# Exercise 1: Optical flow

Martin Božič

## I. Introduction

Optical flow can be described as a motion between individual pixels on an image plane. It assumes, that between two consecutive images, pixel intensities change only by small margin, so we can assume pixels position change just by observing their intensities. Based on that, we can assume motion on specific parts of an image.

There are multiple methods, that enable us to compute optical flow. In this paper we focus on Lucas-Kanade method and Hord-Schunck method. We implement both methods and test them on three different images. There are multiple parameters that determine effectiveness of both methods. We test both methods using different values for parameters, that we think have greater impact on performance of both methods, and report the results. We also make time comparison of both methods and purpose some speed up improvements.

## II. Experiments

First we implement both methods and test them using two test images. The first image consists of randomly generated pixels and we generate second image by rotating the first image by 1 degree to the right. As it can be seen on picture 1, the optical flow, returned by both methods, is correct and goes in circle pattern from left to the right. The result obtained by Hord-Schunck method gives us slightly better results, with smoother optical flow vector predictions.

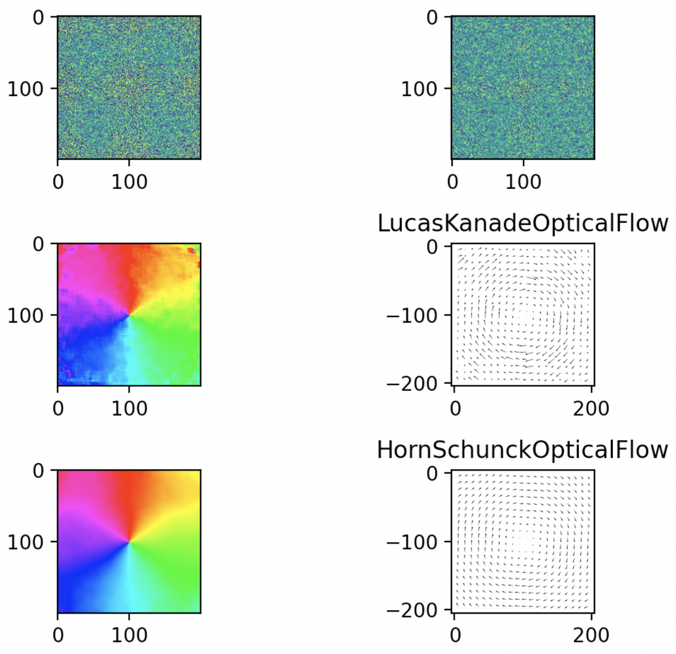

Figure 1. We can see performance of both algorithms on randomly generated image.

In our implementation of Hord-Schunck method, after every cycle of 100 iterations, we check whether there are changes between old and new calculations of optical flow parameters u and v. We measure similarities using cosine distance and in the case of small changes between old and new calculations, we finish with method execution. This gives us a slight improvement in method execution time.

We further test both methods using series of other images, one of them being in the room, where there is a car stationed in the middle of the picture and we move slowly towards it. As we can see on picture 2, both methods give us correct results, with Hord-Schunck method again outperforming Lucas-Kanade method.

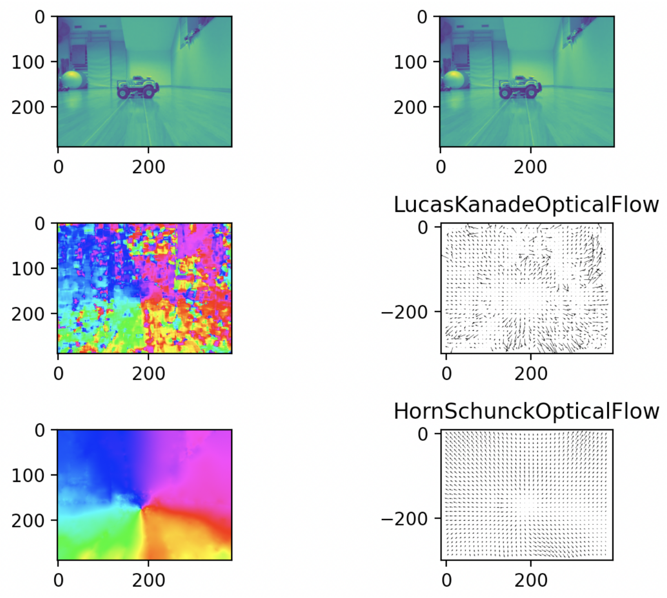

Figure 2. We can see performance of both algorithms. We get better results using Hord-Schunck method.

When calculating optical flow on images, that have a lot of similar surfaces, with very little tangible attributes, like white wall for example, there can be many optical flow calculation errors of, especially when using Lucas-Kanade method. These calculation errors can be seen on the bottom left image, on picture 3. Especially on the part, where there is a white wall. We manage to fix that by correcting all values, in determinant matrix, that are smaller, than certain threshold, with threshold value. In our implementation we set threshold value of 1e-7. The corrected optical flow estimation can be seen on bottom right image, on picture 3.

There are many parameters that determine performance of both methods. When we calculate derivatives in x and y direction, we improve derivative calculations using Gaussian smoothing. To construct Gaussian filter, we have to define and set the value of sigma. We find out, that sigma, has big impact on the performance of Lucas-Kanade method. For example we chose to set sigma around 0.5 and if we increase it to 3.0, the performance of algorithm drops significantly. Drop of performance isn't that noticeable when using Hord-Schunck method, but we noticed, that if we increase value of sigma, we need more iterations to achieve better results. Results of both methods for different values of sigma can be seen on picture 4.

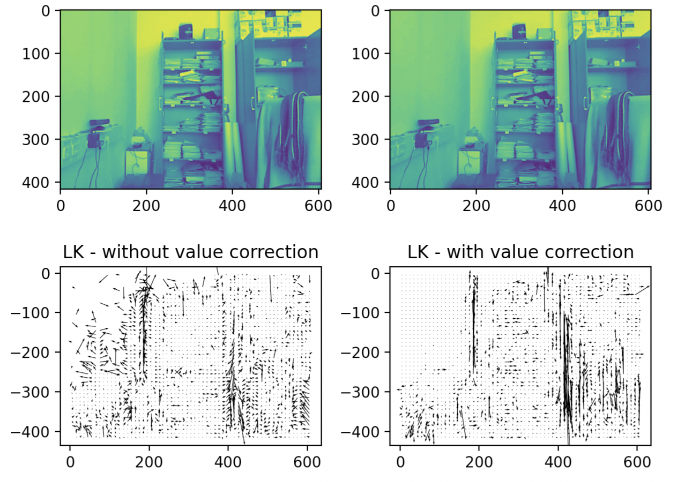

Figure 3. We can see performance of Lucas-Kanade method with and without small values correction in determinant calculation.

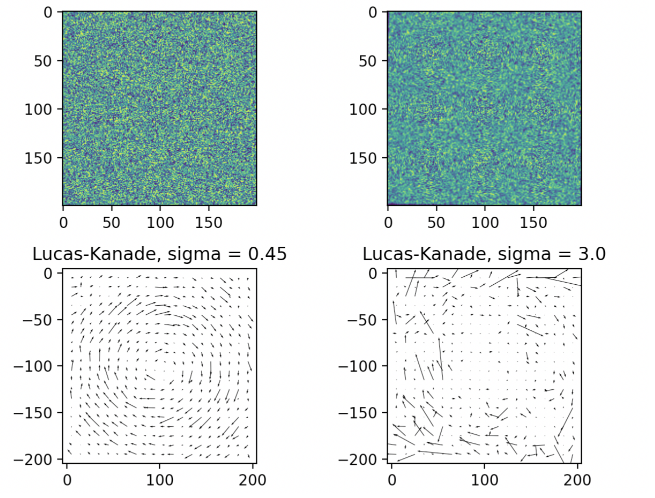

Figure 4. We can see performance of Lucas-Kanade method using different sigma values, when smoothing and calculating derivatives.

Another thing that influences the performance of Lucas-Kanade method is the size of each pixel neighbourhood that is summed together. We test neighbourhoods of 8, 24 and 80 pixels and find out that considering small neighbourhood of 8 pixels (done by 3 x 3 filter) gives us best results.

When calculating difference of two consecutive images in both methods, we decide to also smooth image after subtraction. Here is another sigma parameter, that we have to set. This sigma has smaller impact on performance of both methods and we find out that it's best if we don't set it to high. in our case we set it to the value of 1.0.

Horn-Schunck method depends heavily on the number of iterations and lambda parameters. If we increase the number of iterations, we get better results, but increasing the number of iterations also increases method execution time. For lambda parameter we get similar results as long as we keep it relatively small, e.g. smaller than 1.0. Otherwise we can sometimes get results, that contain very long and unrealistic optical flow vectors, or we sometimes need more iterations, to achieve same results.

When comparing execution times of both methods, we see, that Lucas-kanade method outperforms Horn-Schunck method. Horn-Schunck method execution times dractically depend on number of iterations in method execution. Horn-Schunck method can be speeded up if we use Lucas-kanade method when initializing values u and v. We can see execution times for both algorithms, with speeded up Horn-Schunk and Horn-Schunck method with different number of iterations on table I.

| **Method** | **Execution time** |
|---|---|
| LK - base | 0.0072s |
| HS - base | 17.36s |
| HS - speed up | 7.24s |
| HS - 100 iterations | 0.16s |
| HS - 1000 iterations | 0.71s |

Table I

We can see execution time comparison for Lucas-Kanade method and different versions of Horn-Schunck method.

If we want to achieve comparable execution times of both methods, we have to run Horn-Schunck method in just 3 or less iterations. The results of optical flow, that we get using both methods can be seen on picture 5. We can see that in this case Lucas-Kanade method outperforms Horn-Schunck method.

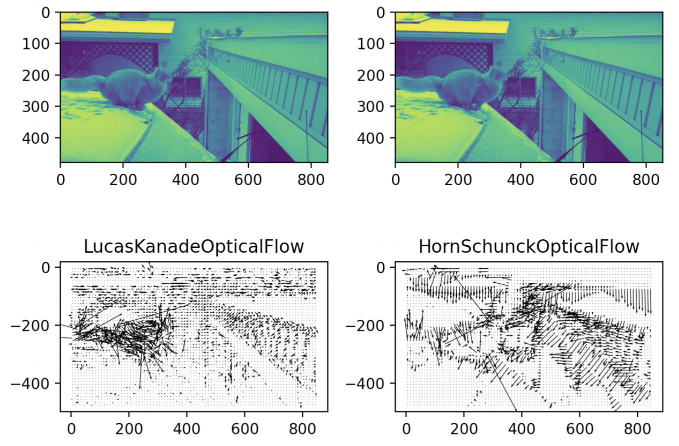

Figure 5. We can see performance of Lucas-Kanade method and Horn-Schunck method done in only 3 iterations.

## III. Conclusion

In this paper we compare two methods for calculating optical flow from two consecutive images. We see that Horn-Schunck method outperforms Lucas-Kanade, but needs much more time to execute. Which can sometimes be a problem, especially if we need to calculate optical flow in real time. We get the best performance if we combine both methods. We can use Lucas-Kanade method to initiate u and v vector values and then just fine-tune them using Horn-Schunck method. In this case we get good results regardless of number of iterations that we can make using Horn-Schunck method.