Optical Flow

Žiga Trojer

I. Introduction

Our main task was to implement two methods of optical flow estimation. During the implementation, we learned the pros and cons of both algorithms and considered when to choose one of them. Also, we tried to figure out how to choose the optimal values of the parameters that affect the final result.

II. Experiments

Basic implementations of Lucas-Kanade (L-K) and Horn-Schunck (H-S) algorithm were tested on rotated random noise images. If not stated otherwise, we are using the following parameters: for L-K, 3×3 neighborhood, $\sigma = 1.5$ when using `gaussderiv` and $\sigma = 1$ when using `gausssmooth` to smooth temporal derivative. For H-S, 1000 iterations, $\sigma = 1$ and $\lambda = 0.5$. Optical flow will be represented with 2 images: first will show be color map - angle is represented by hue and magnitude by saturation, and second - representation with vectors. Figure 1 shows that estimation of optical flow (image I_2 obtained from I_1) from H-S (bottom) is smoother than from L-K algorithm (top). This is due to different technique, since L-K assumes the optical flow is very similar in the local neighborhood and H-S is using global smoothness constraint. Also, H-S is more complex and slower method, so it makes sense that is gives better results. Optical flow from both algorithms is stronger near the edges, since the image was rotated around center and pixels further from center were moved by a greater distance.

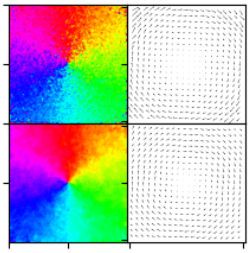


Figure 1: Optical flow of random noise rotation, using Lukas-Kanade (top) and Horn-Schunck algorithm (bottom).

We tested both algorithms on different sets of images. We tried to choose such images to be as different as possible in terms of what the background is, how many sharp edges there are, the size of the shift in the consecutive images, etc... We normalized all images. The differences between the performance of algorithms over normalized and non-normalized images will be highlighted later. When running L-K on those images, the result was not as expected. The problem in the algorithm was in a step when calculating displacement vectors u and v - the division with 0 appeared. Determinant D of a covariance matrix may contain many zeros - covariance matrix is ill-conditioned matrix. We think this is due to the fact that we have a lot of background in many images. With a small shift, we can not find the new position of a pixel with just following the derivatives (change is not local anymore). On Figure 2 we observe bad performance of unstable L-K algorithm - on both images, there is quite a lot of background - vectors there are not defined. Also, some vectors are misaligned and some of them are too long. We solved this problem in a following way; first, we tried to substitute undefined values in u and v with zeros, then we also tried substituting zeros in D with some small number ϵ. Both ways gave similar results, so we used the second approach. We can compare the performance of this improved method in Figure 2 (row 2 vs. row 3). The results are much better, but there are still some shortcomings of this method - in the picture of honey, there are some flow vectors that are misaligned and, above all, indicate a big shift, which is not right.

When comparing improved L-K and H-S, we see that output of H-S is more clear - flow vectors are more aligned and flow vectors for background are almost zero (means that background does not move - very nicely seen in an image of honey). In a picture of a cat, where cat tilts his head, there is more clear presentation of vector flow too: you can almost recognize head of the cat.

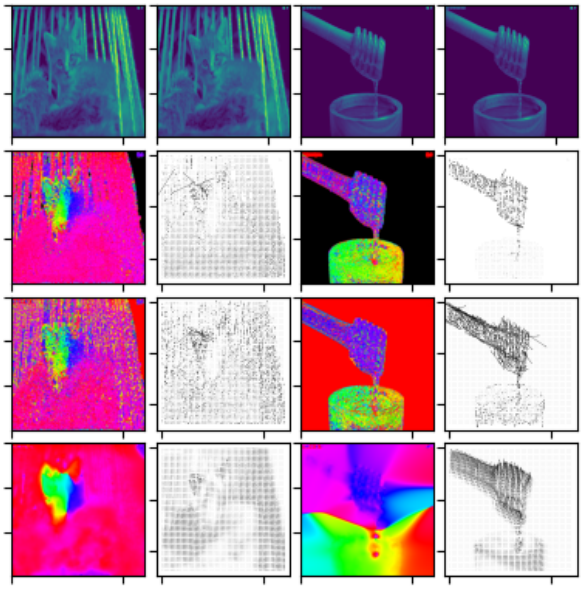
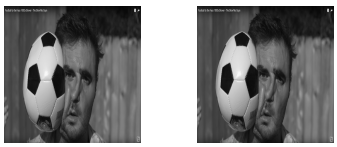


Figure 2: Optical flow for two different images, using unstable Lukas-Kanade (second row), stable Lukas-Kanade (third row) and Horn-Schunck algorithm (fourth row) [1].

From this point on, we will analyze only one image and optical flow be will represented only with vectors (we will omit the presentation of optical flow with color map). Figure 3 represents pair of images for which we will analyze optical flow. Images were taken from a video, where a ball hits man's face [2]. Figure 4 shows flow results for 3 different algorithms on non-normalized and normalized images. Algorithms works better with normalized images, because optical flow is easier to compute when values are in range $[0, 1]$. When the image is normalized, the background changes gain less weight when calculating optical flow. In our opinion, optical flow is best seen in the last map (H-S algorithm) - the reason is that optical flow is smooth and consistent. L-K algorithm performs poorly because it assumes the optical flow is very similar in the local neighborhood. This clearly does not hold everywhere in this

image: some parts of the face moves faster than others, some stays even in the same place for few moments (hair moves slower than cheek). There is also a small downward shift of the whole image, which is clearly seen in terms of the orientation of the vectors (especially in non-normalized images).



(a) fist image (b) second image

Figure 3: 2 consecutive images from video [2].

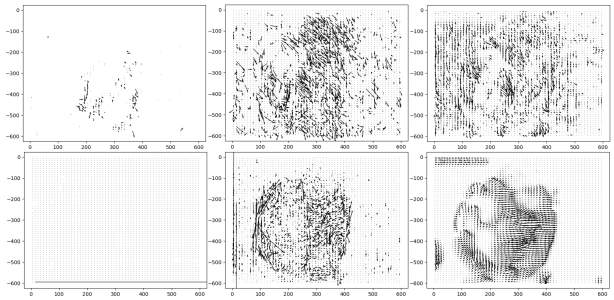


Figure 4: Optical flow from first image to second from Figure 3 using *above*) non-normalized image, *below*) normalized image: *left*) unstable L-K, *middle*) stable L-K, *right*) H-S algorithm.

A. Parameter selection

For both algorithms, we need to determine σ, which is a parameter for calculating derivatives I_x and I_y. On our test images, the optimal choice is $\sigma = 1.5$ for L-K and $\sigma = 1$ for H-S, but for some other images, this parameter may be different. Setting σ too big, optical flow vectors become very long and not accurate, setting it too low, the orientation of some vectors change. This influence of σ is bigger for L-K algorithm.

L-K algorithm takes one more parameter n - size of neighborhood. Figure 5 shows the impact of changing this parameter on the optical flow. Increasing the size of n could improve or could worse the optical flow - depends on an image. If the whole image is moved, then a slight increase in n has a good effect on the optical flow, as the algorithm assumes that the optical flow is similar in the $n \times n$ neighborhood of the pixel. If only part of the image (e.g. an object) is moved, increasing n too much has a negative effect on the optical flow. In our example, setting $n = 10$ has a positive effect (more consistent vector orientation), but setting it to $n = 50$ results in totally different optical flow. If we had a very large image, we would need to consider also larger n.

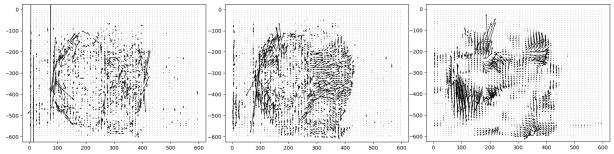


Figure 5: Different choice of parameter n (size of neighborhood) for L-K: $n = 2$, $n = 10$ and $n = 50$.

H-S algorithm takes two parameters: n_iters - number of iterations and λ - it reflects the influence of the smoothness. Increasing the first parameter has a positive effect on the quality of optical flow, as the algorithm takes more steps and thus the probability of convergence is higher. On the other hand, it negatively affects the execution time of the algorithm - larger the parameter, more complex the algorithm and slower the execution. λ on the other hand, increasing the parameter value improves the quality of optical flow in case there is global movement and no local one. Figure 6 shows that increasing λ too much, vectors get a larger magnitude (which makes optical flow less readable).

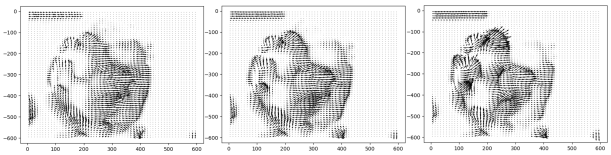


Figure 6: Different choice of parameter λ for H-S: $\lambda = 0.5$, $\lambda = 2$, $\lambda = 10$.

B. Improvement

We measured time of both algorithms and tried to improve the speed of H-S algorithm by initializing the algorithm with L-K output. The fastest algorithm is L-K: execution time was 0.15 seconds. H-S algorithm on the same image took 18.02 seconds. The output of L-K was taken as an input of H-S algorithm - instead of setting u and v to zero before the first iteration, we set them to u and v, that was the output of L-K. With this approach, we optimized our algorithm - execution time was 15.98 seconds. Figure 7 shows the results of those three algorithms. Optical flow of optimized algorithm is very much influenced by the L-K output. If L-K gives a good approximation of optical flow, then it makes sense to use a faster H-S algorithm.

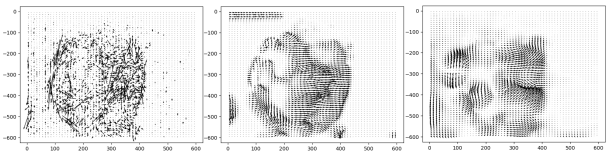


Figure 7: L-K performance, H-S performance and performance of improved H-S by initializing with L-K output.

III. Conclusion

We implemented both methods, highlighting the main advantages and disadvantages. We managed to improve both algorithms - the first by preventing indefinite division, and the second by determining the initial state with an algorithm that is faster. When choosing a method, it is important whether speed or accuracy is important to us. Usually we have to do some trade-off. The first method could not be used for any detailed optical flow analysis, and the second could not be used for live optical flow monitoring.

References

[1] E. Belsky, "Real 12k hdr dolby vision." [Online]. Available: https://www.youtube.com/watch?v=eXju5LkrYs4

[2] T. S. M. Guys, "Football to the face 1000x slower - the slow mo guys." [Online]. Available: https://www.youtube.com/watch?v=On1CsbTwlDs&t=237s