Optical Flow

Lenart Rupnik, 63220472

I. Introduction

Optical flow is defined as the apparent motion of individual pixels on the image plane. It often serves as a good approximation of the true physical motion projected onto the image plane. To calculate optical flow, we use two consecutive images which are taken in time t and $t+\Delta t$. Most optical-flow methods are based on computing estimates of the motion of the image intensities and assume that the colour/intensity of a pixel is invariant under the displacement and that the motions are minimal. Methods presented in this report are Lucas-Kanade and Horn-Shunck.

II. Experiments

Firstly, we created a basic implementation of both methods and compared results on the rotated random noise image. Optical flow generated with Lucas-Kanade method can be seen in Fig. 1 and Horn-Schunck flow can be seen in Fig. 2. Since our rotation was done for small differences, both methods performed well, with Horn-Schunck method getting slightly better results.

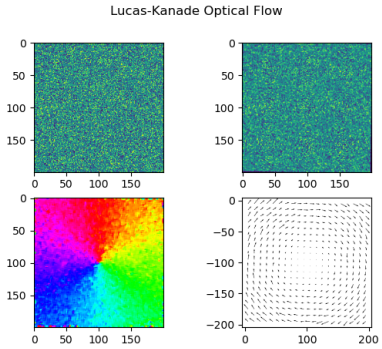

Figure 1: Optical flow of random noise rotation with Lucas-Kanade.

After that, we tested our methods on three additional pairs of images. Images are represented in Fig. 3, 4 and 6. For the first image represented in Fig. 3 we used a ship on the ocean that moves slowly. Both methods manage to capture general optical flow of our ship and waves, probably because there are

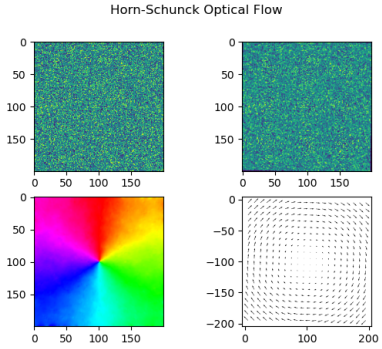

Figure 2: Optical flow of random noise rotation with Horn-Schunck.

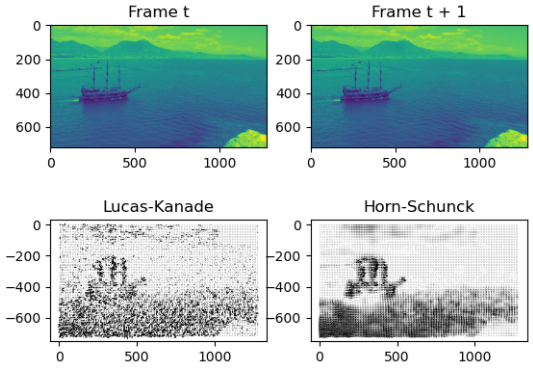

Figure 3: Optical flow of a moving ship.

mostly small motions. Focusing on flow density, Horn-Schunck method gave slightly better results. Second image, Fig. 4, represents traffic movement in one cross-section. The camera for these images is placed on the drone. In this case, Lucas-Kanade method has some problems with estimating optical flow. Small movements of the whole image are well represented, but movements of certain cars aren't perfectly captured. All together we see that image is still well represent optical flow. I would assume that some mistakes in optical flow come from breaking small displacement's assumption. On the other hand, Horn-Schunck provided better general optical flow of moving camera while also estimating general flow of moving cars.

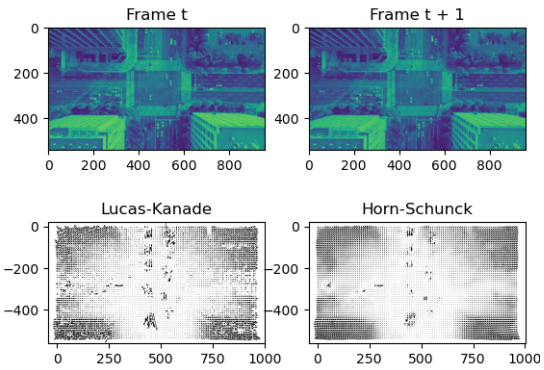

Figure 4: Optical flow of traffic.

In the third image, Fig. 6, we tried to find optical flow for the automated robot line. Similar to pervious example, finding optical flow with Lucas-Kanade wasn't perfect, probably due to violation of small displacement assumption. Even optical flow with Horn-Schunck wasn't perfect in this example, although general movement of robot hand and production line is clearer.

One of the general assumptions when using Lucas-Kanade method is a small motion assumption. If we violate it, flow vectors are out of proportion, which can be seen in Fig. 6. We can improve this by calculating reliability of optical flow in each pixel and ignore the pixels where we breach small motion assumption. This can be done by calculating eigenvalues of

pixel gradient and looking at their sizes and relations between horizontal and vertical direction. Ideally, both aren't small numbers and are somewhat equal. To shorten the calculations of eigenvalues, we can use the response from Harris corner detector.

We used proposed improvement on our images, but our optical flows didn't improve. In all images, the representation of a corrected flow was deformed, with flow vectors being enlarged and out of image scope. Although images weren't clear, we saw that the remaining vectors were chosen correctly, so apparently our edge detector was working. We figured out that there is probably some bug when scaling images in *showflow* function but we couldn't resolve it. After more testing, we found that we can show improvements on already given images. We plotted the results in Fig. 5.

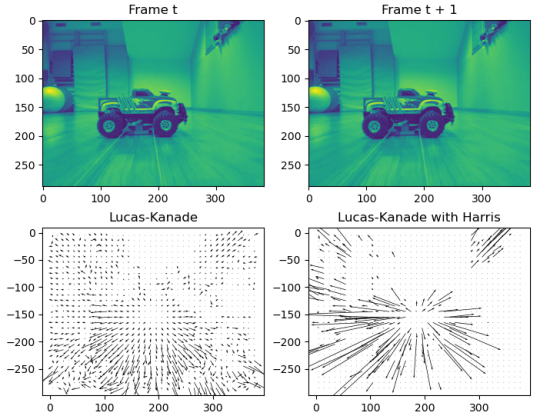

Figure 5: Lucas-Kanade Harris improvement.

In general, there are three parameters that can be used for tunning Lucas-Kanade and Horn-Schunck. For Lucas-Kanade we can change kernel size used in convolution. This way we can manipulate the size of the neighbourhood, removing local spikes. Results of this comparison can be seen in Fig. 7. In our case, the best parameter would be a kernel of a size around 10. For Horn-Schunck method, we can change two parameters. One is the number of iterations and the other is lambda, which is used to smooth the flow. We tested both parameters with 3 different settings. Results are seen in Fig. 9 and Fig. 8. With more iterations we get smoother results, which is similar when we increase lambda. However, we still need to be careful not to over-smooth the image because then it's not recognizable any more. So in cases where movements are short and clearer, we can decrease the number of iteration and smoothness level.

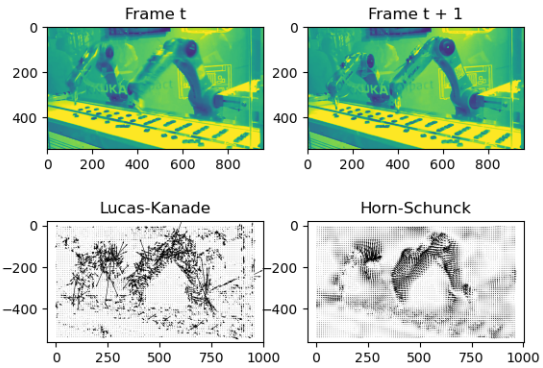

Figure 6: Optical flow of an automated robot line.

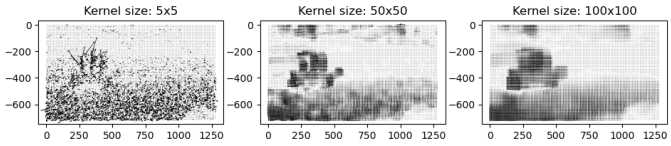

Figure 7: Lucas-Kanade kernel size comparison.

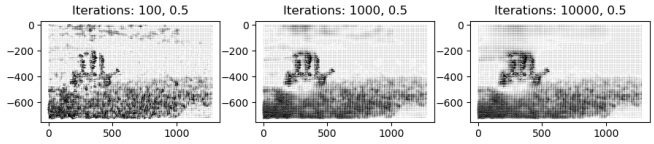

Figure 8: Horn-Schunck iteration parameter comparison.

Based on our results we would suggest to set iterations around 1000 and lambda around 1.

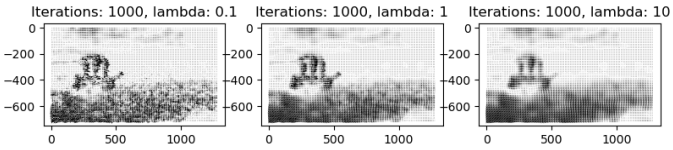

Figure 9: Horn-Schunck lambda parameter comparison.

Up to this point, we can see better performance with Horn-Schunck. To give comparison between methods another dimension, we measured execution times, which can be seen in Table I. As we see, Lucas-Kanade is faster than Horn-Schunck in all tests. Horn-Schunck high time complexity is a consequence of having to go through multiple iterations until we converge.

Image	Lucas-Kanade (kernel size)	Horn-Scunck (iterations)
ship	0.14 s (5)	3.23 s (100)
traffic	0.26 s (5)	5.63 s (100)
robot	0.22 s (5)	4.56 s (100)

Table I: Execution times for both methods with multiple images.

To tackle this problem, we can use additional improvement by initializing U and V values with Lucas-Kanade. That way, we don't start with an empty matrix, but we already have some information about our optical flow, which results in iterations converging faster. Improved times are shown in Table II.

	Horn-Schunck	Horn-Schunck initialized with Lucas-Kanade
ship	55.859 s	22.125 s
robot	121.075 s	55.687 s
traffic	214.95 s	9.109 s

Table II: Improved Horn-Schunck execution times.

III. Conclusion

Both implemented methods gave valid results. Lucas-Kanade is more time efficient, while Horn-Schunck can produce clearer results. We showed that Lucas-Kanade's results can be improved with Harris response and Horn-Schunck's time inefficiency with Lucas-Kanade initialization. Since both methods depend on a small motion and same pixel brightness assumption, they probably wouldn't give the best results for real life implementation.