video by Pysource about the application of the Kalman filter to predict the trajectory of an object in motion. Contrary to the report compiled by Young Min Kim, this video provided multiple demonstrations of the Kalman filter at work in Python and focused on the employment of this technique in code. By providing example codes, we were able to use it as a foundation and a template to get us started on the project which seemed daunting at first.

In summary, our goal was to track a specific vehicle that's been chosen by a user. In order to accomplish this, we created a 2 dimensional box that contains the object we're trying to track. For every frame after the initial frame, we shifted this box around and applied the objective function to find the box that most accurately contained the object so that we can use the Kalman filter to make predictions about the object's trajectory.

Approach

Object tracking is done by utilizing the Kalman filter that assumes a Gaussian distribution. We make the assumption that the object moves in near constant motion over each frame. The Kalman filter will allow measurements to be taken over a period of time and use those measurements to make a prediction of the next movement.

For the algorithm of object tracking, the first step is to be able to identify the initial location of the object, in our case the automobile. We pre-processed our data, video files of vehicles, to find the initial coordinates of the vehicle we want to track. The program receives as input the name of the video file, with the title of the format x_y_x1_y1.mp4, where (x, y) is the initial top left corner coordinates of the bounding box, and (x1, y1) is the initial bottom right corner of the bounding box. For each iteration after the initial frame, before applying the Kalman filter, the position of the vehicle must be found in the new frame. This new position is then given to the Kalman filter to predict the location of the bounding box. At first, since there will not be a lot of data given to the filter, the initial few bounding box predictions will not be very accurate, but as the video goes on and more data becomes available, the prediction is expected to improve.

The new position for the vehicle in the new frame is found by shifting the previous bounding box in the new frame and comparing it with the bounding box in the old frame. The magnitude of these shifts are controlled by a hyper parameter gamma, which we set by multiple experiments to find the best bounding box. Then, by computing the normalized cross correlation of these shifted boxes with the old bounding box, we find the point with the largest correlation, which represents the location of the vehicle in the new frame.

Following every iteration of the algorithm, the model will become better and better at making more reliable predictions. As it receives more input, the Kalman filter will gradually learn the pattern in the path of the vehicle and make better predictions. This means that the Kalman filter makes better predictions throughout the course of the video.

Figure 1 depicts the predictions of the Kalman filter in the first couple frames of one of the videos. The first prediction of the filter is completely off as the model has only received a single input coordinate. Therefore, the filter cannot yet accurately predict the movement of the vehicle. However, over the next couple frames, the Kalman filter adjusts its predictions. Initially, the filter predicts a downwards movement of the object. Then, it is able to recognize the vehicle is moving forward so the other object (another vehicle) also looks as though it is moving forward.

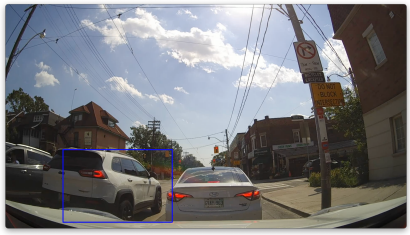
Figure 1 (below) Screenshot of first couple frames The bounding boxes in blue show the prediction of the Kalman filter.

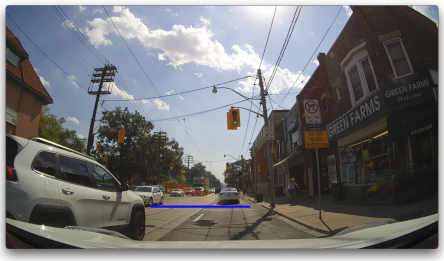
Experiments

The data for our experiments came from the Pedestrian Intention Estimation dataset. We took the videos of vehicles, where it appears to have been taken from a dash cam, and pre-processed the ones we used to find the initial coordinates of the bounding box for the vehicle that we wanted to track. We titled the video files in the format x_y_x1_y1.mp4, where (x, y) is the top left coordinate of the initial bounding box, and (x1, y1) is the bottom right coordinate of the initial bounding box. This data that we used makes sense since we wanted to build an application to track the movement of vehicles in traffic.

We measured the success of the tracking program qualitatively, by seeing for ourselves throughout the output video whether the bounding box was in the right position and correctly tracking the vehicle that we wanted to track. Initially, we found that our program was not able to predict the bounding box very well. A common error that occurred was that throughout the video, the size of the bounding box would get smaller and smaller, as seen in figure 2. To get around this, we set the width and the height of the bounding box to be consistent, so that it would be able to stay around the vehicle.

Figure 2 (below)
The bounding boxes in blue changing sizes





Another issue that we ran into was the program not being able to properly find the coordinates of the vehicle in the new frame. Since the Kalman filter depends on using the coordinates of the vehicle to predict the next location, this caused our program to incorrectly track the vehicle.

Figure 3 (below) Bounding box is not in correct position

Since we find the coordinates of our vehicle in a new frame by taking the bounding box of our old frame, shifting it a certain percentage left and right, up and down, and selecting the box that has the highest normalized cross-correlation with the old frame, we set the percentage that the box would be shifted as a command line flag gamma. By setting gamma to be variable, we were then able to designate just how much around the box our program should “look at” to determine where the vehicle’s new location was. By setting gamma to be a higher value, our program is then able to look in a larger area for the vehicle and find more accurate coordinates of it in the new frame.

To produce a quantitative metric of our algorithm’s performance, we drew near-perfect hand-drawn bounding boxes around the images, and compared their percentile differences in area and centroid (x, y) against the algorithm’s bounding box. Below, the hand-drawn bounding box is in red and the algorithm’s in blue.

Here is a table with results from the first 10 frames of this example:

Figure 4

Frame (Metrics calculated Algo / Hand)	% difference, area	% difference, centroid
1	1.34	0.24
2	1.92	0.89
3	2.57	2.78
4	4.57	4.23
5	7.89	6.12
6	10.28	8.67
7	14.27	10.23
8	19.27	12.45
9	23.81	15.23
10	25.98	17.67

As you can see, the algorithm is less effective in predicting area than centroid, owing to the fact that cars usually speed up or slow down. This difference in precision can also be attributed to the fact that we imposed a fixed-sized constraint on the bounding box so that we would only have to search over 2 degrees of freedom (centroid coordinates), as opposed to 4 degrees of freedom (centroid, width, and height), which would have increased time complexity beyond the practical scope of the project. In this example the white SUV is driving away from the camera and therefore its true bounding box should decrease in size. Typically, when driving on the street, cars move either from or away the observer, so their bounding box size is not frame-invariant. Another issue with recognizing the best next frame, and that explain the accumulation of imprecision in our algorithm, are noise artifacts and subtle changes like in coloring, pedestrians, and occlusions, that come into the frame and which accumulate through time. A solution to mitigate this problem for the future would be to apply some kind of image-mask that touches up these occlusions and subtle changes, for example a binary mask, and then putting the binary mask into the standard normalized cross correlation finder we have built. We decided, however, not to do this because we wanted to leverage the rich RGB channeled data inherent in the original video and not lose out on identifying features like headlights, color, license plates, etc. However this came at the cost of time complexity because multi-channeled data takes longer to compute and results in accumulated noise artifacts that compound in precision over time, as the data above reflects.

Implementation

The skeleton of our code was largely influenced by the Pysource video by Sergio Canu on the application of the Kalman filter to predict the trajectory of an object, specifically a ball, in motion. With this, we modified the code to be suitable for vehicles by changing the way we predict the location of the vehicle in each frame. From the Pysource video, we were inspired to use the OpenCV Kalman filter class. This allowed us to focus on the algorithm rather than implement every function from scratch. Additionally, we modified certain variables such as input coordinates, gamma (used to shift the bounding box in the next frame), format of the input, etc. such that it was more suitable for our project. To summarize our pipeline, we took the centroid of the bounding box from our formula, and passed that into the OpenCV Kalman filter to predict the next location of the centroid, which adding onto the fixed-bounding box size assumption allowed us to draw future bounding boxes. With respect to code, we implemented a zero-meaned cross correlation, which we then fed into a double for loop to loop over the boxes within an area difference gamma from our existing frame, because cars do not suddenly jump huge distances from frame to frame, especially in suburban settings.

** 4 pages not including References

References

[1] Kim, Young Min. pp. 1–5, O*bject Tracking in a Video Sequence* .

[2] Canu, Sergio. “Kalman Filter, Predict the Trajectory of an Object.” *Pysource*, 2 Nov. 2021.

[3] Rasouli, Amir and Kotseruba, Iuliia and Kunic, Toni and Tsotsos, John K., “PIE: A Large-Scale Dataset and Models for Pedestrian Intention Estimation and Trajectory Prediction”, 2019