# STEREO IMAGE BASED COLLISION AVOIDANCE

Amrit Lal Singh

***Abstract*— Optical flow has wide application**

**What is done here is to use the consecutive frames to generate the optical flow in the video. We decide the good points to track and then we track them via Lucas-Kanade method. It is involved in the optical mouse, just on a much lower resolution but much higher frame rate. We could estimate the speed of vehicles on highway. We could use it for image stabilization, taking ground to be fixed and assuming all motion is of the camera. We could use optical flow to increase frame rate, as we could guess where the pixel will be in a frame that is in between those frames and predict the frame that was never captured.**

**We could use optical flow to speed up object recognition in a filmed video as now we do not need to run the object recognition on every frame, we can just run it on one frame and then track the object further, this speed up the process multifold.**

**In case of aerial robotics we could use it to detect if an object is stationary with respect to the ariel vehicle or if we do this with the ground we could detect if the vehicle is stable or not.**

## I. INTRODUCTION

By combining information from several nearby pixels, the Lucas–Kanade method can often resolve the inherent ambiguity of the optical flow equation. It is also less sensitive to image noise than point-wise methods. On the other hand, since it is a purely local method, it cannot provide flow information in the interior of uniform regions of the image.

Initial approach was to see the nearby 8 pixels if this pixel has moved but this approach failed as the pixel could more than one pixel or less than one pixel. Then the realization struct that the problem of optical flow is an under constrained problem.

## II. PROBLEM STATEMENT

**Optical flow is the motion of objects between consecutive frames of sequence, caused by the relative movement between the object and camera.**

Optical flow is the motion of objects between consecutive frames of sequence, caused by the relative movement between the object and camera. You can read up more on Optical Flow here. We can represent the motion of an object between frames as a single vector located at the coordinates that best predicts the change in intensities (between frames) over that window from the image gradients. One of the most popular methods to compute the Optical Flow Field is the Hierarchical Lucas-Kanade (henceforth referred to as LK) method. You will have to implement, from scratch, a function that detects the apparent motion between frames by computing the LK flow fields. Your implementation should be optimised enough to run on super-slo-mo videos. For increasing the accuracy of your Optical Flow model you can introduce higher derivatives too (You will be awarded extra points for doing that). A few sample image sequences and videos for trial have been provided here. If you're interested, you can try to implement RAFT as well.

*The internet and to it's seeming infinite store of knowledge. The gods of stack overflow

Fig. 1. Optical Flow

## III. RELATED WORK

By combining information from several nearby pixels, the Lucas–Kanade method can often resolve the inherent ambiguity of the optical flow equation. It is also less sensitive to image noise than point-wise methods. On the other hand, since it is a purely local method, it cannot provide flow information in the interior of uniform regions of the image. RAFT, aka Recurrent All-Pairs Field Transforms, is a deep learning method to solve optical flow iteratively.

Fig. 2. RAFT Method

## IV. INITIAL ATTEMPTS

Initial approach was to see the nearby 8 pixels if this pixel has moved but this approach failed as the pixel could more than one pixel or less than one pixel. Then the realization struct that the problem of optical flow is an under constrained problem.

## V. Final Approach

**This should cover both columns ( full page ) excluding images**

Write about the final algorithm used or developed which was able to finally solve the problem statement. Write all the assumptions and equations properly with images and result tables / comparison.

If report is about implementation, write all the steps of implementing the report with step-wise syntax and details about the errors faced and their solution.

The least-squares approach implicitly assumes that the errors in the image data have a Gaussian distribution with zero mean. If one expects the window to contain a certain percentage of ”outliers” (grossly wrong data values, that do not follow the ”ordinary” Gaussian error distribution), one may use statistical analysis to detect them, and reduce their weight accordingly.

The Lucas–Kanade method per se can be used only when the image flow vector Vx, Vy Vx,Vy between the two frames is small enough for the differential equation of the optical flow to hold, which is often less than the pixel spacing. When the flow vector may exceed this limit, such as in stereo matching or warped document registration, the Lucas–Kanade method may still be used to refine some coarse estimate of the same, obtained by other means; for example, by extrapolating the flow vectors computed for previous frames, or by running the Lucas-Kanade algorithm on reduced-scale versions of the images. Indeed, the latter method is the basis of the popular Kanade-Lucas-Tomasi (KLT) feature matching algorithm.

A similar technique can be used to compute differential affine deformations of the image contents. The least-squares approach implicitly assumes that the errors in the image data have a Gaussian distribution with zero mean. If one expects the window to contain a certain percentage of ”outliers” (grossly wrong data values, that do not follow the ”ordinary” Gaussian error distribution), one may use statistical analysis to detect them, and reduce their weight accordingly.

In the code we import matplotlib, cv2, pyplot and then we capture the video feed into the cap variable. Now we need this and the previous frame to calculate the optical flow we greyscale the image and also downsize it. Then we call our self defined function LK opticalflow and give it the two frames. Now we convert the two greyscale images to np arrays and store their shape in s. then we apply gaussian filter with a kernel size of 3. Then

Then we use features = cv2.goodFeaturesToTrack() to find good features to track in the image. as we need vertices or good textures in the picture as if edges or surfaces without texture present then the problem is badly constrained. Then we perform a pseudo inversion and we find the vectors. Then we draw the vectors using

## VI. Results and Observation

Compare your results with all the available algorithms which you may have used to tackle the PS. If possible, present your and their results in tabular / graphical format.

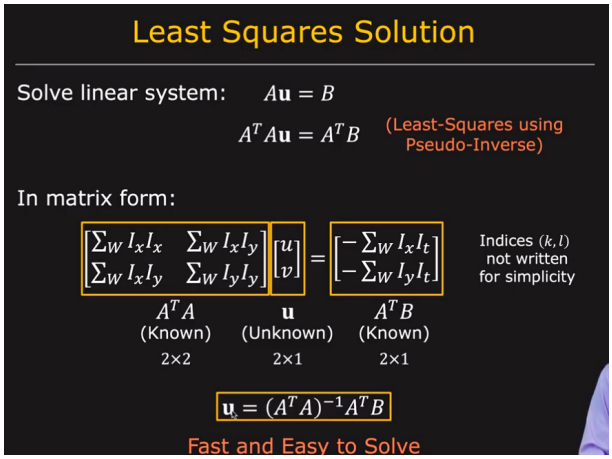


Fig. 3. Math

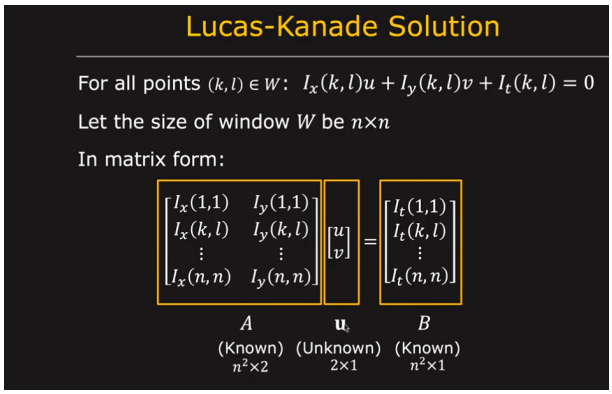


Fig. 4. Math 2

The implementation based on LK method works as expected although coarse to fine estimation would have given better frame rates. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow could also have been implemented and would have given better result also it’s code is available, but would have taken some time to understand and hence was not attempted.

as for this implementation, the outputs are perfect.

## VII. Future Work

The algorithm works perfectly and gives good output, but another method involving grouping very large groups of pixel together and then calculating their flow then dividing them into two parts and then computing sort of method(Coarse-to-fine estimation) seems much faster and can be implemented. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow can also be implemented it seems it’s code was also made available by its authors, but understanding it would take time.

## CONCLUSION

The problem was to implement optical flow from scratch, and the result was successful. The implementation worked according to the theory and processed image as well as super-slo-mo-video with expected speed. The L-K based optical

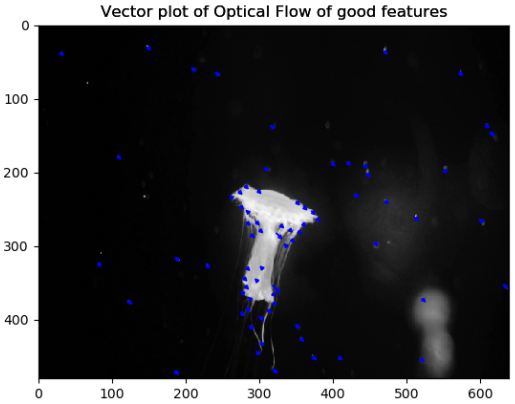


Fig. 5. The Jellyfish and the camera both moving

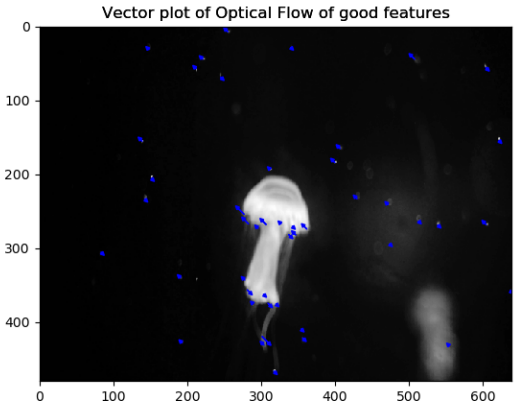


Fig. 6. The Jellyfish and the camera both moving

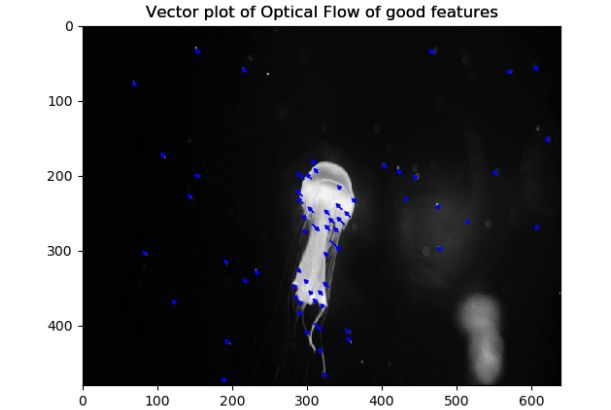


Fig. 7. The Jellyfish and the camera both moving

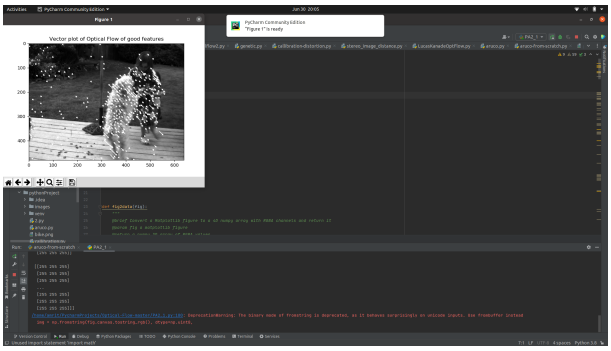

Fig. 8. The dog frames

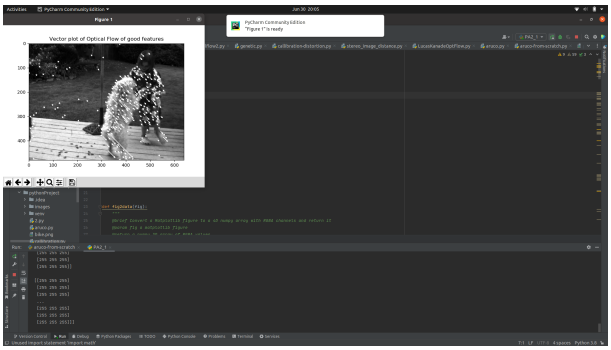

Fig. 9. The dog frame

flow is very useful and has various applications in aerial robotics and is a great tool that once included in a system the ability of the system to 'perceive' is much enhanced. It could be used to perceive the surrounding and other objects. Can be used to stabilize video feed and the aerial vehicle itself.

## REFERENCES

[1] First Principles of Computer Vision, Shree K Nayar, Columbia University