

CSE 576
Project 3
Tracking Moving Vehicles
June.6.2011
Ryder Ziola

I. Introduction

The goal of this project is to locate and track moving vehicles in a video. There are many useful applications for such a system, from data collection for traffic planning or tolling, to security and surveillance. One potential application I hoped to explore was wildlife tracking: counting birds flying past the window.

Several assumptions were made to help define the scope of the project:

- Any moving object should be identified, not just vehicles. Identifying the motion is the interesting part of this, not detecting any vehicle-specific properties.
- Videos can contain multiple vehicles, moving in similar or different directions. They can also contain no vehicles.
- The camera can be moving as well, in any direction.
- Vehicles should be identified only if they are moving. This is not designed to be a general-purpose vehicle detector

II. Related Work

My solution leverages Lukas & Kanade's optical flow algorithm. It uses features that are generated with a Harris detector.

III. Method

My system was written in Python and uses OpenCV.

The basic pipeline involves passing sequentially through the video and applying the following operations:

1. Frames from two subsequent time steps are loaded: $\text{time}=t$ and $\text{time}=t+N$ [See  in appendix]

The best value for N seems to vary depending on the video, likely as a function of the FPS of the video and the speed of the motion of the object. In some cases, the best results came from comparing consecutive frames. ($N=1$) On others, $N=5$ or $N=7$ would give the best results. This value could probably be determined online by looking at a statistic such as the overall speed of motion. All results shown here were with a constant value of $N=3$, to avoid any file-specific intervention on my part.

2. Both images are converted to single-channel images.

3. The images' histograms are equalized to ensure there are as many detectable corners as possible. [See img2 in appendix]

4. A Harris detector is run on the first image (time=t) to find interesting points to track with optical flow. [See img3 in appendix]

[comment: i also tried using a naive grid of points as input to the optical flow algorithm. This avoids the problem of sparsity of points in untextured regions, letting us trust the Lukas-Kanade algorithm to interpolate where we don't have data. This didn't work out so well in practice <why not?>

I limited the detector to finding the best 500 points, but in practice it would usually only find 100-300 points above the quality threshold I set.

5. The optical flow is computed for the Harris-detected points, to find corresponding points in the second image, defining a set of motion vectors. [See img4 in appendix]

6. The resulting vectors are clustered using k-means, with $k=2$. [See img5 in appendix]

This roughly separates the background/camera-motion vectors from the moving objects. The larger cluster is designated as representing the background motion. The centroid of the background cluster is used to capture the direction of the background motion (background motion vector).

Despite the simplicity of this approach, in practice, this seems to result in the larger cluster being a good approximation of the background motion even when the objects are moving in different directions.

I attempted several more elaborate approaches to determining the background points (partially discussed below in Future Work), but none of them worked as well as this simple one on the basic cases.

In contrast with step 8 below, the vectors being clustered here are represented by their (dx,dy) , considering only their magnitude and angle when building the clusters.

7. The background motion vector is subtracted from all of the optical flow vectors, correcting for the camera motion. [See img6 in appendix] The result should approximate the motion of the actual objects in real-world space. The resulting vectors are thresholded fairly aggressively to zero out the small remainders of the averaging process in the background cluster. The ability to threshold in this step accounts for much of our ability to tolerate inaccuracies in the previous step. Outlier vectors which are not surrounded by other nearby vectors are discarded, under the assumption that a real object would generate multiple points and they are likely spurious.

8. K-means is run multiple times on the corrected vectors, with a wide range of values for k. [See img7 in appendix] The resulting clustering are compared statistically. I look at

the absolute differences in centroids, and variances of the clusters to determine the best clustering. In contrast to the previous clustering in step 6, the vectors here are represented by their scaled (x,y) location in addition to their angle and magnitude. This is done to capture some sense of locality in the clusterings. See `find_best_cluster()` for the implementation.

9. The winning clusters are correlated with the boxes from past iterations. [See `img8` in appendix] This is done by looking for the correspondences between a new bounding box and an old bounding box (offset by its previous motion vector) that result in the largest overlap as percentage of their combined area. See `correlate_with_old_boxes()` for implementation. This allows continuity in the labels being applied between frames.

I tried some more complex modeling of the expected locations of each bounding box, in an attempt to provide better correlations between frames. This didn't work very well. The motion vector for a cluster in the occasional frame would sometimes be very erratic, resulting in some wildly inaccurate predictions. Better smoothing and prediction might avoid this problem and allow the use of this sort of technique, but I found the naïve approach described here robust enough.

IV. Results

Please refer to the video at <http://www.youtube.com/watch?v=enESWj6N5wA> for supporting visuals.

“Army Truck in Desert” is a good example of the expected behavior with a single vehicle. The general location of the vehicle can be discerned, but the bounding box varies greatly depending on the specific points detected in that image.

“Car Passing Truck” shows the best results observed. The background motion is constant and uniform. When it is accounted for, the two vehicles can be differentiated quite easily for the full duration of the video, despite overlapping in space. The changing size of the truck's bounding box results from an interesting problem with my approach: untextured regions (the white of the truck) don't provide any corners to track and we have few points to track within it, resulting in more variation.

“Person Walking”, despite not being a vehicle, works fairly well. An interesting failure occurs when the person walks behind the street lamp. Despite attempt to smooth the labels over a large window (almost 10 frames!), the person is issued a new label when they re-emerge from behind the pole. Better modeling of motion, discussed under step 9 above could address this.

The most common point of failure in the more difficult videos is bad background modeling. **“Tank in Desert”** demonstrates this well. Regions of background are

identified as vehicles incorrectly. This highlights a fundamental obstacle my approach faces: the background motion is modeled as a single two-dimensional vector. As a result, it can account only for simple linear translations between frames. In contrast, rotations and perspective skewing can appear quite easily in any realistic camera motion. I captured **“Rotating Building”** to illustrate this effect. The scene is all background throughout, but the conflicting translations that result from the camera rotating causes an arbitrary area to be identified as the background and the rest of the vectors to be clustered as foreground. To avoid this problem, the background motion will have to be modeled with a higher dimensionality, as discussed in future work.

V. Future Work

The two main shortcomings of the current design are the simplistic clustering (step 8) and the linear background motion model.

There is a lot of room to improve the clustering by taking historical information into consideration. Currently, information on the clusters found on past iterations is not incorporated into the pipeline until step 9, correlating and smoothing bounding boxes. I would like to explore approaches that involve using previous bounding boxes to help with the clustering in step 8. This would require better modeling of the motion vectors of each bounding box to predict the likely new location, probably with a Hidden Markov Model. If we can project past boxes into the present frame, this information can be used to evaluate the quality of the proposed clusterings.

Finally, modeling background motion as a simple translation is clearly not robust enough. The approach I would like to try would involve modeling it as homography to capture rotation and skew. This could be accomplished by selecting points, building a homography with them, and calculating the number of inliers and outliers it produces among the full set of points. Given the large number of points we are using (usually 100-300) and the high ratio of background points, the homography could be built from randomly selected points with a high likelihood of success. If we built several random homographies on each frame, compared them, and picked the best, it would likely capture the background sufficiently well. I began some work in this direction, but didn't have time to develop it far enough for it to work well enough. Still, this approach of building a homography would not be able to represent parallax or other motion artifacts. Tackling more complex motion would be a challenge.

VI. Conclusion

In conclusion, it is surprising how well the fairly simple techniques employed on a certain class of videos. At the same time, it is clear that the problems that were encountered require increased sophistication, as opposed to merely increased

refinement, to tackle them. The general structure of my pipeline seems fairly robust, though. There is definitely still room for substituting in better components to perform each subtask (background subtraction, clustering, etc) without encountering the limits of the pipeline.

VII. Appendix Images

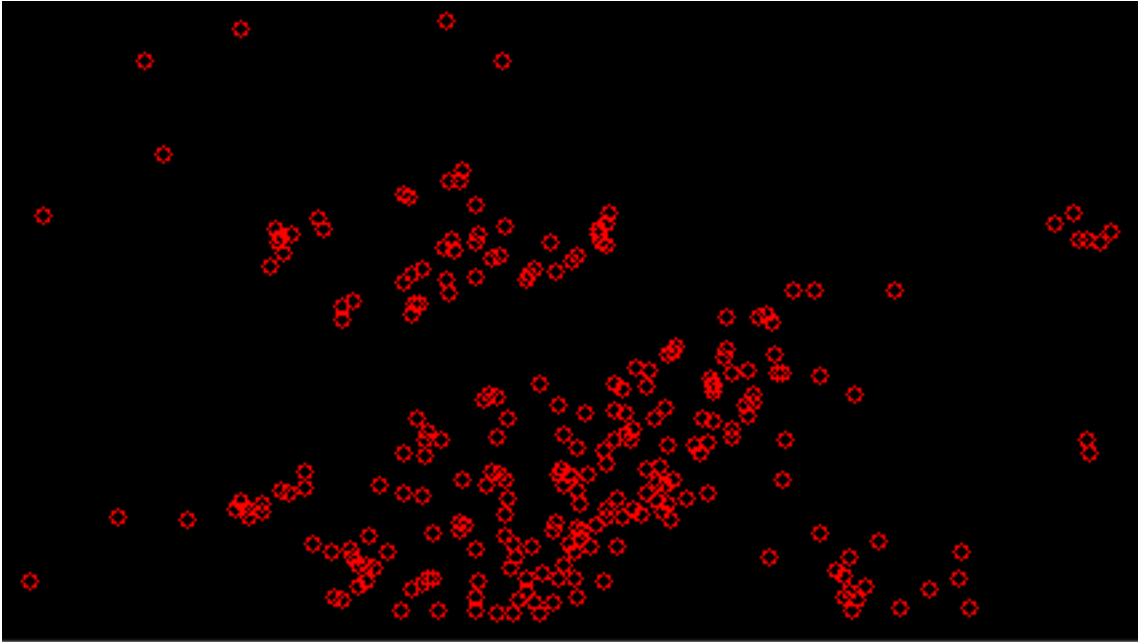
img1



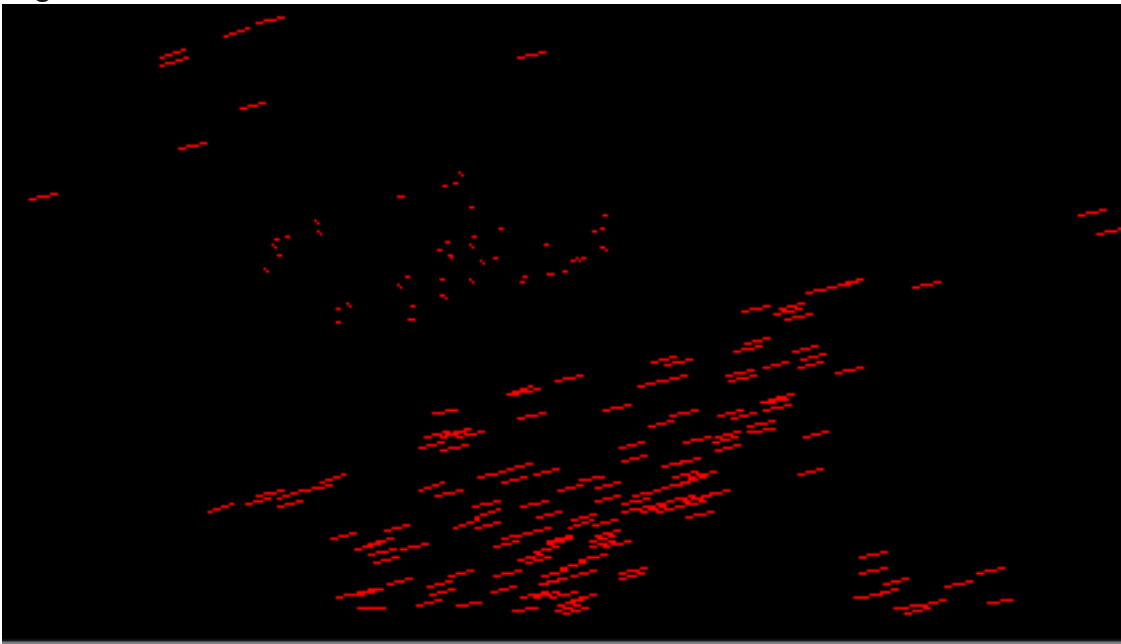
img2



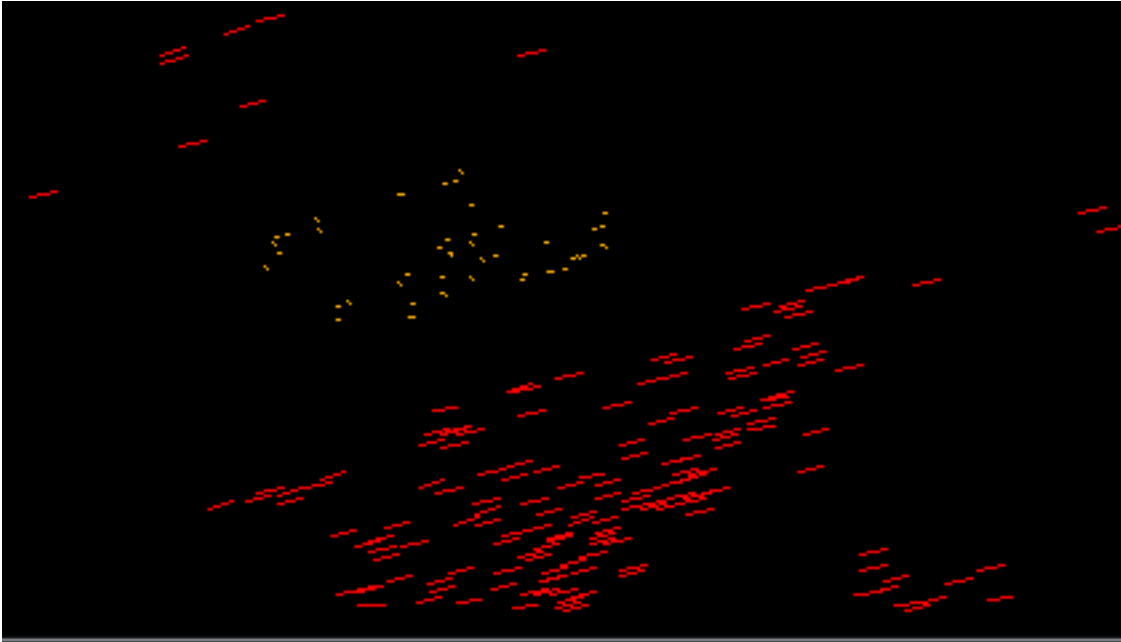
img3



img4



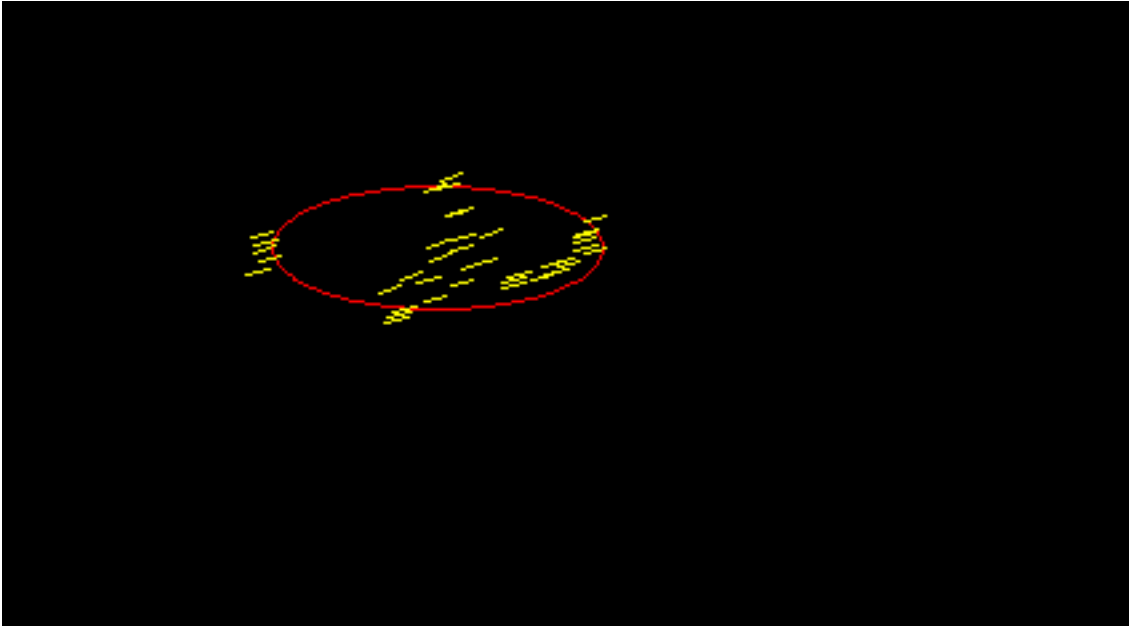
img5



img6



img7



img8

