Learning Optical Flow from Real Robot Data

Parth Shah (pshah9@stanford.edu)

***Abstract*—This project presents a method for teaching robot arms optical flow. By extending recent works on depth based, probabilistic robot tracking, a novel real-world optical flow dataset with dense ground truth annotation is generated. A real, robotic dataset boasts the inclusion of phenomena that synthetic datasets cannot model. With this data a generic convolutional neural network is implemented to predict optical flow. Current performance, with limited computational resources available, struggles to match the performance of classical techniques (Lucas & Kanade, Horn & Schunck, etc). Positive long-term results however would have strong implications in the field of robotic manipulation and grasping – allowing robots to generate and learn optical flow autonomously while also developing a methodology for understanding dynamic scenes.**

Keywords-optical flow, machine learning, robotics, deep learning, computer vision, convolutional neural nets

I. INTRODUCTION

Robotic grasping and manipulation is a challenging problem. Most approaches today rely heavily on models and only work with rigid objects in static or friendly settings. If a robot is to successfully grasp an object it has never seen before in a dynamic scene, it must visually understand its environment very well.

Recent advances in the field of robotics have now made it possible to simultaneously track the robot arm and object during manipulation [1]. The technique is very robust and can handle accelerated object and head motions as well as occlusions for extended periods. A reactive manipulation system can utilize this to handle uncertainty and dynamics in the environment.

To extend this technique to grasp and manipulate unknown objects, the system will need to track and understand the dynamics of the unmodeled object as well. Optical flow is a visual characteristic that conveys this desired information. Traditional techniques such as Lucas-Kanade [2] and Horn-Schunck [6] calculate optical flow with high accuracy for low motion scenes but perform poorly once occlusions, shadows, and large motions are introduced.

With the rise of machine learning, neural network architectures have been implemented to tackle these challenging scenarios. FlowNet [3] utilizes a layered neural network architecture along with a large synthetic data set to predict optical flow. The network can be fine-tuned to handle certain phenomena, but the overall performance is similar to the traditional algorithms.

In order to train this desired neural network, a large, real dataset is needed. All real datasets, such as KITTI and Middlebury, are too small to properly train a network and the large ones, like MPI-Sintel or Flying Chairs, are synthetically generated. Even though these synthetic techniques are pretty advanced, they do not model all the intricacies present in the real world visual experience.

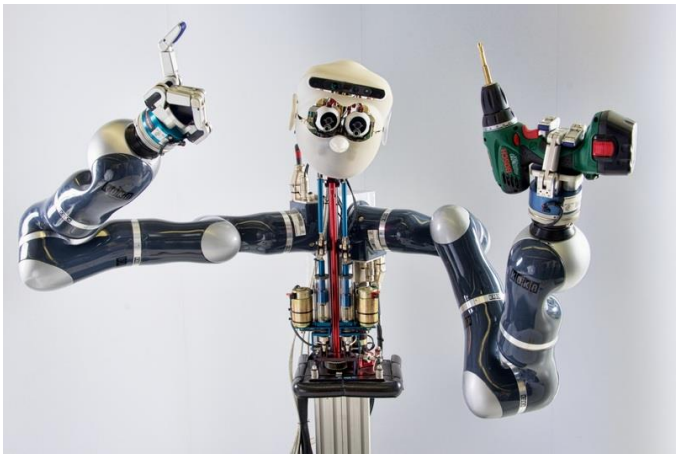

Figure 1. Apollo - the dual-arm robotic platform utilized in this project.

In this project, I will generate a new dataset from a robot arm to train an end-to-end neural network for optical flow prediction. This end-to-end, deep convolutional neural network will receive the two RGB images as input and will output a matrix of predicted flow values. Positive results will imply that a robot arm will be able to teach itself optical flow locally. By recording its own actions, the robot can use its own arms and visual tracking to ground truth annotate real-data autonomously. This data can then be used to train the presented network, thus allowing the robot arm teach itself optical flow.

II. RELATED WORKS

A. Datasets

First I will present the prominent optical flow datasets with ground truth values.

KITTI [4] – This dataset consists of 194 images pairs. The ground truth annotation for these images are not very dense because the dataset was created by recording a real world scene with a camera and a 3D laser scanner. As a result, some motion, such as motion of distant objects like the sky could not be labeled.

Sintel [5] – This dataset consists of 1041 image pairs that were rendered from artificial scenes. As a result, the ground truth annotation for these scenes are dense. The displacements vary in magnitude and additional features such as fog and motion blur have been added into the final version of the dataset. This is the dataset most optical flow algorithms are benchmarked against.

Flying Chairs [3] – This dataset consists of 18000 image pairs. The dataset was rendered by randomly selecting and rendering chairs in an image. The background was a randomly selected Flickr image. Random affine transformations were applied to each object in the image and the background. The ground truth annotation for this dataset is dense. This was the dataset that FlowNet utilized to train their network.

The downside of these datasets is that they are either not densely annotated or they are artificially generated. As noted previously, I believe that if the network is trained on real data then its ability to predict optical flow will improve. This motivated the generation of a densely annotated, optical flow dataset from a real robotic setup.

B. Optical Flow Calculation Techniques

Here I will discuss a variety of techniques that can be used to calculated optical flow given two sequential images.

Horn-Schunck [6] – This is a classic approach to calculate the optical flow between two images. It is a differential method for approximating the desired value by utilizing partial derivatives. The method poses the problem as an energy function, Equation 1, and tries to minimize it.

$$E = \iint [I_x u + I_y v + I_t]^2 + \alpha^2(||\text{grad}(u)||^2 + ||\text{grad}(v)||^2) \quad (1)$$

In the equation above I is the matrix of intensity values for a given image. Vectors u and v represent the two components of the optical flow values. The α is a regularization term and it influences the smoothness of the solution. Using the energy function, a system of equations is setup from the partial derivatives that express distortions in the flow. The output, or solution of this system, is the optical flow values.

This technique and other related difference approaches perform well on images with little motion, but struggle on images with large displacements, a phenomenon often present in real world scenarios. To improve performance, and reduce the number of hand tuned parameters, several groups have begun applying machine learning techniques to this task.

One specific machine learning technique that has dominated optical flow prediction is deep convolutional neural networks. Convolutional neural networks are a very popular approach for large-scale image related tasks. In specific, they have been successfully used in the past for image classification [7, 8], image segmentation [9], etc. Here are a couple works using convolutional neural networks for predicting optical flow:

FlowNet [3] – this is one of the first works to approach this task, and it does so with an end-to-end approach. The paper presents two architectures, and relies heavily on stacked convolutional layers. The first architecture, FlowNetSimple, is a series of convolutional layers that first extracts features from the input images before refining the coarse representation into a fine optical flow prediction matrix. The second architecture essentially mirrors the first one but processes the two images in two separate branches for the first couple convolutions, before combining them using a correlation layer for superior feature extraction. It is important to note that the FlyingChairs dataset was also presented along with the FlowNet paper.

FlowNet2 [10] – This paper followed up the original FlowNet paper with new network architectures that boasted better performances. In essesence, a majority of the architectures that were presented were large architectures that stacked the original 2 FlowNet architectures in different permutations. New architectures that focused on small displacements were also introduced. The paper boasted state of the art results while being orders of magnitudes faster.

FlowFieldsCNN [11] – This paper presents a state of the art, end-to-end technique for calculating optical flow. It differs from the FlowNet papers by not having stacked architectures, but rather multiple architectures in parallel that are all on different scales. The outputs are then scaled and filtered before being combined into one final prediction. The results are currently ranked third among all approaches, with or without learning, for predicting optical flow.

In each of these works, it is easy to observe that the network architectures become more and more complicated, while performance inches forward slowly. Given the robotics manipulation and grasping based motivation, I am not focused on developing a very complicated network that performs well on optical flow, but rather using a standard network architecture with the novel dataset that I have generated. As a result, I will implement a simple, layered architecture derived from FlowNet, but first we need to dive into deeper detail about this new, real robotics dataset.

III. REAL-DATASET

The authors in [1] have presented a technique for accurately estimating the end effector of the robot at all times. If the location of the end effector is known, it is possible to extract the pose of the entire robotic arm by simply using forward kinematics. Basic frame transformations can be applied to shift from the world frame to the camera reference frame.

Throughout this process the model for the robot arm is also known. Combining the tracking functionality in [1] with simple CPU rendering techniques it is possible to render the 3D mesh of the arm for every single image. The 3D mesh is projected back onto the camera plane to determine which pixel each vertex of the mesh corresponds to. Then using barycentric coordinates, the new location of the vertex is determined and the ground truth flow can be deduced.

The procedure above takes care of densely ground truth annotating a robot arm over a series of images, however the arm is not the only object in the scene. The scene often consists of a variety of other objects or humans moving in the background. Because no mesh models are provided for the other dynamic objects, the scene needs to be segmented so the background can be removed. Similar to the Flying Chairs dataset a random background was inserted. This background was randomly selected from 26 Flickr images that were under the query for “cities”. For each frame the background was randomly transformed before being added. This generated known optical flow values for the background as well.

Of the 11 rosbags available from the robot data collected in [1], I have only utilized four of the bags – "arm", "sine-arm", "sine-both", and "static". These four bags have been selected for now because they offer a variety of different motion while not including any challenging properties such as occlusions or complete removal of the arm from the image frame. Together the four bags have generated 4300 image pairs with densely annotated ground truth values.

Figure 2. Two sequential images that compose an image pair. The background is randomly selected from Flickr.

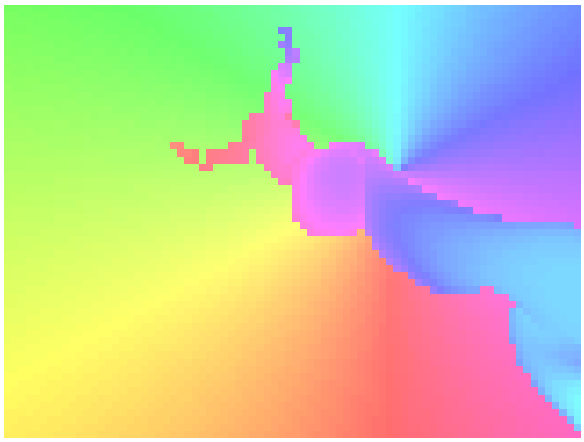

Figure 3. A visual representation of the optical flow between the two images. Each color maps to a vector with a certain direction and magnitude.

All of the 4300 image pairs were at a resolution of 480 by 640 pixels. Of the 4300 image pairs, 3350 image pairs were a part of the training set, while a 100 image pairs composed the validation set, and the remaining 850 image pairs were placed in the test set. It is important to note that the data is time series data, but I tried to remove all dependence from time, outside of the two images that constructed an image pair, by shuffling and augmenting the data. The augmentations involved cropping, flipping, and warping the raw data. Because the RGB images take on values from 0 to 255, the values are normalized to 1. The input features from this novel dataset are the normalized, augmented, shuffled real robot image pairs.

IV. Methods

A. *Baseline – Lucas & Kanade*

As discussed above, traditional techniques for calculating optical flow relied on partial derivatives and energy equations. Lucas-Kanade is a very simple approach that only relies on three partial derivatives, partials in the two direction and the partial in time.

$$I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n) \quad (2)$$

These partials are approximated over two sequential matrices, I, which represents the intensity channel from the input image. These derivatives are evaluated at each pixel, or point q_n. The two unknowns presents are the two V values, which represent the optical flow along the x and y axes. Given large enough input images, the equations can be organized into a system of equations that is overdetermined and can be solved using the least squares principle.

B. *Convolutional Neural Network*

A convolution is a kernel that sweeps across a matrix, applying a specific function that is represented by the weights in the kernel. A convolutional layer in machine learning applies this process over and over on the matrix input to produce the desired amount of output layers. Specifically, convolutions on images are good at learning and extracting features.

The end-to-end network I use for training first stacks convolutional layers to take the input images to a coarse representation of the important features. Then using more convolutions and upconvolutions, the coarse representation is transformed into optical flow predictions. The architecture is a modified version of FlowNetSimple.

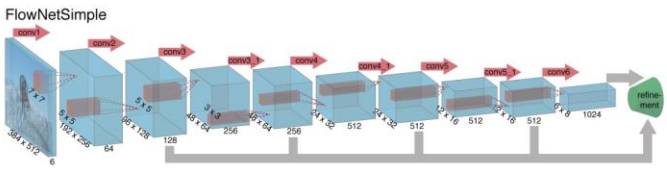

Figure 4. The FlowNetSimple architecture. [3]

- Layer 0: Stacked, augmented input images: 384x512x6
- Layer 1: Convolution with 64 filters, size 7x7, stride 2 with a leaky ReLU (alpha = 0.1)
- Layer 2: Convolution with 128 filters, size 5x5, stride 2 with a leaky ReLU (alpha = 0.1)
- Layer 3: Convolution with 256 filters, size 3x3, stride 2 with a leaky ReLU (alpha = 0.1)
- Layer 4: Convolution with 256 filters, size 3x3, stride 1 with a leaky ReLU (alpha = 0.1)
- Layer 5: Convolution with 512 filters, size 3x3, stride 2 with a leaky ReLU (alpha = 0.1)
- Layer 6: Convolution with 512 filters, size 3x3, stride 1 with a leaky ReLU (alpha = 0.1)
- Layer 7: Convolution with 512 filters, size 3x3, stride 2 with a leaky ReLU (alpha = 0.1)
- Layer 8: Convolution with 512 filters, size 3x3, stride 1 with a leaky ReLU (alpha = 0.1)
- Layer 9: Convolution with 1024 filters, size 3x3, stride 2 with a leaky ReLU (alpha = 0.1)
- Layer 10: Convolution with 2 filters, size 3x3, stride 1 with a bilinear image resize
- Layer 11: Upconvolution with 512 filters, size 3x3, stride 2
- Concat 1: Layer 10, 11, and 8.
- Layer 12: Convolution with 2 filters, size 3x3, stride 1 with a bilinear image resize
- Layer 13: Upconvolution with 512 filters, size 3x3, stride 2
- Concat 2: Layer 12, 13, and 6.
- Layer 14: Convolution with 2 filters, size 3x3, stride 1 with a bilinear image resize
- Layer 15: Upconvolution with 256 filters, size 3x3, stride 2
- Concat 3: Layer 14, 15, and 4.
- Layer 16: Convolution with 2 filters, size 3x3, stride 1 with a bilinear image resize
- Layer 17: Upconvolution with 128 filters, size 3x3, stride 2
- Concat 4: Layer 16, 17, and 2.
- Layer 18: Convolution with 2 filters, size 3x3, stride 1 with a bilinear image resize
- Layer 19: Upconvolution with 64 filters, size 3x3, stride 2
- Concat 5: Layer 18, 19, and 1.
- Layer 20: Convolution with2 filters, size 3x3, stride 1 with a bilinear image resize
- Result: Predicted optical flow matrix 384x512x2

It is important to note that forward pooling is not used in this network architecture. Techniques such as max pooling select the most dominant feature from a patch and use that to be the value that proceeds onwards in the network. With the problem at hand, this is not the best method to use because the output, optical flow, does not specifically depend on the dominant features only, but rather a variety of small and large features. Instead, unpooling is used in the matrix resize step to further extend the feature map during the refinement portion of the network.

V. Experiments

A. *Baseline – Lucas & Kanade*

Trivially run the data through OpenCV implementation of the algorithm to produce baseline results.

B. *Convolutional Neural Networks*

The convolutional neural network was implemented from scratch using Tensorflow [12]. Using the Adam optimizer, it was trained for 200k steps, with a batchsize of 8 image pairs, a learning rate of 1e-4, and a L2 loss function. These hyperparameters were selected to be these values to mimic those that were used to train the original FlowNet architectures.

At the last minute, a PyTorch [13] implementation from GitHub was also forked, fixed, and trained on the FlyingChairs dataset to provide another benchmark to compare to.

Average endpoint error, AEE, is an important metric that is calculated by finding the Euclidean distance between predicted value and the ground truth and averaging it over all pixels present. Optical flow algorithms are benchmarked and compared using AEE as the most important comparison metric.

VI. Results

A. *Quantitative*

Results from the three experiments – baseline, tensorflow on real robot data, and pytorch on FlyingChairs – will be presented.

TABLE I. Result Summary

Technique	Training (AEE)	Test (AEE)
Lucas-Kanade	-	7.94
FlowNetS in PyTorch on FlyingChairs	1.91	2.212
FlowNetS variation in Tensorflow on real robot data*	1.87	14.86

Due to the lack of compute resources available, the FlowNetSimple variation that was implemented in tensorflow was only trained to completion (3 days worth of GPU time) on a small training set of 50 images pairs. The goal was to first see if the network would converge with smaller datasets while assessing for exploding gradients before transitioning to training on the full dataset. The figure below clearly illustrates that the network's training loss converged to a reasonable value.

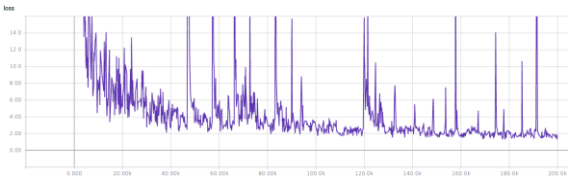

Figure 5. The loss plot of the network converging over 200k steps

As the quarter came to a close, my advisor notified me that an implementation of FlowNetS was available in PyTorch. The original paper had only provided an implementation in Caffe. I forked this implementation, fixed a couple bugs I found, and then tried to train it on the original FlyingChairs dataset to see if I could produce reasonable results. The training resulted in 2.2 AEE on the test set. This is comparable to the findings of the original paper. The figure below depicts how actual network converged after training for 225 epochs.

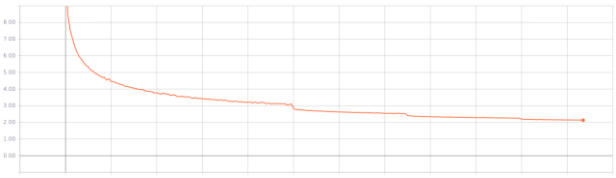

Figure 6. The converging loss plot for FlowNetS trained on FlyingChairs

B. *Qualitative*

To discuss the results in a more visual manner, Figure 7 and 8 present the ground truth and predicted optical flow values for an image pair.

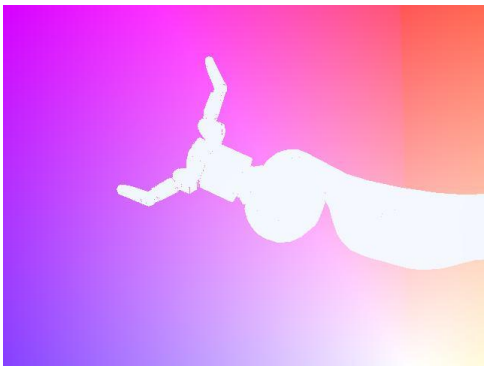

Figure 7. The visualization for the ground truth optical flow values for one set of images

Figure 8. The visualization for the predicted optical flow values for one set of images

The reason there is a discrepancy in between the two images is because of the cropping applied to the image. The original image, as displayed in Figure 7, is 480 by 640. The network is currently built to accept images that are 384 by 512. The random crop was not recorded during this experiment so it could not be trivially applied to the ground truth image during the visualization process. By observation though it is clear that the discrepancy between the two images is tiny. The final AEE for this set of images was 0.393.

VII. Discussion

From the experiments it is easy to see that the architecture that was presented and trained in this project with the new dataset does not perform nearly as well as either the baseline technique or the architectures that have been published. The main reason for this is a result of lack of computational resources which limited the training set size in the completed experiments. The one positive result that came from this completed experiment was that the network's loss values converged towards 0, indicating that the architecture was learning something. This however does not guarantee that the architecture will learn optical flow to the desired level when the entire training set is used. Increased variations present in a larger training set will hopefully reduce any overfitting that resulted in the terrible test set performance.

The other experiments related to the baseline methods and PyTorch implementation offered some successful results. Both experiments provided two values for comparison methods. They both also produced values in the expected range for the given techniques.

VIII. Future Work & Conclusion

Overall, this project led to an end-to-end architecture for predicting optical flow along with a novel real robotic dataset. Due to lack of computational resources, the network was only trained on a subset of this new dataset but it did show that it was able to converge to reasonable training loss values. However, this did not translate to the test set because of the limited size of the training dataset. With more compute and optimization of hyperparameters, a converging network will have a significant impact in the field of robotic grasping and manipulation. These results will suggest that a robotic arm will now be able to autonomously teach itself optical flow by extending the techniques the project was built upon.

The next step of course is taking the time to train on the entire training dataset. Once the resultant values are better than the baseline techniques and comparable to the other benchmarks (FlyingChairs & Sintel), more challenging phenomena like occlusions and shadows can be introduced. It will be interesting to see how resilient the network and technique is to these characteristics that aren't common in large synthetic datasets, aka FlyingChairs.

IX. Contributions & Acknowledgements

I was the only member in the project for 229, so I have done all the work for this project. I would like to acknowledge my advisor, Professor Jeannette Bohg, for her input and help on this project. I would also like to thank the Stanford Vision Lab for letting me using their machines overnight to generate my data and train my network.

REFERENCES

[1] Cristina Garcia Cifuentes and Jan Issac and Manuel Wthrich and Stefan Schaal and Jeannette Bohg. Probailistic Articulated Real-Time Tracking for Robot Manipulation. Robotics and Automation Letters. 2017.

[2] Lucas, B., and Kanade, I". 1981. An iterative image registration technique with an application to stereo vision. Pro~ DARPA lmage Understanding Workshop, pp. 121-130.

[3] Fischer, Philipp, et al. "Flownet: Learning optical flow with convolutional networks." *arXiv preprint arXiv:1504.06852* (2015).

[4] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. International Journal of Robotics Research (IJRR), 2013.

[5] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, ECCV, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012.

[6] B. K. P. Horn and B. G. Schunck. Determining optical flow. Artificial Intelligence, 17:185–203, 1981.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, pages 1106–1114, 2012.

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

[9] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[10] Ilg, Eddy, et al. "Flownet 2.0: Evolution of optical flow estimation with deep networks." *arXiv preprint arXiv:1612.01925* (2016).

[11] Bailer, Christian, Kiran Varanasi, and Didier Stricker. "Cnn-based patch matching for optical flow with thresholded hinge loss." *arXiv preprint arXiv:1607.08064* (2016).

[12] Abadi, Martín, et al. "TensorFlow: A System for Large-Scale Machine Learning." *OSDI*. Vol. 16. 2016.

[13] Paszke, Adam, et al. "PyTorch." (2017).