Assignment 1 report - optical flow methods

Luka Boljević

I. Introduction

Given a sequence of images, optical flow is the velocity field of an image, which shows the apparent motion connecting one image to the next one in the sequence. Basically, it shows us how all pixels in an image connect to pixels from the next image in the sequence. This is difficult to exactly calculate, so various estimation methods have been proposed. One group of such methods are called **differential**, based on partial derivatives of the image. In this assignment, we will implement and test this group's two most popular algorithms: Lucas-Kanade (LK) and Horn-Schunck (HS).

II. Experiments

Let us first comment on the impact of parameter σ used for Gaussian smoothing and derivatives. This parameter was initially set to 0.5% of the smaller dimension of the input image, a "trick" learned at another course. We then tested values from 0.5 up to 10, fixing σ for smoothing and varying the ones used for derivatives, and vice versa. It was established that the initial setting generally produced the "best" results, or almost the best, to the point where optimizing σ further brings marginal improvements. Hence, we decided to stick with the initial setting for all further tests.

The first pair of images we will test are a random noise image, and the same image rotated by 1 degree counter-clockwise. The parameters used and results can be seen in Figure 1. Both algorithms perform quite well here.

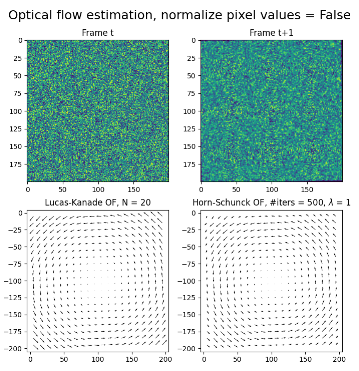

Figure 1. LK and HS tested on a random noise image, and the same image rotated by 1 degree counter-clockwise.

The next pair consists of two images from a lab, as seen in Figure 2. LK performs considerably worse than HS. The flow is completely wrong on the pillar, while elsewhere there is no flow or we see small "flickers". This happens due to the inherent drawback of LK, which is that it cannot reliably estimate flow on edges, and untextured, homogeneous regions. HS gives very satisfying results - it shows larger flow vectors for closer objects, smaller for those further away, and the entire flow field is very smooth. The most noticeable flaw is the lack of flow in the top-right corner. This is most likely because the pixel intensities of the pillar and the wall behind it more or less match, and as that wall patch doesn't exhibit any movement, the same will hold for the pillar.

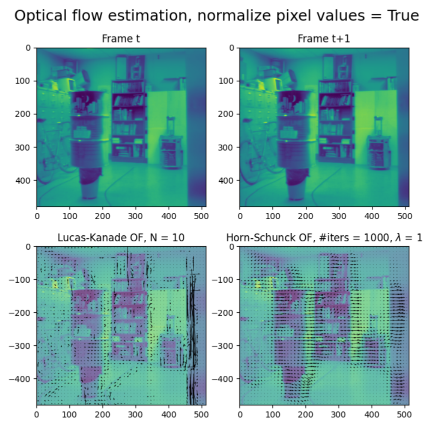

Figure 2. LK and HS tested on a pair of lab images.

In Figure 3, we tested a pair of images taken from Waffles the Terrible. Both algorithms produce very satisfactory results here. LK slightly struggles with untextured homogeneous parts again, but the flow on Waffles himself is pretty good, which can also be said for HS.

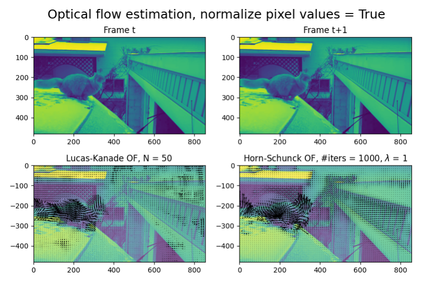

Figure 3. LK and HS tested on images taken from Waffles.

In Figure 4, we used frames of Waffles where he moves really fast. We see that both algorithms struggle to calculate the right flow. LK flow vectors are out of proportion and pointing in wrong directions, which makes sense, since LK generally works best when there are small motions in the image (small displacements assumption). HS does a slightly better job, but still gives incorrect flow in some places.

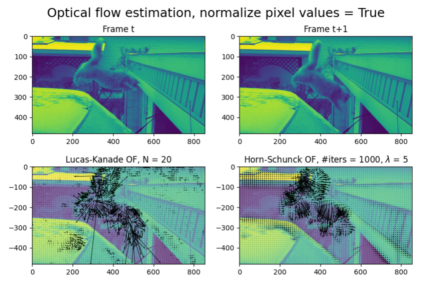

Figure 4. LK and HS tested on images where Waffles moves really fast.

In all of the shown tests, the parameter we needed to set for LK was N, the size the neighborhood used when calculating displacement vectors for a pixel. For HS, we need to choose the maximum number of iterations, and λ. The higher the value of N, the smoother the overall LK optical flow is - this is especially evident when there is lots of movement in the image. Bigger values for N (20, 30, 50) reduce the impact of the aforementioned inherent drawback of LK, and there aren't as many small flickering vectors. This can be seen in Figure 5. Likewise, if we use a very large N, we won't capture the movement in the image, and we will essentially get no flow - it will become "too smooth".

For HS, maximum number of iterations refines the optical flow, slowly bringing it closer to convergence. We generally need 400-500+ iterations to get satisfactory results, as with a small number of iterations the flow vectors are out of proportion and pointing in wrong directions. This is seen in Figure 6.

On the other hand, λ affects how smooth the transitions between flow vectors are, but effects depend on the image and the used number of iterations. Setting λ somewhere in the range 1 to 5 gives good results, depending on the images in question. One example result is shown in Figure 7.

Examples of good parameter settings were provided in Figures 1, 2, 3 and 4. All of them were empirically and separately set for each pair of images.

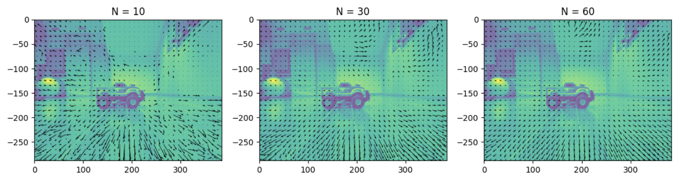

Figure 5. Effects of different values of N for LK.

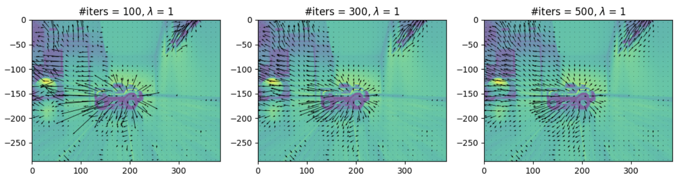

Figure 6. Effects of different number of max iterations for HS.

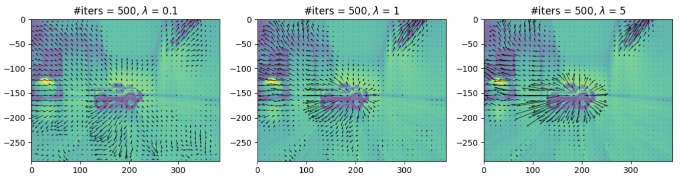

Figure 7. Effects of different values of λ for HS.

The overall better performance of HS is traded for higher execution times than LK. We can see the execution times for LK in Table I, and for HS in Table II. Higher values of N result in larger execution times, which makes sense since convolutions with larger kernels take more time. While not exactly shown here, λ has little to no effect on the execution time - it is mostly the maximum iterations parameter which dictates that. We also tested the execution time of HS when the flow field components are initialized with the output of LK, instead of just 0. We set N rather high so the difference is more noticeable, but in essence, this version of HS takes longer to execute because of the additional calculations coming from LK.

Image pair/N	10	20	50
Lab	0.347	1.162	6.401
Waffles	0.440	1.747	10.543

Table I
Execution times for LK, for different values of N, on two image pairs. Times are given in seconds.

Image pair/HS	Regular	Init. w/ LK
Lab (200 iters)	4.257	10.514
Lab (500 iters)	10.657	17.323
Lab (1000 iters)	22.053	27.475
Waffles (200 iters)	7.796	18.128
Waffles (500 iters)	19.412	29.509
Waffles (1000 iters)	39.244	51.025

Table II
Execution times for HS, for different number of iterations, on two image pairs. Times are given in seconds. For all tests, $\lambda = 5$, N = 50.

HS can be sped up by initializing the flow field components with the output of LK, but not with respect to time, as we saw in Table II. The actual speed up we get is in the number of required iterations. Namely, if we initialize HS with LK, we can achieve the same or even better results in way less iterations, e.g. 100-200, compared to e.g. 1000 when using regular HS. We can see that in Figure 8. Still, the actual performance of this type of HS depends mostly on how good LK's flow field is - if LK flow is not good, HS will not be able to fix it in such a small number of iterations, as can be seen in Figure 9.

If we wanted to speed up HS with respect to time, we could allow HS to "converge", i.e. we stop iterating further if the changes in both flow field components become sufficiently small.

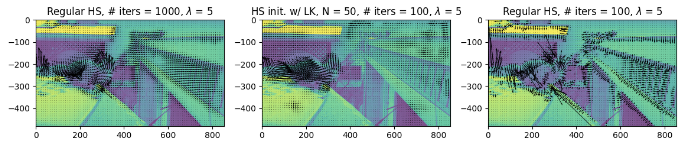

Figure 8. Speeding up HS by initializing it with the output of LK.

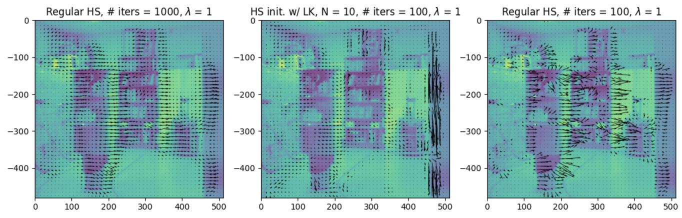

Figure 9. Initializing HS with the output of LK is not necessarily good. See Figure 2 for the LK flow field.

III. Conclusion

We tested two differential methods for estimating optical flow - Lucas-Kanade and Horn-Schunck. In general, HS gives more satisfactory results than LK, as it doesn't assume small displacements and is more reliable when estimating flow on edges and homogeneous regions. We pay for this better performance with larger computation times, but this can be improved by letting HS converge. We also discussed that HS can be sped up, in terms of the required number of iterations, by initializing it with the output of LK. However, the performance of HS in this case depends mostly on the performance of LK.