# Introduction

Optical flow is estimation of two dimensional vector that indicates motion of objects, surfaces and edges in scene(Figure 1). In this paper, for estimation of optical flow Lucas Kanade algorithm and HCVFlow model is examined and compared.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 1 Optical Flow Demonstration

# Lucas Kanade Method

Classical methods dates back to early 80’s. Improving on the brightness constancy assumption, the Lucas-Kanade algorithm formulates optical flow estimation as a local least-squares problem, assuming that motion is constant within a small neighborhood around each pixel [1]. This statement yields derivation in Figure 2.

A math equations with numbers and symbols

AI-generated content may be incorrect.

Figure 2 Optical Flow Derivation

Constancy of flow assumption can be seen in Figure 3. If small patches are examined seperately it can be seen that flow over the small patch is the same.

A black and white background

AI-generated content may be incorrect.

Figure 3 Constancy of Optical Flow

Using these assumptions algorithms starts working by calculating image gradients(vertical, horizontal and temporal). For 5x5 patch we get 25 equations(Figure 4).

A math equations and formulas

AI-generated content may be incorrect.

Figure 4 Least Square Problem Formulation

Since this is a tall matrix, following least square approximation is used. As you can see A is not invertible here however when we multiply with its transpose we get 25x25 invertible matrix(Figure 5).

A math equations with numbers and symbols

AI-generated content may be incorrect.

Figure 5 Least Square Formulation (cont)

A black text with a white background

AI-generated content may be incorrect.

Figure 6 Least Square Solution

You can see that A^T A matrix is very similar to Harris corner detector matrix. Corners occured when eigenvalues are large. This is also where this algorithm works best.

# HCVFlow

With the advent of convolutional neural networks, various attempts in estimation of optical flow is made [2, 3,4,5]. Most of the state of art papers are based on cost volumes which will be further explored in further sections. However, computational complexity problem of constructing this cost volume is either neglected or poorly resolved.

In HCVFlow, instead of 4D volume cost, two 3D global cost volumes are used. Also 4D local cost volume with local search space is introduced. These cost volumes significantly reduces memory consumption while preserving performance[6].

As can be seen from Figure 6, algorithm starts with FeatureNet block that extracts feature map of source and target images in ¹⁄₁₆ and ⅛ resolution of original images.

A diagram of a street

AI-generated content may be incorrect.

Figure 7 FeatureNet in HCVFlow

Then, 2 3D global cost volumes and 4D local cost volumes are created. First 3D global cost volume is horizontal that checks possible horizontal motion. For each pixel in source feature map, HCVFlow calculates its correlation with pixels along the same row limited to window in length of D. This gives horizontal cost volume. For example, in figure 7 correlation between red pixel and green pixels are taken. In this case D(window size) is selected as 2. As a result, correlation volume for horizontal displacement is created.

A diagram of a bar with a red line

AI-generated content may be incorrect.

Figure 8 Horizontal 3D GCV

In the same manner, vertical cost volume is created as shown in figure 8.

A graph of a line and a graph

AI-generated content may be incorrect.

Figure 9 Vertical 3D GCV

After the creation of cost volumes “Top-k” strategy is implemented. Instead of keeping all the correlation scores, only k most relevant pixels are kept. This significantly reduces memory usage. As a result cost volume with dimensions HxWxkxD are created. Note that H and W values are 1/1₆ of original image resolution.

However output of Top-k strategy gives us sparse best matches. In order to enrich the information kept in these volumes, aggregation module is used. It uses cost volume regularization network that looks like U-Net built in 3D. They consist of 3D convolutions, downsampling/upsampling blocks and skip connections. Overall GCV structure can be seen in figure 9.

A diagram of a global cost

AI-generated content may be incorrect.

Figure 10 3D Cost Volumes

Global cost volumes may miss fine grained local matching details due to the Top-k and 1D search approach. To resolve this issue, a local 4D cost volume is proposed. Local 4D cost volumes are calculated with feature map that is at 1/8 resolution of original image resolution. It is calculated between every pixel in source image feature map and corresponding small local search window in target image feature map. The dimension of this calculation becomes HxWx(2r+1)x(2r+1) where r indicates window radius in local search area. The construction of this volume is well illustrated in figure 10.

A diagram of a diagram of a local cost

AI-generated content may be incorrect.

Figure 11 4D Local Cost Volume

Concatanated hybrid cost volume is then fed into convolutional gated recurrent unit. Before entering the ConvGRU loop, the initial features extracted from the input images are passed through a separate context network as h0 which is initial hidden state for GRU. For each t ConvGRU has flow in t-1 and hybrid cost volume. Current flow estimate is used to perform a lookup from hybrid cost volume(By checking relevant correlation information). The ConvGRU updates its hidden state, learning to refine the motion prediction. After a set number of iterations, the final refined flow field is output. The structure of typical ConvGRU is illustrated in figure 11.

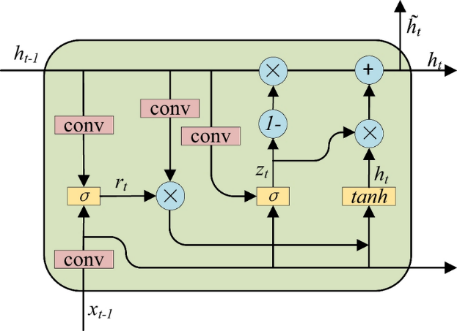


Figure 12 ConvGRU Model

# Experiments

In order to convey experiments, small dataset of three video are collected. These three videos cover all three possibilities which are the one where traditional and CNN based model work best, the one where CNN based model succeed but traditional model fails and finally the one where both of them fail. In order to have better understanding videos can be seen in specified directories.

### First Case

The first case is the where Lucas Kanade method and HCVFlow model works properly. In this case first\_case.mp4 is used. It can be found under the folder *“./experiment\_videos/* first\_case/*”.* Here are the screenshots of some of the frames from video. As you can see from figure 12, object is moving towards left. This shot is taken by sliding phone camera towards right with static bottle.



Figure 13 First Case Frames

Here is the output of Lucas Kanade algorithm(figure 13). As you can see, first good features are detected (using Shi-Tomasi method ) and optical flow algorithm works on them. It worked perfectly. Original video can be seen under file “experiment\_results/first\_case/ first\_case \_lucas\_kanade.mp4”.

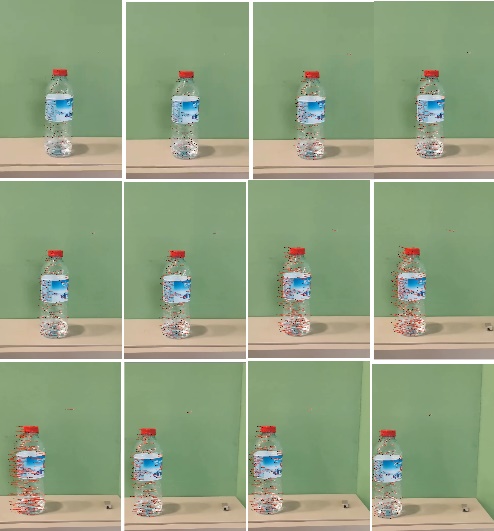


Figure 14 First Case Lucas Kanade Results

Finally,HCVFlow algortihms is run on this video. Result of the model can be seen in figure 14 (Blue is indication of left movement whereas red is indication of right movement). The way algortihm returns pixelwise flow is astonishing compared to traditional algorithm. Original video can be seen under file “experiment\_results/ first\_case / first\_case \_hcvflow.mp4”

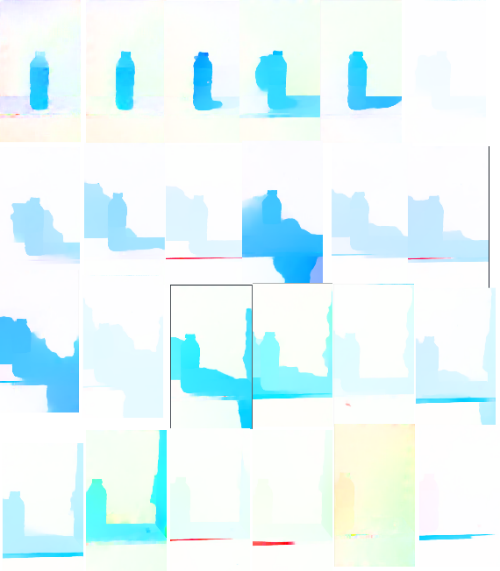


Figure 15 First Case HCVFlow Results

### Second Case

The second case is where traditional Lucas Kanade method fails, HCVFLow model succeeds. Lucas Kanade method fails when object permanance is needed. Those are the scenarios where objects disappear due to occlusion or something similar. Figure 15 shows sampled frame of the video for this case(Video is taken from Youtube). Original video is longer and covers more scenarios(diagonal etc.). It can be seen under *“./experiment\_videos/ second\_case /second\_case.mp4”.*

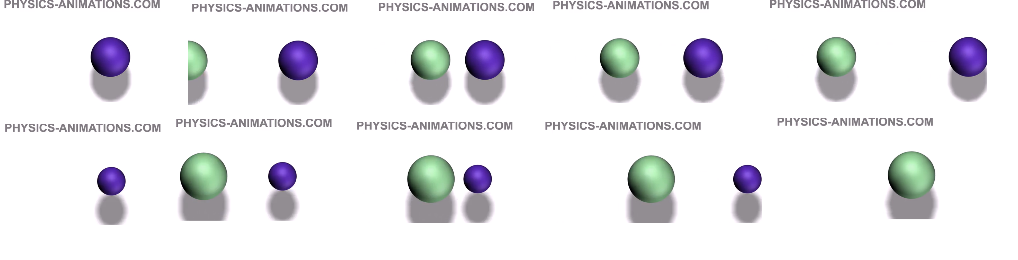


Figure 16 Second Case Frames

Results of Kanade-Lucas optical flow algorithm can be seen in figure 16. Original video can be seen for better understanding under *“./experiment\_videos/ second\_case / second\_case \_kanade\_lucas.mp4”.*

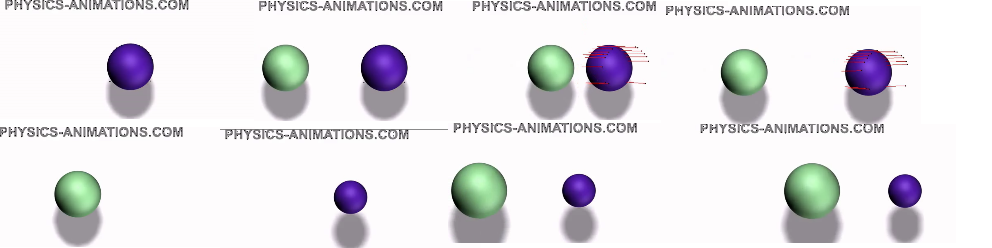


Figure 17 Second Case Lucas Kanade Results

As you can see, in first frame features are detected and optical flow algorithms work until features disappear. In fifth frame (in left lower corner) all the features are disappeared and blue object (with different scale) treated as different object. This is the limitation of the algorithm. It only detects features in first frame and if object disappears from its sight it fails to track. As a result Lucas Kanade optical flow algorithm fails in object permanance. Algorithm also cannot detect optical flow in green object that entered in second frame since the algorithm did not detect its features in first frame. From figure 17, it can be seen that HCVFlow does not have any problem with object permanance and first frame occurence. It even tracks shadow of the object. For better understanding, original video should be seen *“./experiment\_videos/ second\_case /second\_case\_hcvflow.mp4”.*

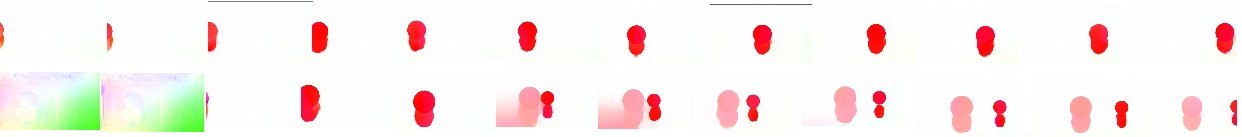


Figure 18 Second Case HCVFlow Results

### Third Case

In this case, HCVFlow algorithm is failed. Video of this case contains quick change in scene and also blurred with Gaussian(27x27 kernel). It can be seen under *“./experiment\_videos/ third\_case /third\_case.mp4”.*

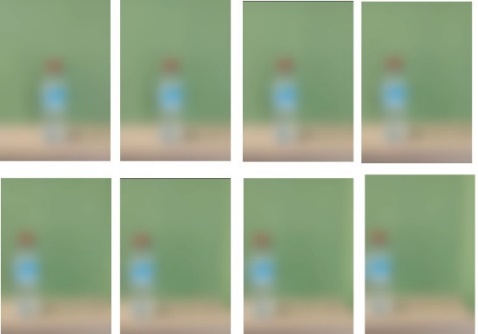


Figure 19 Third Case Frames

Even though this is an extreme case, HCVFlow could not succeed in this scenario. It returns extremely noisy output which does not indicate any flow. One can argue that the reason for failure is small number of frames. However HCVFlow is trained end to end with source target pairs. The real reason for failure is the fact that HCVFlow is not robust to noise because **blur corrupts feature representation**. It is common fact that when training data of model is not representative of this scenario(blurry images) CNN struggles creating feature representations accurately[8].



Figure 20 Third Case HCVFlow Results

### Conclusion

Throughout this work, two model of optical flow is studied: Lucas Kanade algorithm, HCVFlow model. As can be seen from experiments, even though Lucas-Kanade method is robust and works in many cases, it has certain problems like object permanance and feature dependence. However HCVFlow is more robust and works better in the cases where Lucas Kanade model fails thanks to modern approaches like cost volume, CNN based feature detection. Moreover, HCVFlow is also memory efficient model in comparison to contemporary optical flow models. Video inference was not supported by original code in github for HCVFlow so I developed. I also created a case for HCVFlow to fail. Figure 1-6 are taken from [7]. Figure 7,10,11 are taken from 6. Figure 8,9 13-20 are created by me.

### References

[1] Lucas, B. D., & Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. \*Proceedings of the Imaging Understanding Workshop\*, 121–130.

[2] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. 2015. Flownet: Learning optical flow with convolutional networks. In Int. Conf. Comput. Vis. 2758–2766

[3] Zachary Teed and Jia Deng. 2020. Raft: Recurrent all-pairs field transforms for optical flow. In Eur. Conf. Comput. Vis. Springer, 402–419

[4] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. 2017. Flownet 2.0: Evolution of optical flow estimation with deep networks. In IEEE Conf. Comput. Vis. Pattern Recog. 2462–2470

[5] Haofei Xu, Jiaolong Yang, Jianfei Cai, Juyong Zhang, and Xin Tong. 2021. Highresolution optical flow from 1d attention and correlation. In Int. Conf. Comput. Vis. 10498–10507.

[6] Zhao, Y., Xu, G., & Wu, G. (2024, September 6). Hybrid Cost Volume for memory-efficient optical flow. arXiv. <https://arxiv.org/abs/2409.04243>

[7] <https://www.cs.cmu.edu/~16385/s15/lectures/Lecture21.pdf>

[8] Igor Vasiljevic, Ayan Chakrabarti & Gregory Shakhnarovich (2016) *Examining the Impact of Blur on Recognition by Convolutional Networks*