Exercise 1: Optical flow

Nikolay Vasilev

I. Introduction

Optical flow refers to the displacement of objects, surfaces and edges over pair of frames. It is a velocity field, which transforms one image into the next image in a sequence. In the following paper we will evaluate the implementation of two popular techniques for estimating optical flow from a sequence of images - the **Lucas-Kanade** and **Horn-Schunck** methods [1]. The Lucas-Kanade method assumes that the optical flow is very similar in the local neighborhood and uses that to define system of equations, which can be solved using the least squared method. On other hand, the Horn-Schunck algorithm is based on a global smoothness constraint by minimizing a global energy.
By implementing these method in Python, we will be able to compare the results from them by testing them on random noise and on pair of images. We will find ways to improve the Lucas-Kanade's estimated optical flow by using different techniques into our implementation. We will also try and discover, what are the best parameters and techniques, which will help us improve both algorithm's optical flow and performance. Of course, we will also evaluate the pyramidal Lucas-Kanade and figure out, in which cases and why is it better than the non-pyramidal and how close is its flow to the one calculated from the Horn-Schunck method.

II. Experiments

A. Random noise

We start our evaluation by running the Python code, i.e. by testing the Lucas-Kanade, Horn-Schunck and pyramidal Lucas-Kanade methods on random noise. This means that for the first image we have a random noise, which is then on the second image rotated. In Figure 1 we can see example of this, for rotation by one (left column) and three (right column) degrees. The parameters in this case do not change much the output, which is why they will not be our main focus right now. As we can see, for different rotations we can make different conclusions:

- **One degree rotation** – first thing, which we can see here, is that the Lucas-Kanade and the Horn-Schunck implementations for rotation of one degree both work pretty good. But if we look at outside part of the circle, we can see that the Horn-Schunck's optical flow looks more smooth, which was expected, since we have highly non-rigid motion. This non-rigid motion is also the explanation, why in this example the pyramidal Lucas-Kanade have worse flow than the non-pyramidal.
- **Three degree rotation** – in this case, we have more rigid and larger motion, which is why the pyramidal Lucas-Kanade works pretty good, while the other two methods can only calculate somehow accurate flow of the inside part of the circle. Even thought the Lucas-Kanade and Horn-Schunck methods do not estimate good in this case, if we look closer at the Lucas-Kanade method, we can see that its optical flow is even better than the Horn-Schunck's flow. The reason for that again is the larger motion of the pixels between both images in this example.

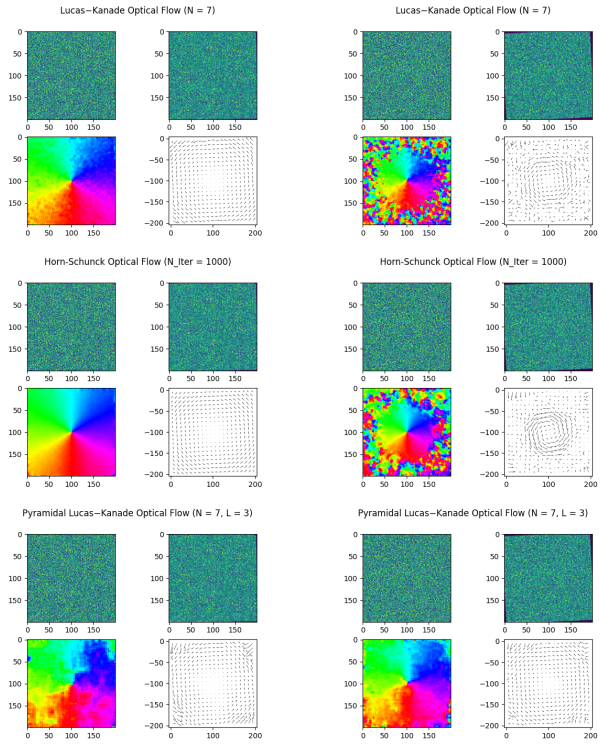

Figure 1. Lucas-Kanade (1st row), Horn-Schunck (2nd row) and pyramidal Lucas-Kanade (3rd row) methods applied to a random noise, which is rotated by one degree (left column) and three degrees (right column).

B. Images

Now, when we made sure these algorithms work on random noise, we have to evaluate them also on images of the real world and see how we can change their parameters and improve them in order to get better optical flows and better performance. For the test we have chosen pair of images from the *collision* directory (00000166.*jpg* and 00000167.*jpg*), from the *disparity* directory ($office2\_left.png$ and $office2\_right.png$) and from the *lab2* directory (020.*jpg* and 021.*jpg*).
After doing some experiments on the following images, we were able to improve the algorithms. The Lucas-Kanade algorithm was improved the following way:

- **Harris response** – to improve the accuracy of the optical flow estimation, we needed to identify the corners and the edges in the image. Usually this may be done by computing the eigenvalues λ_1 and λ_2 of the squared matrix and making sure that the following conditions are valid: *a)* eigenvalues λ_1 and λ_2 are not too small; *b)* ratio λ_1/λ_2 is not too great. Instead of computing the actual eigenvalues, we calculated the Harris response, which not only allows faster performance, but also allow us to use a single threshold. It is calculated the following way:

$$det(A^T A) - \alpha * trace^2(A^T A)$$

, where $det(A^TA)$ is the determinant of the squared matrix, which we already need to calculate for the u and v components, α is some constant and $trace(A^TA)$ is the trace of the squared matrix, which may be calculate the following way:

$$trace(A^TA) = \sum I_x^2 + \sum I_y^2$$

With the help of the Harris response, we create a mask, which is a matrix, where a pixel has a value 1 if the Harris response is equal or bigger than the threshold, or if the pixel's value of the Harris response is the maximum in a chosen neighborhood, with the same size as the kernel. Otherwise the corresponding pixel in the mask will have value 0. In the end, this mask is multiplied with the u and v components. With a lot of testing we discovered that the best values for the parameters are the following: *a)* for the threshold $1e-8$; *b)* for the α 0.02;

- **Determinant value** – Because we use the determinant in division to calculate the u and v components, it must not be 0. That is why we have added a threshold, which is used in order to check if the values in the determinant matrix are smaller than it, i.e. close to 0. If they are actually so small, the value is set to the threshold value. We figured out that this threshold may change the optical flow a lot and the best fitted value that we found for this parameter is $1e-3$.

For the kernel size we found that the best value is 7 and for the maximum level of the Gaussian pyramid in the pyramidal Lucas-Kanade we got best results, when using 3 levels, since the images are not so big. For the Horn-Schunck method we set the parameters *iterations* and *lambda* to 1000 and 0.5, which proved to be the best in our cases. Of course, we also set a threshold in order to check for convergence of the optical flows and stop the iterations before, if needed. For that we are using the value $0.87e-3$, which is the smallest possible that still returns a good optical flow. It helps us increase the performance by stopping 301 iterations before, but only for the image pair from the *collision* directory, since there we have the least displacement. In Table I we can see the best performance time of each method on each image pair, after we used all these parameters. As expected the optical flow of each method is calculated the fastest for the collision image pair, since there we have the smallest displacement. This we can see also in Figure

Table I
Performance time of each method

	Speed
LK (collision)	0.11s.
LK (disparity)	0.26s.
LK (lab2)	0.25s.
PLK (collision)	0.16s.
PLK (disparity)	0.37s.
PLK (lab2)	0.36s.
HK (collision)	6.38s.
HK (disparity)	20.61s.
HK (lab2)	20.96s.

2, where the optical flows from each method for each image pair are visible. The first thing, which we can see in this scenario, is that the pyramidal Lucas-Kanade here performs better than the non-pyramidal, but still not so good as the Horn-Schunck method. The Lucas-Kanade method calculates pretty good opticle flow for the *collision* image pair. Since the pyramidal

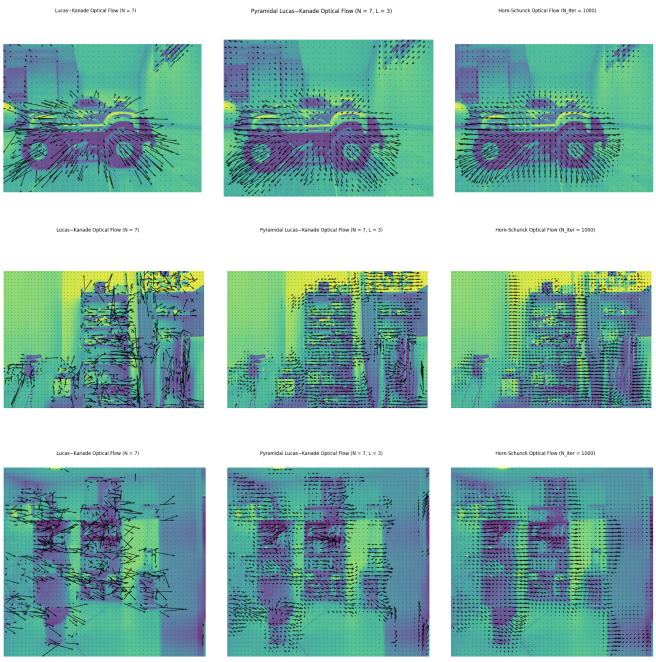

Figure 2. Lucas-Kanade (1st column), pyramidal Lucas-Kanade (2nd column) and Horn-Schunck (3rd column) methods applied to images from the folder collision (1st row), disparity (2nd row) and lab2 (3rd row).

Lucas-Kanade improves the non-pyramidal so much, we would think that just running the Lucas-Kanade iteration multiple times on the same scale would improve the performance as well. This is not always true, because it may improve the accuracy at some cases, but the problem of scale changes may be overcome only by multi-scaling the images. Now, when everything looks good, there is one last thing that may be improved, which is the performance of the Horn-Schunck. In some cases, we can speed it up by initializing it with output of Lucas-Kanade. It depends on that, how good the Lucas-Kanade algorithm will perform on the image. In the first example we were able to improve and decrease the needed iterations by 30 (more than 1 second), but on the other 2 pair images this wasn't the case. As we can see the Lucas-Kanade performs pretty good on the first image pair, so it is expected that it would also speed-up the Horn-Schunck method. Of course, we can decrease the wanted threshold for the convergence in the Horn-Schunck method and see that also in this case the Lucas-Kanade may help us speed up the method, but the estimated optical flow won't be as good.

III. Conclusion

Even thought the Horn-Schunck method usually has better optical flow than the Lucas-Kanade, the pyramidal implementation of the Lucas-Kanade method may help us get an optical flow that is pretty close to the one we get from the Horn-Schunck method. If we find the best parameters and techniques for chosen method, we can get very good optical flow estimation with pretty good performance time.

References

[1] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/kanade meets horn/schunck: Combining local and global optic flow methods," *International journal of computer vision*, vol. 61, pp. 211–231, 2005.