Comparative Analysis of Optical Flow Techniques: Classical Computer Vision vs Deep Learning Approach

Arun Madhusudhanan* Francis Jacob Kalliath* Hardik Devrangadi* Tejaswini Dilip Deore*

***Abstract*—Optical flow estimation has found applications in various computer vision applications like object detection and tracking, movement detection, robot navigation and visual odometry. Two major approaches for estimating optical flow are classical methods and deep learning-based methods. Classical methods, such as the Lucas-Kanade, Horn-Schunck, and Farneback techniques, rely on well-established mathematical principles to compute flow estimates. Lucas-Kanade and Horn-Schunk methods calculate sparse optical flow based on the features and Franeback technique estimates dense optical flow. Deep learning methods, such as FlowNet and FlowNet 2.0, leverage the power of artificial neural networks to generate dense optical flow information from raw image data. This paper presents a comprehensive study comparing the performance of both the classical and deep learning approaches for estimating dense optical flow. We used the Farneback method as a representative of classical techniques and FlowNet 2.0 as a representative of deep learning-based methods. Our experimental results highlight performance comparison of both the methods on a defined dataset using appropriate metrics - L1 error, Average end point error and Average angular error. The results show that FlowNet 2.0 provides significantly better results than Farneback Algorithm.**

***Index Terms*—Optical Flow, Computer Vision, FlowNet, Lucas-Kanade, Farneback**

I. Introduction

Optical flow estimation is a fundamental problem in computer vision, as it enables the extraction of motion information from video sequences. It has numerous applications in areas such as object tracking, action recognition, video stabilization, autonomous navigation, and human-computer interaction. Given the critical importance of optical flow estimation, significant research efforts have been devoted to developing accurate and efficient methods for estimating optical flow. There are two main types of optical flow: sparse optical flow and dense optical flow.

Sparse optical flow focuses on a limited number of feature points within an image sequence. It involves tracking a select set of points, usually high-contrast or distinctive features, such as corners or edges. The main advantage of sparse optical flow is its computational efficiency, as it requires less processing power and time compared to dense optical flow, but it does not provide as much information about motion of the entire image. Popular algorithms for estimating sparse optical flow include Lucas-Kanade and Shi-Tomasi methods.

Dense optical flow computes motion vectors for every pixel in the image sequence, providing a more comprehensive representation of the apparent motion in the scene. This approach offers a detailed understanding of the motion, but at the cost of increased computational complexity and processing time. Popular algorithms for estimating dense optical flow include classical methods like Horn-Schunck, Farnebäck, and deep learning-based methods like FlowNet and RAFT.

Optical flow estimation techniques can be categorized into two main approaches: classical methods and deep learning-based methods. Classical methods such as Lucas-Kanade, Horn-Schunck, and Farneback techniques, are built upon well-established mathematical principles and optimization techniques. These methods typically rely on assumptions such as unvarying brightness, spatial smoothness, and temporal coherence to compute flow estimates. Although classical methods have demonstrated their effectiveness in many applications, they may encounter difficulties in handling complex and non-linear motion patterns and large displacements.

Deep learning-based methods, on the other hand, leverage the power of artificial neural networks to learn features, patterns and relationship between these features and patterns from raw image data to optical flow information. These methods, such as FlowNet, FlowNet 2.0 and RAFT [1], have emerged as promising alternatives to classical techniques, offering improved accuracy and robustness in challenging conditions.

We will be focusing on dense optical flow estimation techniques, which provide motion information for all points in the image and can result in a more complete representation of the overall motion in the scene. In this paper, we present a comprehensive study comparing the performance of classical and deep learning approaches in dense optical flow estimation. We implement the Farneback method as a representative of classical techniques and FlowNet 2.0 as a representative of deep learning-based methods. We use the MPI Sintel flow dataset [2] to generate optical flow outputs using both techniques and evaluate their performance using appropriate metrics such as L1 error, Average endpoint error, and Average Angular error. Additionally, we conduct an experiment to track facial motion using optical flow, and we compare the performance of Farneback and FlowNet 2.0 using a suitable metric, the percentage of bounding box overlap. The illustration of project scope is shown in Figure 1.

II. Related Work

Optical flow estimation is a constantly evolving field. There are various papers that have made substantial contributions in

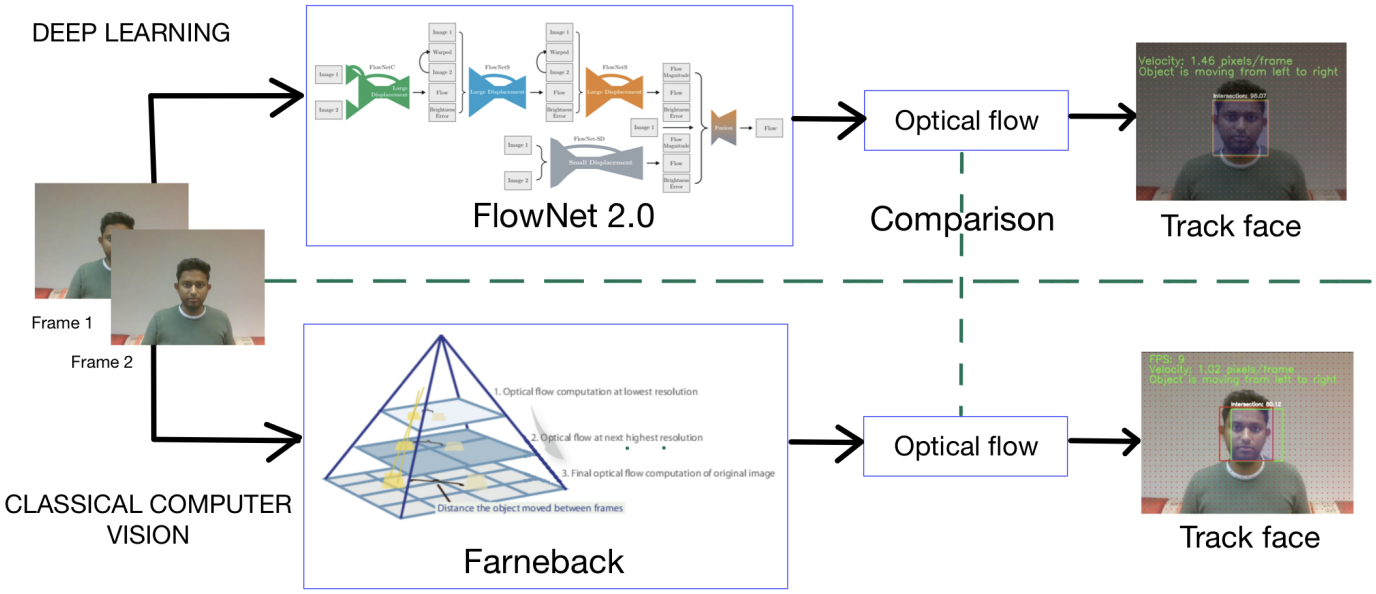

Fig. 1: An illustration of the proposed scope of the project. The frames are given as input for FlowNet 2.0 and Farneback algorithm. The optical flow outputs are used for performance comparison using metrics such as L1 error, Average End Point Error and Average Angular Error. The optical flow outputs are used for tracking face and performance will be compared using the percent overlap of the bounding box predicted by optical flow to that predicted Harr-Cascade classifier method.

the Optical Flow concepts. Some of the research works that have highly influenced our project are: Farneback Algorithm [3], FlowNet [4] and FlowNet 2.0 [5]

The Farneback algorithm uses a multi-scale approach to estimate the optical flow at different levels of resolution, or scales, by constructing an image pyramid where each level has a lower resolution than the previous level. At each level of the pyramid, the algorithm computes the optical flow field by tracking the motion of each pixel between two consecutive frames. This is achieved by using a polynomial expansion to approximate the image intensity around each pixel and then solving an equation to estimate the displacement of the pixel between the two frames. Once the optical flow fields are computed at each level of the pyramid, they are combined to create the final optical flow field by taking into account the motion estimated at each level of the pyramid. The result is a dense optical flow field that describes the apparent motion of objects in the image.

FlowNet is a convolutional neural network (CNN) model designed to estimate optical flow. The authors presented two architectures for optical flow estimation, FlowNetSimple and FlowNetCorr. Both architectures, as shown in figure 2, are end-to-end learning methods, meaning that they learn to estimate optical flow directly from images, without the need for hand-crafted features. FlowNetSimple is a simple architecture that stacks two consecutive input images together and passes them through a convolutional neural network. The network learns to extract features from the images and use these features to estimate the optical flow. FlowNetCorr is a more complex architecture that first generates separate representations of the two input images. These representations are then merged in a ”correlation layer” to learn a higher-level representation. The network then uses this higher-level representation to estimate the optical flow. The correlation layer is utilized to make comparisons between two feature maps by multiplying patches. To be more precise, it takes two feature maps, f1 and f2, each with dimensions w, h, and c, representing width, height, and number of channels. The correlation between two patches centered at x1 in the first map and x2 in the second map is defined as shown in equation 1.

$$c\left(\mathbf{x}_1, \mathbf{x}_2\right) = \sum_{\mathbf{o} \in [-k,k] \times [-k,k]} \left\langle \mathbf{f}_1\left(\mathbf{x}_1 + \mathbf{o}\right), \mathbf{f}_2\left(\mathbf{x}_2 + \mathbf{o}\right) \right\rangle \quad (1)$$

where, the center of the first and second maps are denoted as x1 and x2, respectively, with a square patch of size K = 2k+1 centered around them. The authors limit the maximum displacement for computation purposes. Specifically, for each x1 location, the range of x2 is restricted by computing correlations in a neighborhood of size D = 2d+1, where d is the maximum displacement specified. The resulting output has dimensions (w*h*D^2). The authors then concatenate the feature map extracted from f1 using a convolution layer with the output.

Both FlowNetSimple and FlowNetCorr have refinement steps to increase resolution during upsampling. This is done by upsampling the coarse flow prediction and then concatenating it with the corresponding feature maps. The resulting feature maps are then passed through another convolutional layer to refine the flow prediction.

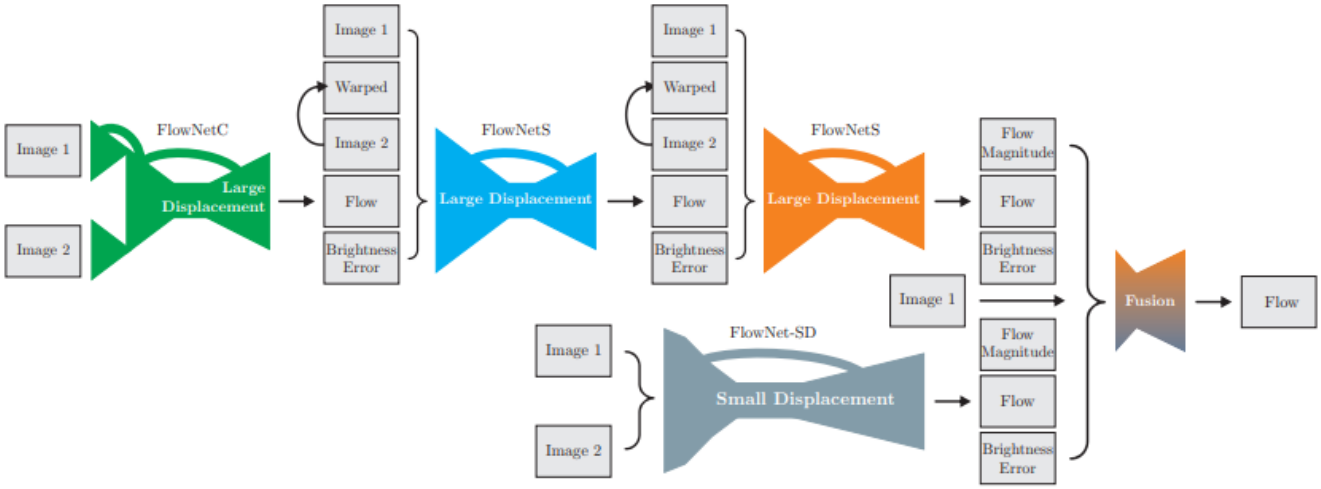

Fig. 2: Diagram of the complete architecture of FlowNet 2.0. For computing large displacement optical flow, multiple FlowNets are combined, with concatenated inputs shown within braces. The brightness error is determined as the difference between the first and second images, warped using the estimated flow. To effectively handle small displacements, smaller strides and convolutions between upconvolutions are added to the FlowNetS architecture. The final estimate is obtained using a small fusion network.

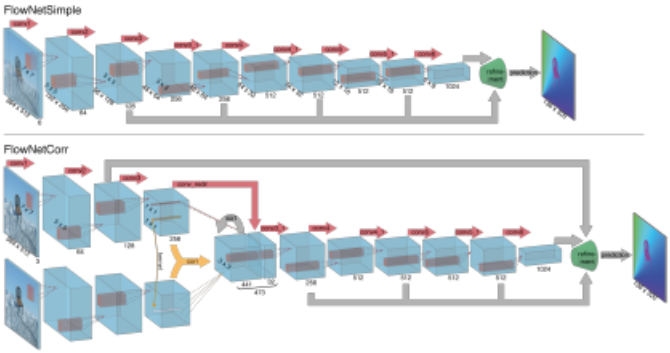

Fig. 3: The two FlowNet network architectures: FlowNetSimple on top and FlowNetCorr on bottom.

FlowNet 2.0 is an improved version of FlowNet 1.0. The main architecture of FlowNet 2.0 is shown in figure 2, and the paper makes four main contributions: (1) the schedule of presenting data during training is important, (2) a stacked architecture is proposed, (3) a sub-network specializing in small motions is introduced, and (4) a fusion architecture is proposed. The model uses a two-stage training process, where in the first stage, it is trained on a simpler dataset, such as FlyingChairs, to learn basic features of optical flow estimation. In the second stage, the model is fine-tuned on a more complex dataset, such as Sintel, to learn more sophisticated features. FlowNet 2.0 also uses an iterative architecture of multiple networks, where the output of one network is used as the input to the next network. This enables the model to learn more complex relationships between the two input images, and to refine the estimated optical flow field with each iteration. To improve the accuracy of the model for small displacements, FlowNet 2.0 uses various techniques. Firstly, it modifies the network architecture to include more layers that can capture fine-grained details in the images. Secondly, it fuses outputs from networks trained on different displacement datasets to obtain a more robust and accurate estimation of optical flow.

In our project, we have used this Haar-Cascade pre-trained classifier [6] to detect a face in a video frame.This method uses a cascade of boosted Haar-like features to detect objects in images. The two main steps of their methods are: feature selection and cascaded classification. In feature selection, they use AdaBoost to select the most effective Haar-like features for detecting the target object. In cascaded classification, a series of classifiers detect different parts of the object, with classifiers that reject non-target objects placed at the beginning of the cascade. The paper shows that their method could detect faces in real time with high accuracy. It is one of the simplest, most efficient, and most effective algorithms for detecting a wide variety of objects.

III. Experiments

In this section, we provide an explanation of the various experiments we have done in our project to compare the performance of Farneback and FlowNet 2.0.

A. Performance on MPI Sintel Dataset

The MPI Sintel dataset [2] is a large, high-quality dataset of rendered artificial scenes with realistic image properties. It is derived from the open-source 3D animated short film Sintel, and it contains two versions: the Final version, which includes motion blur and atmospheric effects, and the Clean version, which does not. Sintel is one of largest dataset (1,041 images

from 23 categories) available for optical flow evaluation, and it provides dense ground truth for small and large displacement magnitudes. The dataset is challenging due to its complex motion patterns, occlusions and different range of displacements which make it an ideal benchmark for evaluating the performance of optical flow algorithms. We used a derived dataset for training FlowNet 2.0. The distribution of data used for training, validation and testing are

- Training data (Sintel Final) : 23 categories, total : 744
- Validation data (Sintel Clean) : 23 categories, total : 744
- Testing data (Sintel Final) : 23 categories, total : 320

The training is completely done on Google Colab. The model was trained for 60 epochs to not exceed the user limitations of Google Colab. The parameters used for training are listed in table I. The model considers Average End Point Error (EPE) as loss function. Validation on the dataset was done during every 5 epochs. The results for L1 error and Average End Point Error for training and validation are shown the figure 4 and 5 . As shown, the losses were not reducing significantly after 60 epochs. Hence we decided to use the pre-trained weights provided by the authors of FlowNet 2.0 for our further experiment. The parmeters used for training by by the authors of FlowNet 2.0 are listed in table I.

Model	Trained From Scratch	Pre-trained model
No of epochs:	60	10000
Loss :	Average End Point Error	
Batch Size:	8	
Optimizer:	Adam	

TABLE I: Parameters used for training

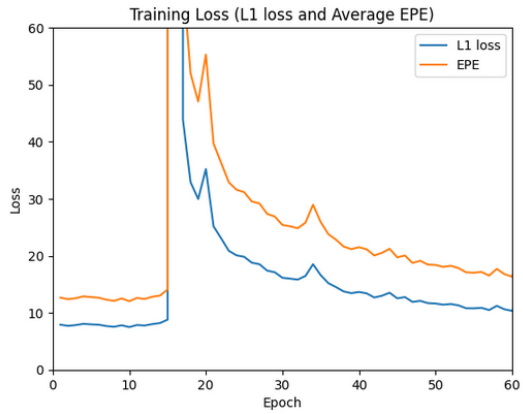

Fig. 4: Training loss

The optical flows on testing data of derived dataset were generated using the test dataset by both Farneback algorithm and FlowNet 2.0 method. To compare the performance of the optical flow outputs from Farneback and FlowNet 2.0 we used the metrics : L1 error, Average End Point Error (EPE), Average Angular Error (AAE). L1 error is the average of the absolute differences between the estimated and ground truth optical flow vectors at each pixel. Average End Point Error is the

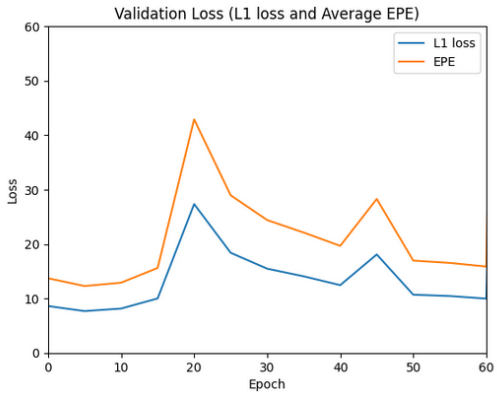

Fig. 5: Validation loss

average of the Euclidean distances between the estimated and the ground truth optical flow at each pixel. Average Angular Error is the average of the angles between the estimated and the ground truth optical flow at each pixel.

B. Implementation

The input video that we use to generate the optical flow for, is first processed before giving as an input to FlowNet 2.0 and the Farneback Algorithm. We apply a Gaussian Blur so that the background noise due to low light is eliminated. This prevents any possibilities of erroneous results in the obtained optical flow. After preprocessing, the video is given as an input to each of the algorithms.

1) Deep Learning Optical Flow Processing: The optical flow field is generated for all frames in the input video. The model then returns a series of (.flo) files, that holds the optical flow data between each of the successive frames. These flow files are represented as a grid of arrows which show the change in direction and magnitude as seen in Figure 6. This gives us an easy visual representation on how to understand the flow field. This grid is then overlaid on the original video so that the change in the optical flow with respect to the video can be easily visualized.

Fig. 6: Image showing the optical flow field represented as a grid of arrows

2) Classical Computer Vision Flow Processing: The optical flow field is generated for all the frames in the input video or live stream from webcam. The flow obtained from the

Farneback algorithm is then represented as a grid of arrows the same way as discussed previously. This flow is appended to the frame obtained from the webcam or video.

3) Tracking Face: In the first frame of the video, the face is detected using the Haar-Cascade pre-trained classifier, and a bounding box around the detected face is displayed. This bounding box is then considered as our Region of Interest (ROI) where all calculations and estimations are performed. The mean flow inside the ROI for every frame is calculated, and the ROI coordinates are updated based on this mean flow. This results in the eventual tracking of the object over successive frames, in this case, the detected face. Based on the optical flow field, the ROI changes, and tracks the movement of the face.

In the ROI, the velocity of the movement and the direction of movement sideways of the object is also calculated based on the mean flow and is displayed on the left corner of the image. Figure 7 shows the face being tracked and the velocity and movement direction on the output image.

Fig. 7: Screenshot showing the face being tracked, along with the velocity and movement direction

Another addition that was implemented was the addition of a User-Interface (UI) to make the output window more user friendly. A button "Reset ROI" and a step size trackbar was added. Since the bounding box tracks the face only on the basis of the optical flow, it is not always accurate and due to this inaccuracy, the box loses track of the face. When the button is clicked, the optical flow tracked box resets to the detected face, and thus completely overlaps with the ground truth. The trackbar which varies the step size can be varied from 10 to 64. This step size changes the number of flow points plotted on the output window. When the step size is 10, many more flow points are visualized then when the step size is 64. A screenshot of this UI is shown in Figure 8. This increases the amount of data used to calculate the mean flow and leads to better tracking, however, the FPS decreases due to the increased number of computations. By default, the step size is set to 16.

Fig. 8: Screenshot of the UI showing the Reset ROI button and the trackbar

IV. Results and Discussions

In this section, we discuss the results of the performance comparison between the optical flows and face tracking implemetation.

A. Performance on MPI Sintel Dataset

A few examples from the MPI Sintel test dataset with its ground truth and optical flow outputs from Farneback and FlowNet 2.0 is provided in figure 9. As expected, the FlowNet 2.0 algorithm is giving better results compared to the Farneback algorithm. There are white patches in the optical flow outputs from Farneback. To evaluate the performance of optical flow, we used the metrics L1 error, Average End Point Error, and Average Angular Error. The values are listed in the table III. The results from FlowNet 2.0 is significantly better compared to Farneback Algorithm.

The better performance of FlowNet 2.0 over Farneback might be due to a number of reasons . In regions with constant intensity, the Farneback algorithm may not be able to detect the variation in color and, therefore, fail to estimate the optical flow correctly. Occlusions can disrupt the continuity of the flow and make it difficult for the Farneback algorithm to estimate the flow vectors correctly. Large motion can also disrupt the continuity of the flow and make it difficult for the algorithm to estimate the flow vectors correctly. In low-texture regions, there is not enough information for the algorithm to accurately estimate the optical flow. This can result in white patches in the output. FlowNet 2.0 is trained on variety of datasets to perform better in all of the cases.

B. Implementation

Apart from the bounding box which is the ROI, there is another bounding box also present, which uses the Haar-Cascade classifier to detect a face in every frame. This box is considered as the ground truth, as it always tracks the face in every frame. Ideally, both these boxes have to overlap, which proves that the optical flow based tracking is always correct. However, since this is not possible, a performance metric is used.

The intersection performance metric is used to calculate the accuracy of optical flow based face tracking. The percentage

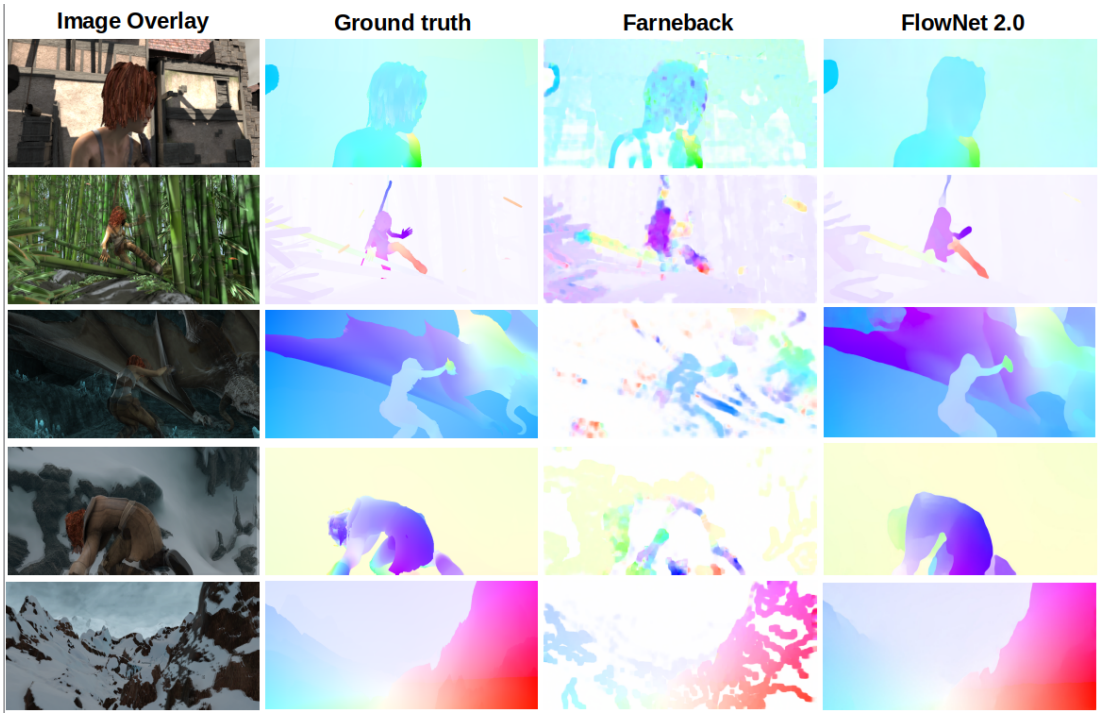

Fig. 9: Examples of flow fields from different methods estimated on MPI-Sintel. FlowNet 2.0 performs much better than Farneback and is able to extract fine details.

Frame Number	Farneback (% overlap)	FlowNet 2 (% overlap)
0	100%	100%
100	96.05%	95.39%
200	97.31%	94.07%
300	89.11%	99.33%
400	75.77%	93.01%
500	82.24%	89.15%
600	69.67%	100%
700	74.19%	99.33%
800	83.42%	98.70%
900	93.28%	95.99%
1000	90.47%	95.85%

TABLE II: Comparison of percentage overlap of Farneback and FlowNet 2 for the first 1000 frames

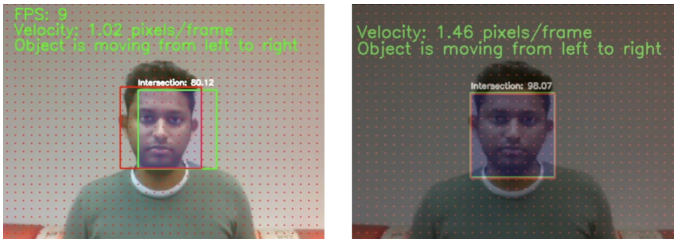

Fig. 10: Comparison of (i) Classical Computer Vision Overlap Percentage and (ii) Deep Learning Overlap Percentage

overlap between the two bounding boxes is calculated and displayed on the output window. Figure 10 shows the face being tracked and the overlap percentage for the same frame in both approaches. It can be seen that the deep learning based optical flow tracking is much more accurate and has a high intersection/overlap percentage, which can also be seen in Table II.

V. Summary

In this project, we were able to perform a comprehensive analysis of Optical Flow using Classical Computer Vision with the Farneback Algorithm and Deep Learning Model with FlowNet 2.0. Both algorithms were compared using a variety of performance metrics - L1 error, Average Endpoint Error and Average Angular Error. Additionally we were able to visualize the flows obtained from both approaches as a grid of arrows, and performed face tracking using a Haar-Cascade Classifier. Additionally, the optical flow field was analysed; the velocity of the movement, and sideways direction of movement was also calculated. A user-friendly UI was created for the Classical Computer Vision approach, to visualize the optical flow with different step sizes, with the ability to reset the ROI at the click of a button.

The optical flow field generated by FlowNet 2.0 performed much better at tracking over the flow generated by Farneback in all the tests across all performance metrics. Due to lack of texture dependencies, large displacement, dependency on pixel color values, the performance of the optical flow for tracking by the Farneback algorithm was subpar. Since FlowNet 2.0 has been trained on various datasets covering multiple scenarios with varying picture characteristics, it performs much better across all performance metrics.

Category	L1 Error		EPE		AAE	
	Farneback	**FlowNet 2.0**	**Farneback**	**FlowNet 2.0**	**Farneback**	**FlowNet 2.0**
Alley_1	0.6416	0.0924	23.6546	0.1476	31.0392	2.2396
Alley_2	1.5901	0.1524	56.9127	0.2445	27.4461	2.2787
Ambush_2	38.9716	1.2146	882.3624	2.2024	78.4506	2.6711
Ambush_4	14.7739	1.5013	426.9432	2.5994	74.6500	3.4777
Ambush_5	26.4725	2.3284	689.7455	3.8740	75.3315	4.3254
Ambush_6	34.6250	2.0365	865.6840	3.3719	73.8862	5.1622
Ambush_7	5.3257	1.7847	164.3199	2.9512	73.0746	5.0952
Bamboo_1	0.5527	1.6738	24.9192	2.7517	27.2078	5.2876
Bamboo_2	2.8247	1.5164	117.2136	2.4906	61.4336	5.6894
Bandage_1	0.5896	1.4095	23.6486	2.3147	36.0795	6.2677
Bandage_2	0.2237	1.3756	8.7772	2.2529	34.6466	6.1733
Cave_2	14.0825	1.3351	336.8272	2.1813	49.0335	6.0147
Cave_4	10.7466	1.4874	300.5732	2.4141	59.0035	6.4774
Market_2	0.1977	1.4268	9.4361	2.3152	53.6362	7.0750
Market_5	13.9899	1.4077	386.5746	2.2801	71.4026	7.0595
Market_6	5.7798	1.4447	186.9603	2.3299	61.3453	7.0261
Mountain_1	1.4578	1.3892	44.0406	2.2385	46.2691	6.8608
Shaman_2	0.1783	1.3163	6.2656	2.1207	27.7953	6.7706
Shaman_3	1.8477	1.2461	45.7262	2.0076	62.2367	6.7328
Sleeping_1	0.9926	1.1840	29.7839	1.9071	38.7935	6.5560
Sleeping_2	0.5207	1.1261	18.2515	1.8139	24.3843	6.3979
Temple_2	1.3428	1.0776	45.7841	1.7365	32.9923	6.2335
Temple_3	37.5473	1.1768	905.8085	1.8953	79.8069	6.1366
Mean	9.3598	1.3349	243.4875	2.1931	52.1715	5.5656

TABLE III: Performance comparison on public benchmarks. AEE: Average Endpoint Error. AAE: Average Angular Error

REFERENCES

[1] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 2020, pp. 402–419.

[2] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *European Conf. on Computer Vision (ECCV)*, ser. Part IV, LNCS 7577, A. Fitzgibbon et al. (Eds.), Ed. Springer-Verlag, Oct. 2012, pp. 611–625.

[3] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer, 2003, pp. 363–370.

[4] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2758–2766.

[5] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2462–2470.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, vol. 1. Ieee, 2001, pp. I–I.