Attention based Visual Odometry

ECE-GY 7123: Deep Learning
Jagenath Hari, Navoday Borkar, Venkata Amith
New York University
jh7454@nyu.edu, nyb2055@nyu.edu, vp2201@nyu.edu
GitHub Link :https://github.com/navoday01/Attention-Based-Visual-Odometry

Abstract

In this project, we are developing a novel artificial neural network model that can be used to calculate visual odometry. The proposed model is a temporal-based attention neural network, the model takes in raw pixel and depth values from a camera and uses these inputs to generate feature vectors. The model then stores a history of the generated feature vectors, which allows it to take into account the temporal relationship between the current and past data points. The resulting output of the model is a prediction of the odometry value, which represents the movement of the camera or object. The proposed model has been tested in a variety of scenarios and has shown robust performance in unknown and cluttered environments.

1 Introduction

Visual odometry (VO) refers to the process of estimating the pose (position and orientation) of an agent (such as a robot or a vehicle) based on the input from a camera or a set of cameras. This information is essential for autonomous navigation of the agent in various environments, as it allows the agent to determine its position and orientation relative to a given reference frame.

Traditionally, VO has been computed using techniques such as optical flow, which rely on the detection and tracking of features in the images. However, these techniques can be prone to errors in featureless environments or when the features are too sparse, as there are not enough points to track.

In recent years, there has been an increasing trend towards the use of learning-based methods for VO. There are several different approaches to using neural networks for VO, including using RGB images alone, using RGB and depth maps, using RGB and motion information, and using RGB, depth, and temporal information. These methods can be trained end-to-end to learn the relationship between the input images and the agent's pose, or they can be used as a supplement to traditional methods

In this project, we built a novel attention-based neural network architecture for VO, which will use a sequence of RGB images and their depth maps as input. By using depth information in addition to color, the network may be able to better estimate the pose of the agent in a sparsely-featured environment.

2 Literature Survey

Early work on the learning-based VO is reviewed in this section, discussing various algorithms and their differences from others. There are mainly two types of algorithms in terms of the technique and framework adopted: used to supplement traditional pipelines or build an end-to-end architecture.

In (el.at. 2017) proposed DeepVO, a deep Recurrent Convolutional Neural Network(RCNN) based visual odometry algorithm. The geometric features of the environment are learnt using convolutional networks, and motion estimation and dependence between the different RGB frames are learnt using RNNs. They were the first to demonstrate VO problem can be addressed in an end-to-end fashion based on DL, i.e., directly estimating poses from raw RGB images. Neither prior knowledge nor parameter is needed for Architecture to compute odometry.

The NeuralBundler proposed by (Li, Ushiku, and Harada 2019) used unsupervised learning to develop a DL-based VO. The algorithm follows the traditional VO pipeline, but each component in the pipeline is replaced by a neural network. NeuralBundler uses temporal and spatial photometric loss as main supervision and generates a windowed pose graph consisting of multi-view 6DoF constraints. It was the first attempt was to combine deep learning-based VO with the classic graph optimization technique.

The Vision transform(ViT) proposed by (el.at. 2020) gave a state-of-the-art performance on several image recognition benchmarks. Recent research tried to incorporate the ViT into the deep VO pipeline, DPT-VO proposed by (et.al 2022) is one such algorithm. DPT uses the ViT as a backbone to compute the depth of the images and traditional algorithms for feature detection and motion estimation to compute the odometry.

The DeepAVO proposed by (Zhu and el.at 2021) is another algorithm which tries to incorporate spatial attention for computing visual odometry. The algorithm first computes the optical flow between two frames and then it is passes optical flow through convolutional network. The algorithm has four-branch network to learn the rotation and translation by leveraging Convolutional Neural Networks (CNNs) to focus on different quadrants of optical flow input. To enhance the ability of feature selection, we further introduce an effective channel-spatial attention mechanism

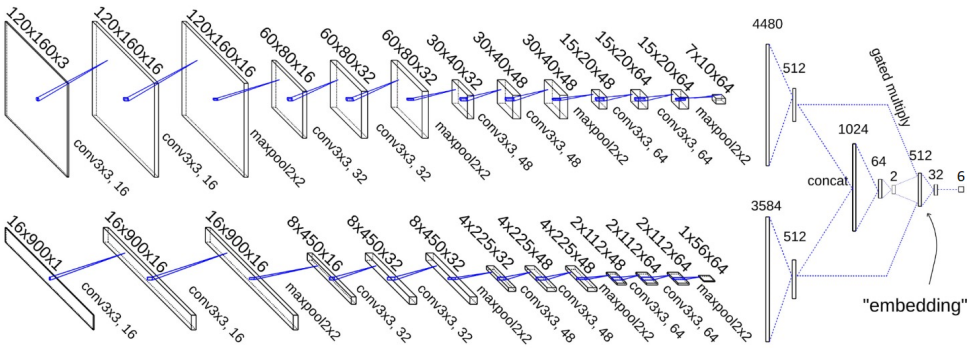


Figure 1: Network architecture

to force each branch to explicitly distill related information for specific Frame to Frame (F2F) motion estimation.

The prior and current research mostly focuses on the spatial attention of each frame. There has been very little research on soft temporal attention-based visual odometry algorithms. Though temporal attention has been used for navigation of UGV(Patel et al. 2017) but to our knowledge, it has not been used for VO problems.

3 Attention Network

The network used in the project has been inspired by the soft temporal architecture proposed by (Patel et al. 2017). The network structure combines past information through a soft attention mechanism.

The network shown in 2 has two parts: an encoding part that combines the camera and depth images through convolutional and fully connected layers to produce fixed-size embeddings and a decoding part that uses the past and current embeddings to calculate the translation and rotation between the previous and current frames. The encoding stage includes 8 convolutional layers and 1 fully connected layer for each type of data, with a rectified linear unit (ReLU) activation function between each layer. The final layer of each type of data produces a fixed-length (1 x 512) vector that balances the contribution from each sensor. These vectors are combined and passed through 2 fully connected layers to create gating coefficients. These coefficients are to used to weight the vectors, which are then added together to create a final vector called the embedding. The embedding is combined with past embeddings (based on the window size) to calculate the translation and rotation between the previous and current frames. The size of the window depends mainly on two things for this problem - The image and depth frames generated per second(FPS) by the camera and the speed at which the camera is being moved. If FPS of the camera is there needs to be a larger window size as the movement between the window size will be very small. Models with a window size of 5,15 and 25 frames were created, and it was found that for the available datasets 25 frames window generated accurate translations compared to other window sizes.

The decoding stage shown in 2 takes as input a sequence history that stores 25 past embeddings (equivalent to 1.5 seconds of feature embedding history) and the current embedding. The history of feature embeddings is flattened so that all features in all sequences contribute to determining the attention mask. The flattened vector is passed through a fully connected layer to create a vector with the same length as each of the embeddings (1 x 512) that summarizes past states. This vector is combined with the current embedding and passed through a fully connected layer with a softmax activation function to produce a temporal attention mask (1 x 25). The attention mask depends on both the past embeddings and the current embedding, giving greater emphasis to the current instance. The mask is multiplied by the sequence history (25 x 512) to generate the history contribution vector. This mechanism automatically evaluates the temporal history to determine which parts of the history are most relevant in producing the final output. Finally, the current embedding and history contribution vectors are combined and passed through 2 fully connected layers to calculate the translation and rotation between the frames.

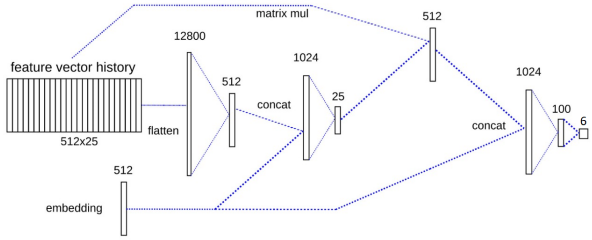


Figure 2: Temporal Attention Network

4 Datasets

This section talks about the datasets used for training and testing the proposed network. Section 4.1 covers the dataset used for training the network and Section 4.2 talks about the dataset we created for evaluating our network.

4.1 7Scenes

For training the network a dataset containing RGB images, Depth maps and camera poses had to be chosen for training. The famous 7Scenes dataset (el.at. 2013) was chosen containing RGB images ($480 \times 640 \times 3$), Depth maps with depth in millimeters(mm) ($480 \times 640 \times 1$) and camera poses in the form of transformation matrices (4×4).

The dataset was captured using a Kinect RGB-D camera at 640×480 resolution and KinectFusion system to obtain the 'ground truth' camera tracks. The camera calibration matrix was provided which had focal length(f_x, f_y) as $(585, 585)$ and principal point(c_x, c_y) as $(320, 240)$ though these were not used.

Additionally, the dataset was created on different environments or scenes named as Chess, Fire, Heads, Office, Pumpkin, RedKitchen and Stairs, having 7 different scenes hence the name "7Scenes".

Each of the scenes have different sequences containing $500 - 1000$ frames each.

4.2 NYU sparse dataset

To evaluate the robustness of our network we decided to create a dataset of an environment having sparse features.

To create the RGB Images, Depth Maps and camera poses we used the ZED2 camera [1]. The ZED2 Camera was set to VGA mode producing RGB Images($376 \times 672 \times 3$), Depth Maps($376 \times 672 \times 1$) created using its neural depth sensing mode and camera poses in the form of transformation matrices(4×4) which was created using sensor fusion between traditional VO and Interial Measure Unit(IMU) and Barometer exclusively for estimating the camera height.

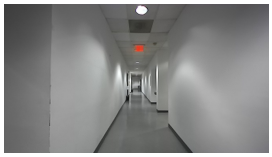
The dataset was created in the basement of the 5 Metrotech building as it primarily consisted of white walls which indicates low or sparse features. The camera calibration matrix of the ZED2 camera in the previously specified mode had focal length(f_x, f_y) as $(261, 261)$ and principal point(c_x, c_y) as $(335, 193)$ though these were not used. Robot operating system(ROS) along with the ZED_wrapper for ROS was used get the necessary frame, the frames were extracted manually to create the dataset.

The created dataset consisted of 1311 RGB images, Depth maps and camera poses similar format in Section 4.1. The camera was moved for $56m$ in total and has 2 . Figure 3a and Figure 3b are the RGB image and Depth map of a frame from corridor 1. Figure 4a and Figure 4b are the RGB image and Depth map of a frame from corridor 2. In Figure 3b and Figure 3b the color variation is from yellow to blue which correspond to pixels closer and pixels further away from the camera respectively.

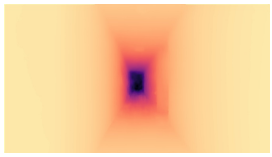
5 Experimental Results

This Section covers the entire evaluation of our proposed network architecture. The training is explained in Section 5.1 which was done on 7Scenes dataset. Benchmarks against other VO algorithms are given in Section 5.2. Testing was conducted on both the datasets in Section 4.1 and Section 4.2 is explained in Section 5.2 where there trajectories of the camera were generated.

[1]https://www.stereolabs.com/zed-2/

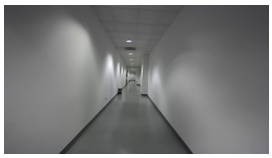


(a) RGB Image

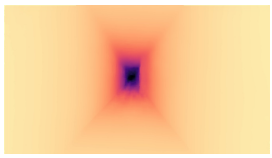


(b) Depth Map

Figure 3: A frame from corridor 1 from our dataset RGB(left) and Depth Map(right)



(a) RGB Image



(b) Depth Map

Figure 4: A frame from corridor 2 from our dataset RGB(left) and Depth Map(right)

Though most of the current research treat visual odometery as a depth estimation to estimate the scale and pose estimation jointly. Moreover, these algorithms are employed for only monocular cameras as stereo cameras are able to use the baseline information between both the lens and their disparity maps to retrieve the scale. To keep the problem simple we have not computed the depth maps but instead used the depth maps from the respective datasets in Section 4.1 and Section 4.2.

5.1 Training

The model was trained on dataset in Section 4.1 for 4500 epochs. From each scene two sequences were used for training the model. Batches of 25 RGB Images and 25 Depth Maps in sequence were created and passed into the model for training. The model was trained to predict the rotation and translation between the 24^{th} frame and 25^{th} frame.

For validation, different sequence from the each scene was taken and model was validated on that. Later the model was tested on different sequence from each scene and trajectory of these sequences were created, it is further discussed in 5.2.

To learn the model parameters the euclidean distance between the ground truth pose(p_t,ψ_t) and predicted pose($\hat{p_t}$,$\hat{\psi_t}$) is minimized. The loss function is composed of Mean Squared Error of position and orientation:

$$Loss = \frac{1}{N}\sum \|p_t - \hat{p_t}\|^2 + \alpha * \|\psi_t - \hat{\psi_t}\|^2$$

where N is number of samples and α is the scale factor to balance the weights of translation and rotation. The rotation in the F2F pose is two orders of magnitude smaller than the displacement. In order to balance the estimation in translation and rotation better rotational error α is 100.

For optimization of weights, Adam(el.at. 2014) optimizer was used with a learning rate of 0.001. Besides, the training data was normalized with standard ImageNet(el. at. 2009) dataset mean and standard deviation, and Xavier weight initialization are used to make the network converge faster and better.

The fig. 5 show the evolution of training loss. The fig.5 suggests the loss decreased in first 100 epochs and later it decayed. fig.8 flows the similar trends as fig.5, but it takes more epochs to decay. The fig. 6 and fig. 9 show the euclidean distance between the predicted and actual translation between two frames in meters. Since the motion of camera between the frames is less the translation loss is very less. Similarly fig.7 and fig.10 show the euclidean distance between the predicted and actual euler angles between the frames in radians.

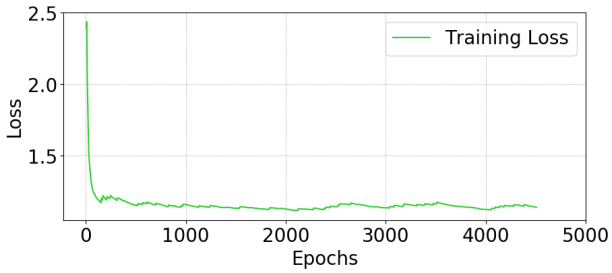


Figure 5: Training Loss

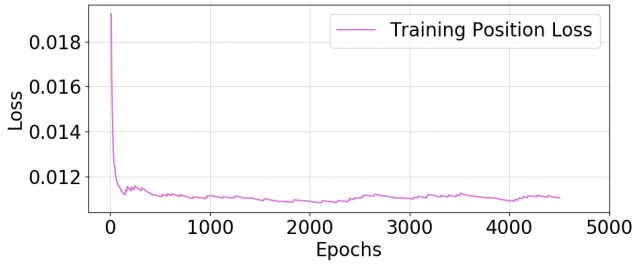


Figure 6: Training Position Loss

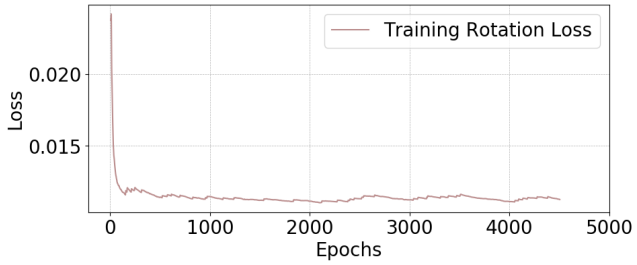


Figure 7: Training Rotation Loss

5.2 Benchmarks

Table1 shows a comparison of the performance with existing learning based methods on the 7Scenes dataset4.1. Following the standard benchmarking metrics, we report the median pose error. The error is calculated over all the sequences present in each scene. In most of the sequences the translation results from the proposed model is close to State of the art results of VLocNet++. State of the art results were achieved in office and pumpkin scene with the least translation error. The great results of the model can be attributed to the accurate input depth.

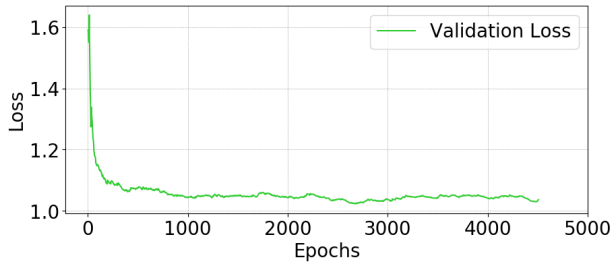


Figure 8: Validation Loss

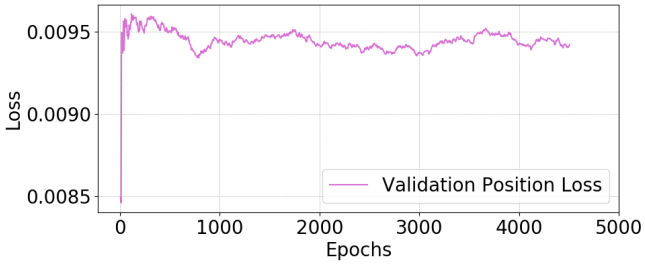


Figure 9: Validation Position Loss

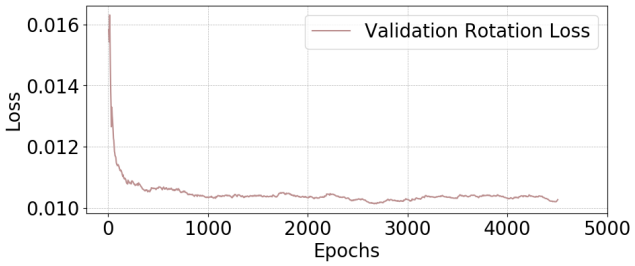


Figure 10: Validation Rotation Loss

Trajectory Generation Trajectories were generated for each sequence to evaluate the accuracy of our model. Trajectories were created for sequences not seen previously by the model to evaluate the performance in an unbiased manner.

To generate the trajectory, the model takes 25 frames of RGB Images and Depth Maps as input, then it predicts the relative pose between the 24^{th} and 25^{th} by using the temporal relationship between all the frames. The predicted relative poses are converted into transformation matrix and multiplied with the next transformation matrix to get the current camera position, doing this sequentially gives the entire trajectory. Each of the trajectories generated have both the actual trajectory and the predicted trajectory by the model.

Table 1: Evaluation of our models performance against others for the 7Scenes dataset

Scene	SCoRe Forest	DSAC	VLocNet++	NNnet	PoseNet2	Ours
Chess	0.03m, **0.66**°	0.03m, **0.66**°	**0.0018m**, 1.17°	0.13m, 6.66°	0.13m, 4.48°	0.01m, 3.35°
Fire	0.05m, 1.50°	0.04m, 1.50°	**0.009m**, **0.61**°	0.26m, 12.72°	0.27m, 11.28°	0.013m, 2.55°
Heads	0.06m, 5.50°	0.03m, 2.70°	**0.008m**, **0.60**°	0.14m, 12.34°	0.17m, 13.00°	0.009m, 1.03°
Office	0.04m, 0.78°	0.04m, 1.60°	0.016m, 0.78°	0.21m, 7.35°	0.19m, 5.35°	**0.014m**, **0.77**°
Pumpkin	0.04m, **0.68**°	0.05m, 2.00°	0.009m, 0.82°	0.24m, 6.35°	0.26m, 4.75°	**0.008m**, 0.75°
RedKitchen	0.04m, **0.76**°	0.05m, 2.00°	**0.0017m**, 0.93°	0.24m, 8.03°	0.23m, 5.35°	0.21m, 4.34°
Stairs	0.32m, 1.32°	1.17m, 33.1°	**0.010m**, **0.48**°	0.27m, 11.28°	0.35m, 12.4°	0.011m, 2.89°

*__Bold__ values indicate the best performance for that sequence

Figure 11, Figure 12, Figure 13, Figure 14, Figure 15, Figure 16, and Figure 17 depict the trajectories for the dataset in Section 4.1 and Figure 18 depicts the trajectory generated for the dataset in Section 4.2.

The predicted trajectory when compared with the actual trajectory for each sequence indicates that model is predicting the camera positions correctly while the camera is in motion, this can also be seen in Table 1 which describes the translation error and rotation error on all the sequences in each scene.

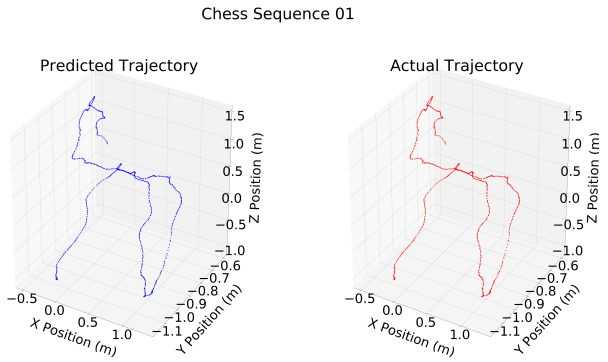


Figure 11: Trajectories for Chess sequence 1

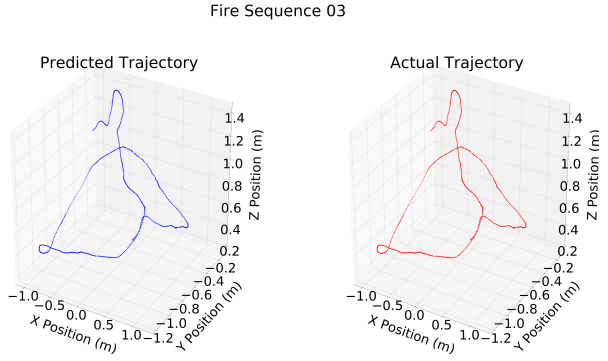


Figure 12: Trajectories for Fire sequence 3

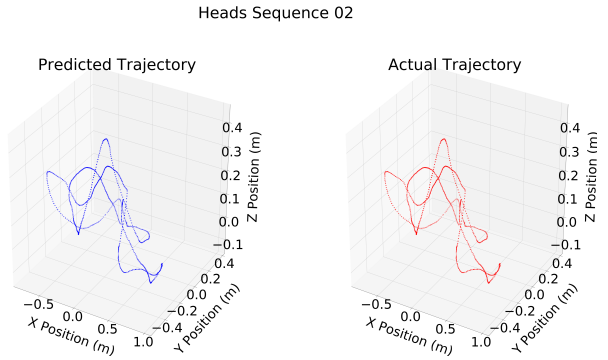


Figure 13: Trajectories for Heads sequence 2

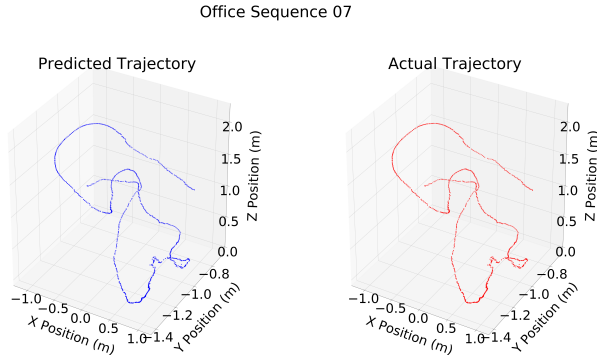


Figure 14: Trajectories for Office sequence 7

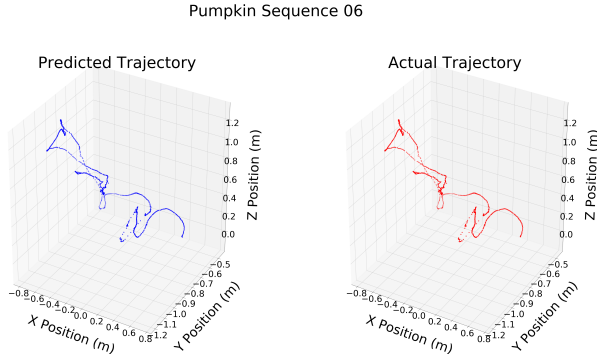


Figure 15: Trajectories for Pumpkin sequence 6

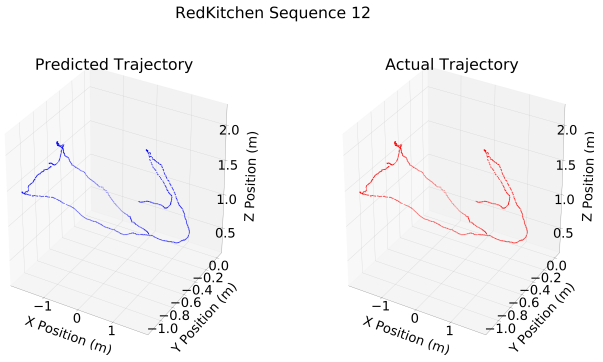


Figure 16: Trajectories for RedKitchen sequence 12

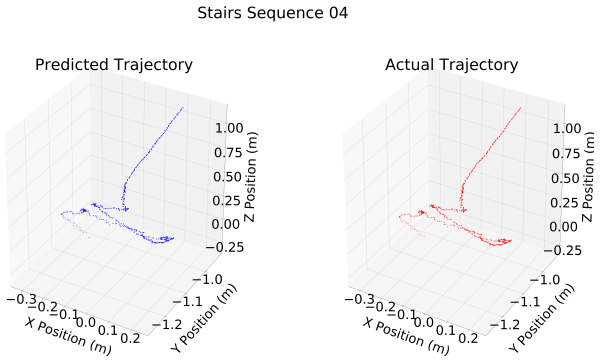


Figure 17: Trajectories for Stairs sequence 4

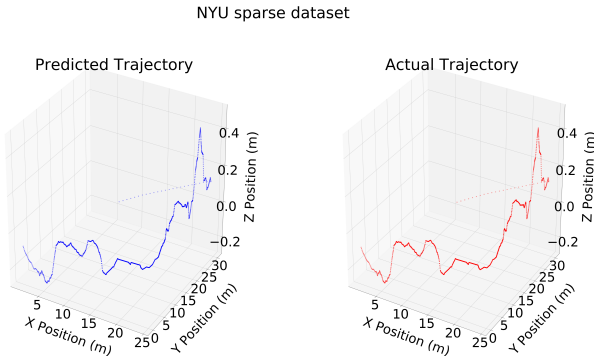


Figure 18: Trajectories for NYU sparse dataset

6 Conclusion

A temporal attention (TA) network was created to calculate the visual odometry of the most recent frame using a set of previous frames.

A new dataset called the NYU sparse dataset was created for testing the performance of a visual odometry algorithm in sparse feature environments. The network being tested showed promising results on this dataset and a separate dataset as well. The model's robust performance can be attributed to the fact that the Depth Maps are not being computed but metric Depth Map are taken directly from the dataset, doing this eliminates the scale requirement as it is already encoded in the Depth Map.

The intent of this project was to create a TA network which was accomplished, further training and testing was conducted on a dataset used in VO benchmarks. Our other goal was to test the model against other state of the art algorithms, to see where it stands. Additionally, the last deliverable was to create a sparse featureless dataset which was also accomplished successfully.

7 Future Work

The present network takes depth maps as inputs, in future the model can be modified to detect the depth of input images and later use that depth to detect the visual odometry. Currently the model only uses temporal attention, in future the convolution layers can be replaced with ViT to add spatial attention. The NYU sparse dataset can be extended to benchmark the performance of visual odometry algorithms in featureless environment.

Acknowledgment

We express our gratitude to the all the authors, who have worked hard to engineer such effective solutions for visual odometry. We would like to thank everyone whose comments and suggestions helped us with the project. We express our sincere gratitude to Professors Chinmay Hegde, Arsalan Mosenia, and the teaching assistant Teal Witter.

References

el. at., D. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255.

el.at., A. D. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *CoRR*, abs/2010.11929.

el.at., G. 2013. Real-time RGB-D camera relocalization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*.

el.at., K. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

el.at., W. 2017. DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks. In *ICRA*.

et.al, A. O. 2022. Dense Prediction Transformer for Scale Estimation in Monocular Visual Odometry. In *2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)*.

Li, Y.; Ushiku, Y.; and Harada, T. 2019. Pose Graph optimization for Unsupervised Monocular Visual Odometry. In *ICRA*, 5439–5445.

Patel, N.; Choromanska, A.; Krishnamurthy, P.; and Khorrami, F. 2017. Sensor modality fusion with CNNs for UGV autonomous driving in indoor environments. In *IROS*, 1531–1536.

Zhu, R.; and el.at. 2021. DeepAVO: Efficient Pose Refining with Feature Distilling for Deep Visual Odometry. *CoRR*, abs/2105.09899.