Exploring Various Methods for Stereo Vision and their Impact on Visual Odometry using the KITTI Dataset

Harris Nisar: nisar2
Saharsh Sandeep Barve: ssbarve2

1. Introduction

Visual odometry is the process of estimating the motion of an agent (vehicle, human, robot) by using the input of attached cameras. The pose of the agent can be estimated by analyzing changes in consecutive frames of an image as the agent moves through the scene. Put simply: as the agent, and thus the camera, moves through the scene, the changes induced in the captured images will be due to the motion of the agent. If the same points in two consecutive images can be matched, the transformation matrix that led to the changes in pixel space of the matched points can be computed. This transformation directly encodes the movement of the camera. Visual odometry has many applications in autonomous vehicles, robotics, and augmented reality. In fact, the visual ***odometry*** was based on its similarity to wheel ***odometry***, which aims to recover the motion of a vehicle from analyzing how the wheel changes over time. The main advantage of visual odometry over wheel odometry is the lack of mechanical disturbances than can impact the results in a vision system. For example, wheel odometry suffers from wheel slip in uneven terrain. Additionally, wheel odometry can sometimes be impossible in navigation tasks. Further, sometimes other sensors that could be used for navigation (i.e., GPS) are not available. Examples of these navigation tasks include space, underwater, or aerial navigation. For example, the first motivation of visual odometry was for the navigation of rovers during NASA's Mars exploration program. Here, wheel odometry suffers because of the rough terrain, and GPS is not possible to fix those errors [1].

The visual odometry problem is a subset of the structure from motion (SFM) problem. SFM tackles the more general problem of 3D reconstruction from ordered or unordered images. Visual odometry aims to recover similar information but is confined to dealing with consecutive images. The first works in visual odometry, focused on space exploration, were done by Moaravec in which a rover was fitted with a camera on rails [2]. The rover would move in a stop-and-go fashion. When stopped, the camera would slide on the rail to capture stereo images to recover depth. This depth information was used to convert from pixel space to metric space for all the pixels in the images. Features in these images were also matched with the previous image to estimate the motion as a rigid body transformation that best aligned the 3D correspondences in both images.

Many of the techniques from Moaravec's work continued to be utilized in future implementations of stereo visual odometry which rely on pairs of images of the same scene to compute depth, and subsequently, compute metric information about the camera. What changed was the hardware to capture the stereo cameras, the feature representation, the matching algorithms, or the trajectory estimation algorithms. It is also noteworthy that monocular camera based visual odometry is also a highly researched field to recover the trajectory without considering the actual scale.

This project focuses on understanding stereo vision based-visual odometry using the KITTI visual odometry dataset [3]. This dataset provides everything needed to build a visual odometry pipeline that can be experimented with, and that is exactly what was done. Specifically, this dataset consists of 22 stereo sequences, of which the first 11 also include the ground truth trajectory. The final 11 contain ground truths but are used by the authors of the dataset to evaluate performance of algorithms submitted by research groups. The dataset also includes camera calibration information. Using the stereo pairs, depth can be recovered using various algorithms. Here, depth is recovered using stock OpenCV functions and a deep learning approach using the ChiTransformer [4], with the goal to compare their performance. Then, for the next image, features were matched between the left cameras of the current and subsequent frame. Features were represented using SIFT or ORB, again to compare their performance. To match features, brute force matching and KDTrees were explored. With the depth computed and the features matched, the motion of the camera was estimated. Specifically, the rotation and translation that best described how the matched 3D points changed between was computed using the Perspective-n-Point (PnP) algorithm. The rest of the report is organized as follows: first, the details of the dataset and the visual odometry pipeline are discussed; second, the trajectory computation results are shown for various configurations of the pipeline; finally, the approach is discussed, highlighting the strengths and weaknesses of the pipeline and suggestions for next steps to improve the results.

2. Details of the approach

In this section the KITTI dataset and the visual odometry pipeline is discussed. Figure 1 summarizes the visual odometry pipeline. Blocks of the diagram are discussed in the subsequent sub-sections.

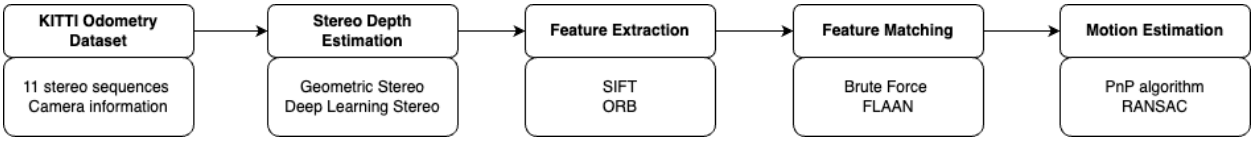

Figure 1 Visual odometry pipeline.

2A. KITTI Odometry Dataset

The KITTI dataset [3] was obtained from a Volkswagen Passat B6 fitted with the following sensors:

- 1 Inertial Navigation System (GPS/IMU): OXTS RT 3003
- 2 Laser scanner: Velodyne HDL-64E
- 2 Grayscale cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3M-C)
- 2 Color cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3C-C)
- 4 Varifocal lenses, 4-8 mm: Edmund Optics NT59-917

These sensors are fitted as shown in figure 2. In addition, the calibration information for all the cameras is included as rectified projection matrices with respect to the first grayscale camera. This means that these projection matrices take 3D points in a particular camera's coordinate frame to 2D points on the image of the camera with which the projection matrices are rectified against. For example, the calibration second grayscale camera would take 3D coordinates and would project them into the image plane of the first grayscale camera.

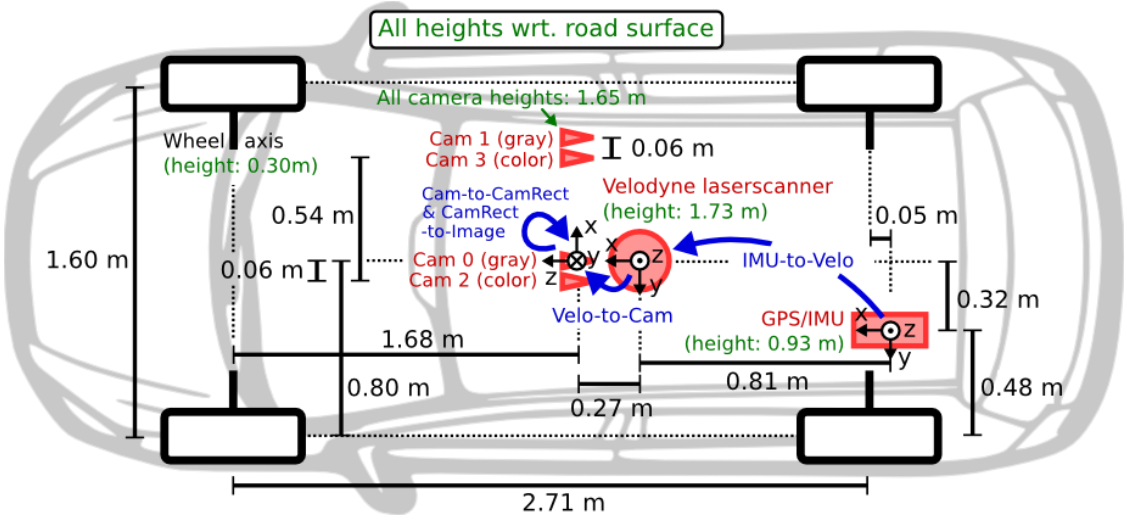

Figure 2 KITTI dataset vehicle setup.

Specifically, the odometry dataset consists of the output from the 2 grayscale cameras, 2 color cameras, and the 2 laser scanners. Additionally, the ground truth poses for the sequences are also included. Specifically, the outputs from the grayscale cameras are utilized for this implementation. These cameras output images of size 1382x512 pixels which are captured at 10 frames per second to match with the laser scanners' revolution time. It is important to note that the stereo images are already rectified and are therefore slightly smaller than what they were captured at. Furthermore, the sensor measurements are placed with respect to the first gray camera (Cam 0). Figure 3 shows a sample stereo pair from one of the sequences along with some examples of the ground truth trajectories.

Figure 3 Top two are the stereo image pair. Bottom is ground truth trajectories.

2B. Depth Estimation

One way to estimate the motion of a camera is by recovering 3D correspondences between matched points in subsequent images. However, when an image is captured of a scene, the 3D information is lost due to the projection. One way to recover this information is through stereo depth estimation. Once this depth is known, it can be utilized to recover the metric information from the pixels in an image. Here, two approaches for depth estimation were explored: geometric stereo and deep learning stereo.

Geometric Stereo

Binocular stereo is a vision-based depth perception technique that mimics the way humans perceive depth using two eyes. In binocular stereo vision, a pair of cameras are used to capture images of a scene from slightly different viewpoints. The key for binocular stereo is that the images are usually taken from cameras with image planes that are parallel with each other. The focal lengths of the cameras are the same and the camera centers are typically also the same height. Then, by the epipolar constraint, the epipolar lines fall along horizontal scan lines of the image. This means that corresponding 2D pixels in the two stereo images can be found by scanning the second image for patches that best match the patch of interest in the first image. If the image planes are not parallel, a homography is found to project the view onto a common plane in a process called stereo image rectification. Because the KITTI dataset already provides rectified images, this was not needed. The key to stereo is that the difference in pixels between where a patch appears in one image compared to the other is inversely related to depth. This difference in pixels is referred to as the disparity. It can be easily shown using similar triangles that $Z = \frac{fT}{d}$, where Z is the depth of the pixel, d is the disparity of the pixel from where it is in the other image, f is the focal length of the camera and T is the baseline. Note, that a small disparity means the pixel is far away and a large disparity means the pixel is close. Once this depth is determined, it can be utilized to recover the 3D information of the image; that is the 3D point corresponding to each pixel.

Given a pair of rectified images two algorithms for computing the geometric stereo depth map are explored. First, a brute force approach is implemented which is a simple search algorithm. Specifically, for each pixel x in the first image, the epipolar scanline in the second image is searched to find the best match. The epipolar scanline simply refers to pixels in the same row number in the second image as the row number of the pixel of interest in the first image. To determine which is the best match, a patch with a given window size around x and around a pixel on the scanline in the other image can be extracted. Then, these patches can be compared using various cost functions such as sum of squared differences (SSD), sum of absolute differences (SAD), and normalized cross correlation (NCC). Another, more sophisticated algorithm, is semi-global block matching (SGBM) which calculates the similarity between corresponding pixels in the left and right images, generating a cost volume. SGBM employs a semi-global approach, considering both local and global consistency, and aggregates costs over multiple

paths in the image. The disparity map is then computed by finding the disparity value with the minimum cost for each pixel. OpenCV implementations for both algorithms were utilized in this pipeline [5]. Figure 4 shows a sample output of depth maps computed with these two approaches.

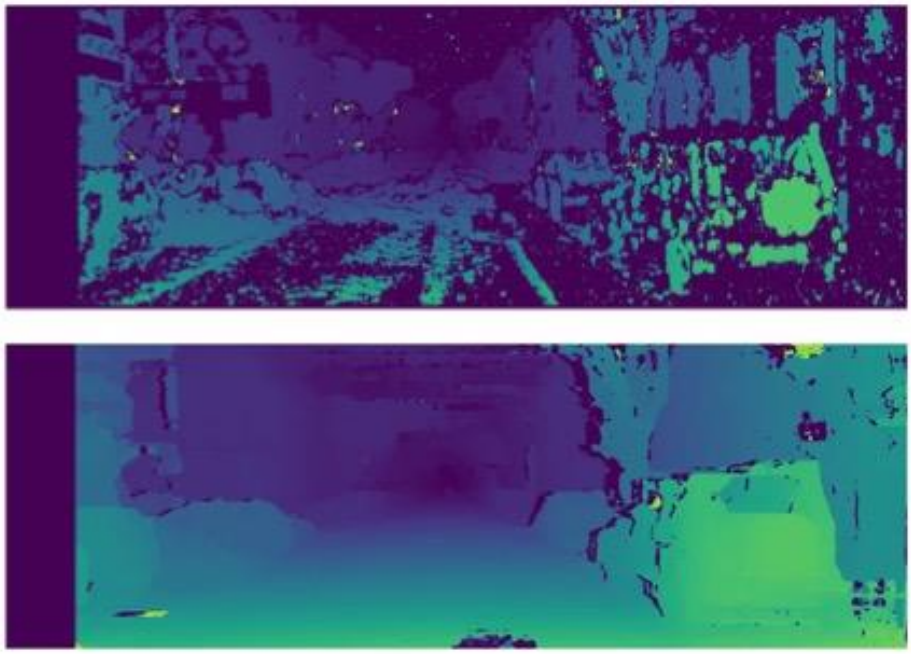

Figure 4 Disparity maps from stereo SGBM and stereo BM approaches.

Deep Learning Stereo

Deep learning models have been effectively employed for binocular stereo matching problem to generate accurate depth maps of the scene. ChiTransformer is one such deep learning based binocular depth estimation network [4]. It consists of an encoder decoder-based architecture. The encoder comprises of a pair of hybrid vision transformer model (ViT) backbones with ResNet-50 for patch embeddings. Figure 5 below showcases the model architecture in greater depth.

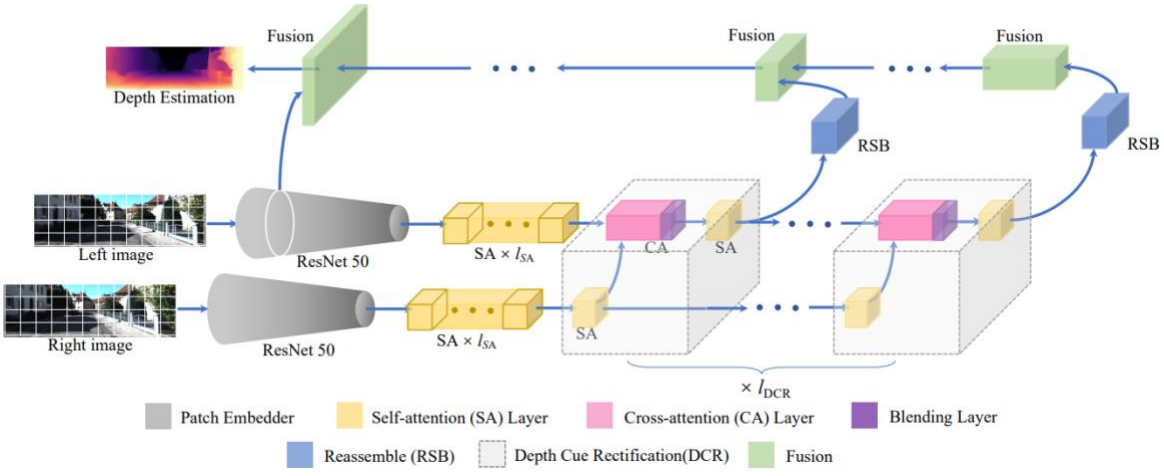

Figure 5 ChiTransformer architecture.

For training, the authors utilized the KITTI 2015 stereo dataset. The model follows a self-supervised stereo training method, wherein it predicts the target image from the source image in stereo image pair. As an intermediary step, it generates depth estimates and utilizes additional configuration information such as camera calibration, stereo camera setup to produce the final correspondences between the two images. During training, the objective is to

minimize the photometric reprojection error. The pre trained model provided by the authors is utilized to generate the depth maps in our experiments as shown in Figure 6.

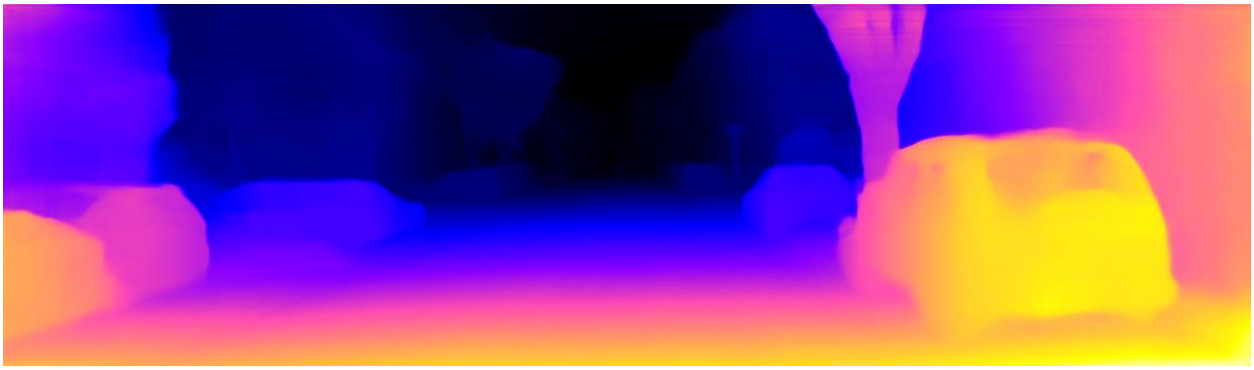

Figure 6 ChiTransformer depth map output.

2C. Feature Extraction

Once the depth has been recovered, the next step in visual odometry is to determine corresponding matches between subsequent frames. To determine matching regions, reliable features must be extracted. Several approaches exist for feature extraction from images. Here, two such approaches are explored: scale invariant feature transform (SIFT) [6] and Oriented FAST and rotated BRIEF (ORB) [5]. Figure 7 shows sample output comparing the features extracted with both approaches.

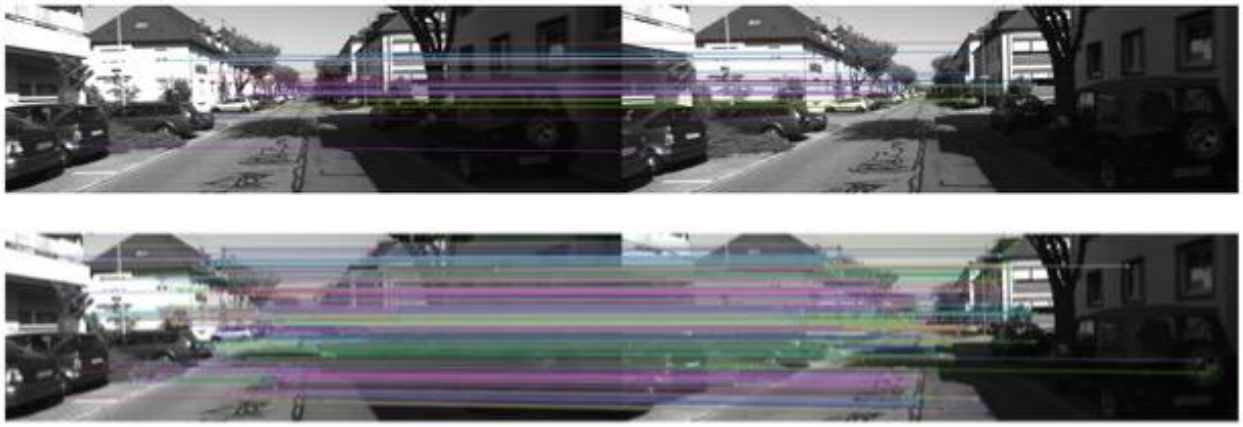

Figure 7 Feature extraction using ORB (top) and SIFT (bottom) approaches.

Scale Invariant Feature Tracker (SIFT)

SIFT begins by detecting potential keypoints as local extrema in a scale-space representation, considering multiple scales. This makes it scale invariant. Keypoint localization ensures accuracy, filtering out points with low contrast or poorly defined edges. Each keypoint is assigned a dominant orientation to achieve rotational invariance. Descriptors are then generated based on gradient orientations within localized regions, providing distinctive representations. These descriptors are 128-dimensional vectors of continuous values which can be compared by the Euclidean distance between them.

Oriented FAST and Rotated BRIEF (ORB)

ORB begins with the FAST keypoint detector, identifying keypoints based on intensity patterns in circular neighborhoods. The BRIEF descriptor generation technique then creates binary descriptors for these keypoints by comparing pixel intensity pairs. ORB addresses scale invariance by using a scale-space pyramid, and it introduces rotational invariance through "oriented FAST," which computes keypoint orientations. ORB descriptors are 256-element, binary vectors. Therefore, Euclidean distance is not used to compare them and, instead, Hamming distance is used, which is a measure of the dissimilarity between two binary vectors.

2D. Feature Matching

Once the features have been detected, either as SIFT features or ORB features, they must be matched. While the similarity measurements between the two feature descriptions are different, similar approaches can be used to find matches. Brute force matching, a straightforward yet computationally intensive technique, involves exhaustively evaluating all possible correspondences between features in consecutive frames. This method is especially valuable for its simplicity and applicability in scenarios with distinct features. On the other hand, Fast Library for Approximate Nearest Neighbors (FLANN) provides an efficient alternative by employing indexing structures to accelerate the search for approximate nearest neighbors, making it well-suited for large-scale feature matching tasks. To refine the matches, Lowe's ratio is utilized. To implement this, the 2 best matches in the second image for each keypoint of the first image are kept. Lowe's ratio checks that the two distances are sufficiently different. If they are not, then the keypoint is eliminated [6].

2E. Pose Estimation

Upon establishing the correspondence between 3D object points in camera coordinates and their corresponding 2D image points, the Perspective-n-Point (PnP) algorithm is applied to estimate the camera motion between a pair of images. The algorithm comprises three main steps. Firstly, an initial estimation of rotation and translation is obtained using the Direct Linear Transform (DLT) algorithm. Subsequently, refinement is undertaken through iterative processes employing the Levenberg-Marquardt (LM) algorithm, enhancing the precision of the pose estimation. To bolster the algorithm's robustness in real-world scenarios against potential outliers and noise in the data, the Random Sample Consensus (RANSAC) technique is incorporated.

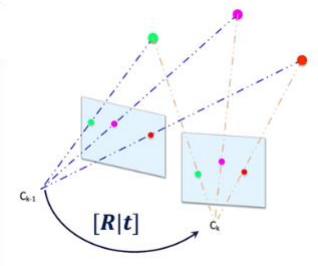

Figure 8 PnP algorithm.

Results

With the pipeline in place, the next steps were to compare visual odometry performance between the two depth estimations described above: geometric stereo and deep learning-based stereo. It is hypothesized that the deep learning-based approach would outperform the geometric stereo based on the outputs shown above. The deep learning-based stereo had smoother output with no missing values, as opposed to the geometric stereo which was noisier and had missing values.

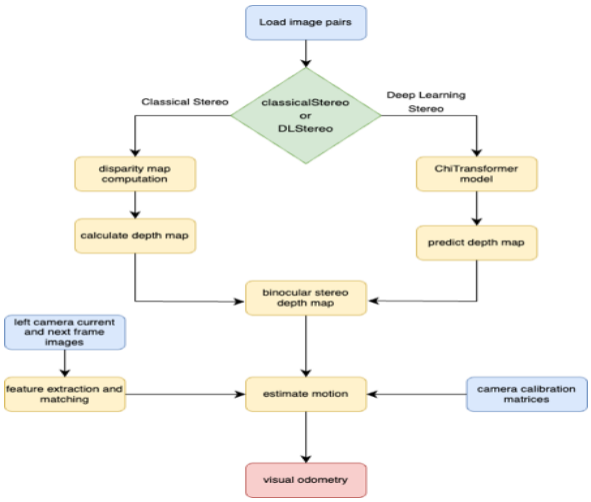

Figure 9 Experimental design to compare stereo approaches.

To compare the performance, three of the 11 stereo sequences in the KITTI dataset with ground truth were passed through the pipeline. In the first pass the depth estimation was done with geometric stereo and in the second pass it was done with deep learning-based stereo. Figure 9 shows our experimental protocol. Figure 10 shows the trajectory results. See this link for a video showcasing the results.

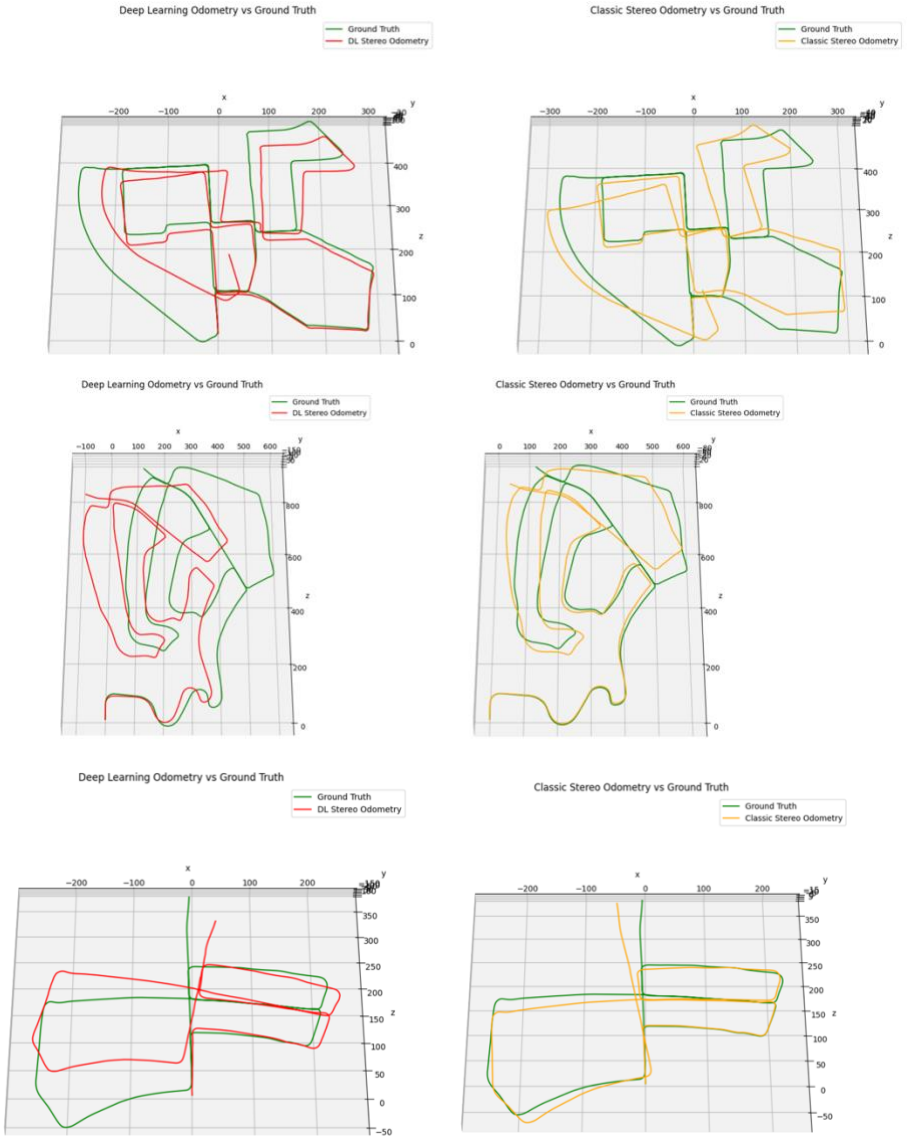

Figure 10 Trajectories generated for the two stereo depth estimation approaches compared to the ground truth trajectory. Top is sequence 00, middle is sequence 02, bottom is sequence 05.

To quantify the difference between the trajectories, the mean Euclidean distance between the translation vectors for the estimated poses and the ground truths was computed for each sequence. Figure 11 shows these results.

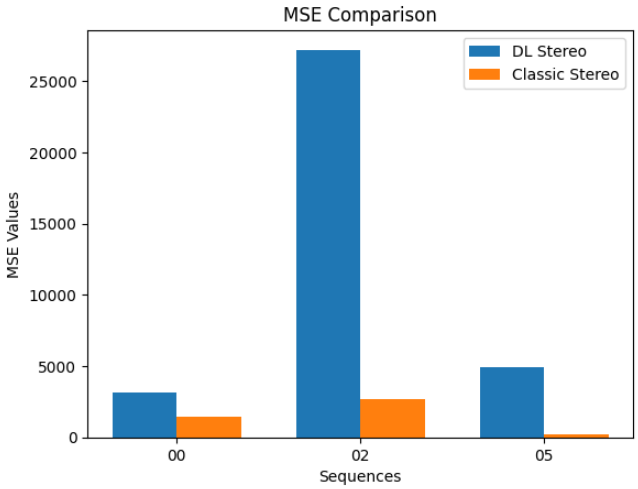

Figure 11 MSE for the three trajectories for the different stereo approaches.

Figure 12 shows an example trajectory where the odometry did not extend far enough when deep learning was used, although, the shape is maintained.

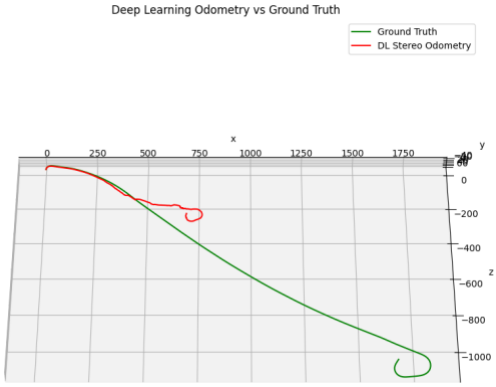

Figure 12 An example of the deep learning stereo in which the shape of the trajectory was predicted but not the scale.

Discussion and Conclusions

First, qualitatively it can be seen in figure 10 that both deep-learning and geometric-based stereo yielded trajectories that were similar in shape to the ground truth. As time went on, both estimations were susceptible to drift as expected in continuous estimation procedure. This can be understood by the fact that as estimations are made, they are not perfect. These errors accumulate and lead to degrading estimations, usually observed in the form of drift. One approach that tries to correct for this eventual degradation is the Kalman filter which aims to combine sensor readings from multiple modalities to bolster the estimation certainty. The basic idea is that the Kalman filter will check to see if reading from the first sensor is probable given the sensor reading from the second sensor. If one sensor is more confident in its output, that sensors measurement would receive a higher weight. In practice, visual odometry is not the only sensor available in many applications. For example, the KITTI dataset is fitted with a LiDAR scanner. Future extensions of this pipeline could include the incorporation of this LiDAR data with the visual odometry output to see if that improves drift in both the deep learning or geometric stereo approaches.

More surprisingly, it appears that geometric stereo had less drift than the deep learning-based approach. Further, the MSE between the predicted trajectories and the ground truths was higher when deep learning was, particularly for sequence 02. It makes some sense that sequence 02 was the worst for deep learning because it was also the worst for the classic approach. This can be expected because this sequence had a lot of curves. While the disparity maps (and consequently, the depth maps) for the geometric approach had more noise, they did not seem to have as much of an impact as expected. There could be several reasons for this. For instance, the deep learning

model was taken as is without any fine tuning. One idea could be to train the stereo model to perform better for the task of odometry. For example, the L2 loss between the computed trajectory and the ground truth could be differentiated through the deep learning model in a self-supervised fashion. Another idea would be to explore other types of deep learning models for stereo depth estimation. Yet another idea could be to find or build an end-to-end deep learning model for visual odometry. Another interesting result was that the deep learning model seemed to estimate depth poorly for some of the trajectories. For instance, some of the trajectories matched the overall curvature but did not extend far enough in the z-direction, as shown in figure 12. This could directly be because the deep learning model is not outputting depth perfectly. It could also be that our pipeline fails to address the interface between the deep-learning stereo estimation and visual odometry.

For this project, visual odometry was explored. Specifically, the KITTI dataset was used to test a visual odometry pipeline. After implementation of the pipeline, two different stereo depth estimations were compared in their visual odometry performance. Several key components in computer vision were revised, learned, and implemented including stereo vision, feature extraction, feature matching, the PnP algorithm, and RANSAC. It was amazing to see all the concepts explored in class come together to solve a practical problem that was visual odometry. In addition, it was solidified that taking deep learning models off the shelf is not an intelligent solution to any problem. While these models have impressive performance for the task they are trained on, specific considerations must be taken if they are to be utilized in the context of a larger pipeline.

Statement of individual contribution

Both authors contributed equally in both coding, writing the report, generating figures, and proofreading.

Acknowledgements

We'd also like to acknowledge the work at this repository for making the concepts of visual odometry easy to understand and providing a lot of code that we reused for the pipeline.

References

[1] D. Scaramuzza and F. Fraundorfer, "Visual Odometry [Tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, no. 4, pp. 80–92, Dec. 2011, doi: 10.1109/MRA.2011.943233.

[2] H. Moravec, "Obstacle Avoidance and Navigation in the Real World bya Seeing Robot Rover." Sep. 1980. [Online]. Available: https://www.ri.cmu.edu/pub_files/pub4/moravec_hans_1980_1/moravec_hans_1980_1.pdf

[3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI: IEEE, Jun. 2012, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[4] Q. Su and S. Ji, "ChiTransformer:Towards Reliable Stereo from Cues," 2022, doi: 10.48550/ARXIV.2203.04554.

[5] G. Bradski, "The OpenCV library," *Dr Dobbs J. Softw. Tools*, 2000.

[6] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: 10.1023/B:VISI.0000029664.99615.94.