3D Reconstruction - Team APA

Aidan Lawford-Wickham, Predrag (Pedja) Muratovic, Armaan Lalani

CSC420 - Final Report

December 18, 2020

Abstract

The project at its highest level is the process of reconstructing a 3D object from a set of distinct images of that object. This specific problem has several very important applications, one of the most important being computer vision for self-driving vehicles to assist in determining their relative position. In solving this problem, a couple of approaches were considered and researched, including the use of neural networks and the more typical approach of using stereo reconstruction. The use of neural networks presented a few challenges which would be extremely difficult to overcome as well as offering minimal opportunity to consolidate course concepts. As a result, the use of stereo reconstruction was pursued which required a number of smaller subtasks including camera calibration, matching of key points between images, determining the fundamental matrix, generating disparity and depth maps, image rectification, post-smoothing and processing the disparity map to create a more cohesive reconstruction, etc. After creating and processing the disparity map, results could be visualized by combining the map with the RGB values of a reference image to generate and explore a point cloud in 3D.

Introduction and Literature Review

At the onset of this project, determining an adequate and efficient method of tackling this problem was the first priority. With a strong background in the use of neural networks from ECE324: Introduction to Machine Intelligence, the use of neural networks to solve this problem seemed like a creative and interesting method to solve this problem.

The use of neural networks would require the following: a 3D model of an image and several 2D images of that model. The neural network would essentially be finding shapes that are consistent with the images fed to it based on the different angles in order to reconstruct the 3D shape it was provided with. [1] The network would utilize these 2D images to obtain a 3D model by computing loss relative to the provided 3D model.After the training of this network, the model would be able to generalize to various forms of 2D images provided to it in order to create an approximate 3D reconstruction.

Furthermore, the papers written by Choy et al. and Han et al. on 3D reconstruction using deep learning, specifically Generative Adversarial Networks (**GAN**s), were used as a guide for the neural network approach [2][3]. These papers provided a means to develop a loss function for the network, which would allow the model to assess the strength of its output 3D reconstruction of an input scene. In our approach, given a set of input images of an object/scene, the GAN would then output a voxelized representation of said object/scene. Note, a voxelized representation is simply a collection of voxels, 3D version of a pixel (i.e., cubes), that form a shape of an object or scene. In order to compute the loss between the output and a ground-truth 3D model, one could use one of two metrics; such as, Intersection over Union (**IoU**) or a 3D voxel-wise softmax. We came to a consensus that the Intersection over Union was a suitable metric for our GAN as this metric is popular in deep learning reconstruction methods. The IoU is simply the ratio of the intersection between the volume of the GAN's predicted shape and the ground truth's volume, over the union of the two mentioned volumes [2]. The IoU is represented mathematically as,

$$IoU = \frac{\hat{V} \cap V}{\hat{V} \cup V} = \frac{\sum_i \{I[\hat{V}_i > \varepsilon] * I[V_i]\}}{\sum_i \{I[\hat{V}_i > \varepsilon] + I[V_i]\}} \qquad (1)$$

Where $I[]$ denotes the indicator function, $\hat{V}_i$ denotes the predicted value at the ith voxel, V_i is the ground-truth value at the ith value, ε is a given threshold value [2]. As one can see, the IoU is a metric to be maximized, rather than minimized as for typical loss functions; thus, the larger the IoU value the more successful the model was at its reconstruction attempt.

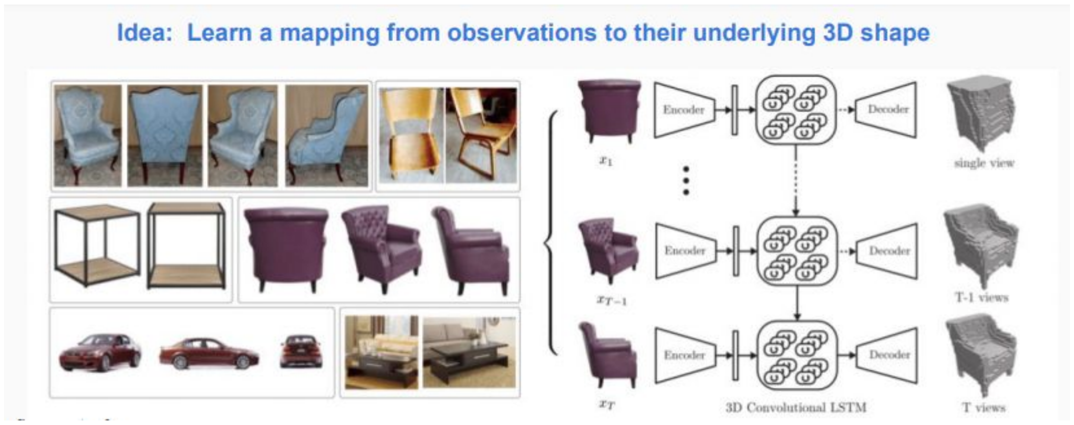


Figure 1: a generic overview of utilizing 2D images in a neural network to create a 3D model

The neural network would combine the use of convolutional layers, LSTMs (Long Short Term Memory), and linear layers. The use of convolutional layers is to identify the most important features in the original images by learning the aspects of the image that are most crucial when determining an accurate 3D model. LSTMs are memory cells which are very strong at identifying long range dependencies and storing in memory the most important aspects of the sequence of images passed through the network. The use of LSTMs increases the rate of learning for the network which is extremely essential for a training loop of this magnitude. In conducting research on the implementation of a model for 3D reconstruction, the following schematic was found and replicated to a certain extent using PyTorch.

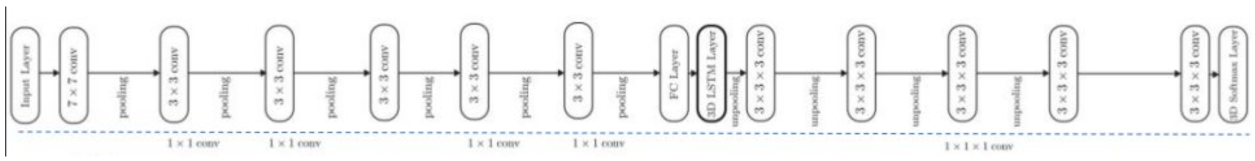


Figure 2: schematic representation of 3D reconstruction neural network

To begin the process of 3D reconstruction with neural networks, finding an adequate dataset was the clear first step. The dataset had to meet a number of needs in order to be useful including:

- Multiple images of the same 2D object or scene
- A 3D representation of that object or scene (i.e. CAD, .ply, etc.)
- A significant number of similar, but different 2D scenes to form a training set
- Availability
- Dataset must be of practical size, i.e., not too large such that download time was on the order of several hours

Several publicly available datasets were downloaded including: ShapeNet, Pix3D Plus, and Pascal 3D. When exploring the contents of these datasets, there were a few problems with each that would make the implementation of this task extremely difficult including:

- Computation during the training would have far too 'complex'. For instance, when loading 3D representations of the ground truth voxels as arrays, the resulting arrays contain on the order of millions of data points. Thus, computing the IoU metric would have been both expensive with respect to time and computation.
- Most datasets did not provide voxelized representations of their objects but rather CAD models of said objects, which provided a surface-based representation of the object/scene. Therefore, in order to compute the IoU metric, one would have to either develop a voxelized representation of the ground-truth CAD models, or convert the GANs output voxel representation to a surface and use a different metric, such as the Chamfer distance to compute loss. As mentioned a priori, methods of creating and loading voxelized representations are very inefficient; hence, we decided not to convert the CAD models into their corresponding voxels. Moreover, in converting voxel representations to their corresponding surface-based representations, one would have to map the voxelized representation to a triangular mesh. This process is also very complex and would require third party software like SolidWorks to do so. In conclusion, this method is very time consuming and would reduce the amount of time for training purposes.

There are definitely ways to get around some of the issues we were facing including curating our own dataset, utilizing online 3D models to generate various 2D images, etc. however, the time required to accomplish these tasks would have been immense and would have left little to no time for actual implementation and debugging.

Therefore, as a group, a consensus was reached to pivot to the more typical form of implementation using stereo reconstruction. Much of the material used for stereo reconstruction came from lecture notes, OpenCV tutorials [4], as well as Zimmerman and Hartley's Multiple View Geometry in Computer Vision.

Methodology, Results, and Experiments

3D reconstruction can be solved in a number of ways with stereo reconstruction. Based on our research and viewing a few online tutorials, we felt the best way to work towards a solution would be to each attempt to implement our own method of solving the problem, while continuously being in contact regarding our progress so we could inherit progress from each other. The following three subsections provide an overview as to the process each one of us took in solving this problem.

<u>Approach 1 (Aidan):</u>

In order to achieve a baseline implementation of stereo reconstruction our first approach was to perform a reconstruction using only pairs of two 2D images which were taken using the same camera, with the only difference in the image being a translation in one direction between the two images. This could be easily extended to using more than two images in the future, and we could ensure the correctness of our reconstruction approach without being concerned about issues like rectification. By using a dataset of 2D stereo image pairs sourced from Middlebury College [5] that also contained metadata like camera intrinsics, the task was largely simplified to focusing only on reconstruction. For this stage, the reconstruction process took the following steps:

1. Efficiently applying a Gaussian filter to smooth the input image pair.
2. Finding corresponding key points between the two images using SIFT feature detection algorithm.
3. Generating the disparity map. In this approach, the disparity map was generated manually by triangulation using the detected feature pairs rather than using a library like OpenCV. Since the image pairs taken from the Middlebury dataset only differ by a translation in the x direction, this process simply involves iterating over the point matches and computing the difference in their x position, then storing this in the disparity map with position corresponding to the reference image coordinates.
4. An alternative method to finding the key points and disparity map was also explored, and this method allowed more key points to be distinguished, hence providing a more consistent overall reconstruction. This method involved computing the disparity for any point in the reference image to be the minimum sum of absolute differences of all pixels within a sliding window that moves across the corresponding pixel row. When this method was used, the disparity map was more consistent as it contained a much larger number of points. After repeating the process multiple times too find the ideal parameters, the resulting raw disparity map is the following:

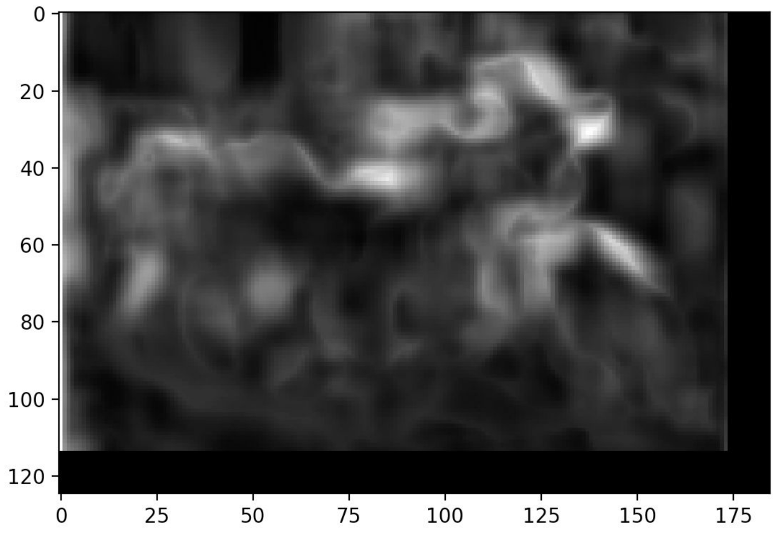


Figure A1.1: Raw disparity map after using the alternative method defined above.

5. Post-processing the disparity map. Since the disparity map corresponds to depths of different areas in the reconstruction, and in this approach the disparity map is based only on two images with a translation, the resulting disparity map appears to have drastic jumps between depths. We applied post-processing techniques to smooth these jumps and give the reconstruction more realistic characteristics. This involved smoothing over a number of iterations, where at each iteration the smoothing function updates disparity according to statistical measures like mean and mode of a sliding window around it, then thresholding the value. Combined smoothing according to these statistics in combination with applying another Gaussian filter gave the most realistic results and reduced the number of unrealistic gaps.
6. Computing a point cloud. Finally, after post-processing the disparity map, the disparities can be translated into points in three dimensions. The x and y coordinates for each point correspond to the position in the reference image, while the z coordinate is computed based on the disparity, baseline, and focal length as follows:

$$z = \frac{baseline * focal\ length}{disparity}$$

We also applied thresholding to the z values to prevent values from exploding when disparity values were extremely small. When paired with the RGB values from the original reference image, the resulting 3D point cloud could be displayed as follows:

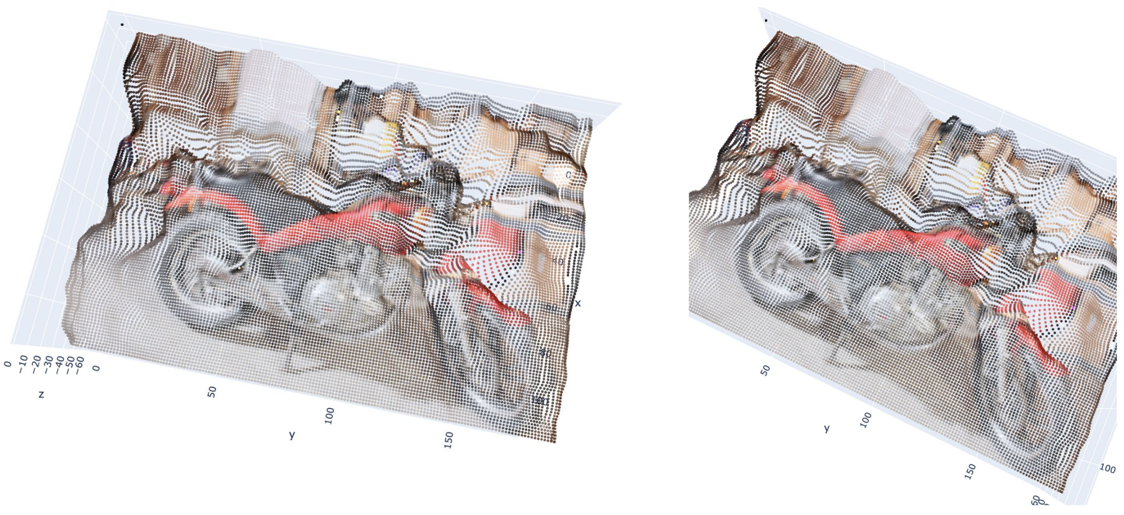


Figure A1.2: Point clouds reconstructed from the disparity map.

<u>Approach 2: Predrag (Pedja)</u>

The second approach is similar to that of the first approach done by Aidan, as well as the third approach done by Armaan. In this approach, instead of considering a translation between the two images in one dimension, the approach focused on translations in all three dimensions as well as a rotation. Furthermore, as in the first approach, only two images were used in order to develop a three dimensional reconstruction of the scene described by the images: The images and camera metadata (intrinsics) were also acquired using the Middlebury Dataset. Finally, this approach was tailored to class concepts regarding Homographies and Stereo Vision, as well as methods described in Zimmerman and Hartley's Multiple View Geometry in Computer Vision. The steps for this approach can be outlined as follows:

1. Acquire the image pair and the scenes intrinsics from the aforementioned dataset: Intrinsic matrices for the camera models and their corresponding baseline (seperation).
2. **Feature Detection**: Using the SIFT feature detection algorithm, obtain the best corresponding points between the left and right images, up to a certain threshold.
3. **Computing the Fundamental Matrix**: Once having obtained the corresponding points between the images, compute the fundamental matrix, F. In order to obtain the fundamental matrix, the Normalized 8-Point algorithm, as described in Zimmerman and Hartley, was used. First, the corresponding points in both images must be normalized; such that, the centroid of the points is at the origin of their corresponding coordinate system and the RMS distance from each point to the origin is $\sqrt{2}$ [6]. Then these points were used to obtain a matrix, *A*, of the form shown in lecture *Stereo Part 2: Slide 19.* Then computing the SVD of A and setting F to be the vector corresponding to the

smallest singular value of A. Once doing so, set the 3rd singular value in the SVF of F to be 0 to enforce epipolar constraints. Finally, the fundamental matrix was formed by denormalization using the transforms mentioned a priori.

4. **Epipole Detection**: Next, the left and right epipoles were found by computing the SVD of F; where, the right and left epipoles corresponding to the left and right null-spaces of F, respectively.
5. **Rectification I.** In order to rectify the images, one needs to find the homographies that will map the epipoles to infinity. First, the essential matrix, E, was computed using the fundamental matrix and the left and right intrinsic matrices. Second, the essential matrix was decomposed to extract the rotation and translation components needed for rectification. However, there are four possible choices of a rotation and translation that can yield the same essential matrix. Only one of these choices is suitable for rectification because it will be the only one to map the points in front of both cameras [6].
6. **Rectification II**: Once the correct rotation and translation pair was obtained, the corresponding homographies for each image were developed to make the image planes parallel to one another. However, after applying the homography transform, the resulting points are in the camera coordinate system. To map these points back to the pixel coordinates, the intrinsic parameters for the camera models were used as such,
    a. Pixel = (Intrinsic Matrix) x (Camera-System Points)
7. **Disparity:** Found as described in Approach 1, step 3.

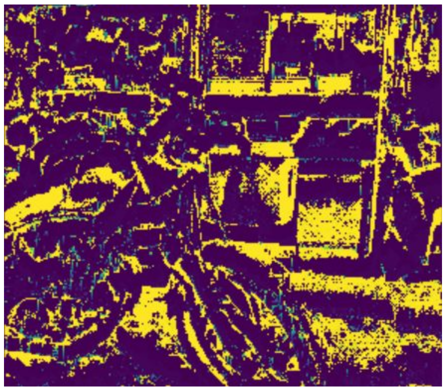
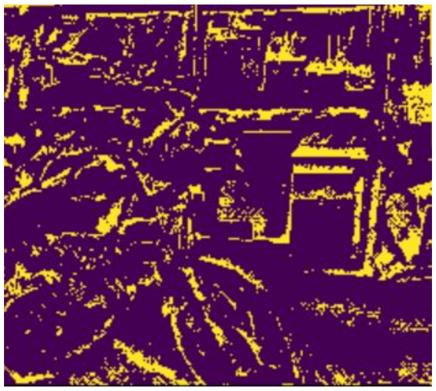


Figure A2.1 Disparity Map

8. **Depth**: Also found as described in Approach 1, step 5.



<u>Approach 3: Armaan</u>

The primary task involved in my implementation was to explore the various functions available through OpenCV. By developing a functional process using OpenCV [4] [7], this could easily be translated to our own functions moving forward. The process followed can be displayed based on the following steps:

1. Camera Calibration: using OpenCV's findChessboardCorners function, it would be possible to determine the intrinsic properties of the camera being used for 3D reconstruction. This was done by taking roughly 20 pictures of a generic chessboard and utilizing the function to obtain camera properties.

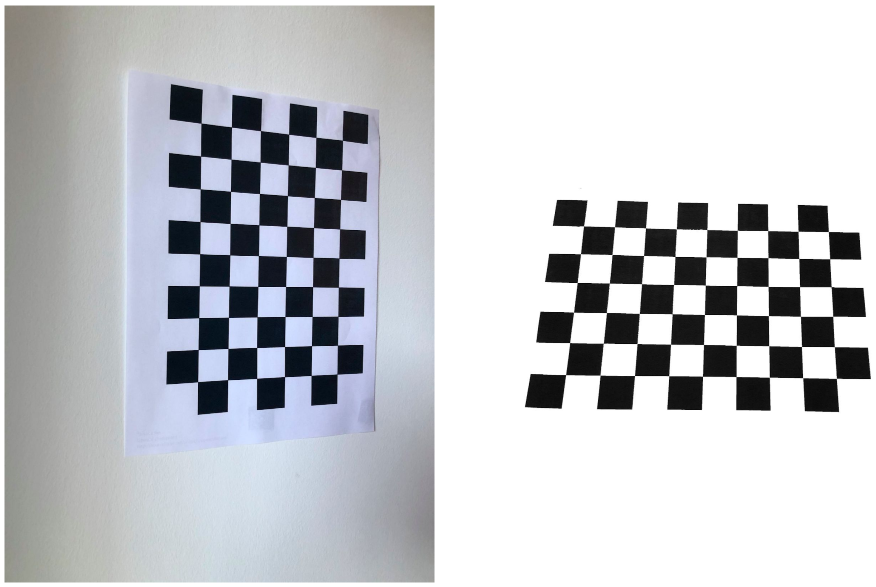


Figure A3.1: a sample image of the chessboard as well as one with a filtered background to make chess board recognition easier

2. Loading in two images from which a 3D model will be developed
3. Finding the fundamental matrix F: first found the matching points between the two images using a very similar approach seen in Assignment 3 with brute force matching. By utilizing SIFT, the strongest matching points were found with a threshold value of choice. The fundamental matrix F could then be determined based on these matching points by utilizing the OpenCV function findFundamentalMat.
4. Image Rectification: using both the fundamental matrix and camera intrinsic matrix, the two images could be rectified in order to be placed on the same plane using the OpenCV function initUndistortedRectifyMap.
5. Disparity Map: a disparity map was then created based on the rectified images in order to determine the 'distances' between points on the two images. This was done using the

OpenCV function StereoSGBM_create. As shown in the image below, the disparity map was not very smooth, and as a result, smoothing the disparity map was also attempted. However, when transitioning from the disparity map to the depth map, a lot of important information lacked from the disparity map which caused the depth map to be even worse. As a result, the original disparity map was used to create the depth map.

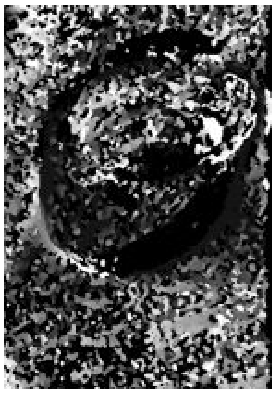


Figure A3.2: a disparity map of a roll of tape on a table

6. Depth Map: using the relative distances between points in the disparity map, the depth of pixel values could be determined using the focal length as well as the baseline.

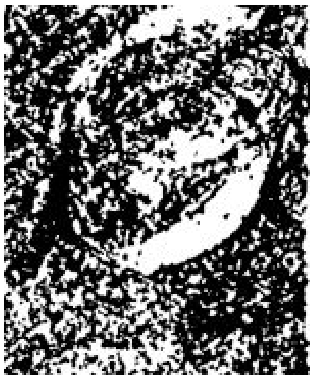


Figure A3.3: depth map of the roll of tape on the table

7. 3D Model: developing a 3D model from the image and its associated depth map was very difficult in this case. First of all, the depth map lacks 'smoothness' if compared to the depth maps we saw on Assignment 4. Another primary issue was the difficulty in obtaining the rotation and translation vectors needed for 3D projection. As a result, the final point clouds developed were not accurate.

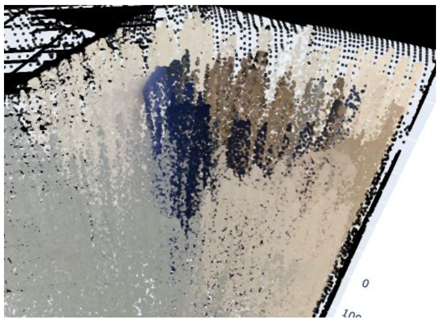


Figure A3.4: 3D point cloud of the roll of tape

As shown from the above results, using OpenCV for the primary computations did not turn out to be the most successful. There was a significant amount of variability between pictures used due to the number of parameters required in some of the OpenCV functions. Changing some of these parameters would have drastic effects on the final results and it was often hard to determine the ideal set of parameters. In conducting research on these functions and following online tutorials, there is no ideal method for selecting and determining these parameters. Several online tutorials simply state to alter the parameters until the ideal output is reached, however, this can often vary between the type of images used.

Another problem that arose from using OpenCV came from discussion with Pedja about the fundamental matrix. Using OpenCV functions used to calculate the fundamental matrix varied from when the fundamental matrix was calculated manually. As a result, in Pedja's approach to solving the problem, F was calculated manually rather than using the OpenCV function. The reason why the fundamental matrix was calculated manually was because, the average value of $e'^T F e$, where e's denote the epipoles, using openCV was found to be in the hundreds, but this quantity should be zero. When using the manually computed F, the average value was about 0.1; thus, providing a stronger fundamental matrix.

Based on the results of using OpenCV functions, we collectively came to an agreement to use manual functions as much as possible to avoid discrepancies. This would make the process of debugging much easier and also allow for a stronger understanding of course material.

Conclusion

As seen in our methodology and results above, our different methods to stereo reconstruction gave a mixture of both positive and negative results. By approaching this problem from a number of perspectives we were able to overcome some key challenges like efficient and accurate detection of matching features across the image pairs and image rectification, and these various approaches would provide a strong platform for future projects to build upon. Going forward, one of the next steps for this project would be to consolidate the successes of each of the aforementioned stereo methods into a single system that leverages our learning from each method. In particular, the ideal next step would be to use the image rectification techniques from approaches 2 and 3 to expand the algorithm from approach 1 to support a wider variety of images, and then extend this approach to support reconstruction from more than 2 images. The efficiency of this solution could also be improved by utilising functionality from OpenCV that was validated in approach 3.

The entire project provided us with the opportunity to consolidate many of the key topics that were taught during the semester. However, we learned that implementing these techniques can be very difficult and lead to varied results depending on the series of steps that are followed. All three approaches followed the same basic principles, however, the results varied significantly as shown above. OpenCV provided a wide range of processing steps to choose from, however, the abundance of various functions available made it very difficult to implement 3D reconstruction effectively. Approaches 1 and 2 kept these considerations in mind by avoiding the use of OpenCV wherever possible to make the debugging process easier and creating a more controlled set of results.

Authors' Contributions

Neural Network Solution Research: Collaborative
    Neural Network Architecture and Research: Aidan
    Neural Network Dataset and Implementation Research: Predrag (Pedja)
Video Slides: Armaan
Video: Aidan, Pedja, Armaan
Abstract: Aidan, Armaan
Introduction: Pedja, Armaan
Methodology, Results, and Experiment: Collaborative → individual contributions discussed
Conclusion: Collaborative

References

[1] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction," *Computer Vision – ECCV 2016 Lecture Notes in Computer Science*, pp. 628–644, 2016.

[2] X. Han, H. Laga, and M. Bennamoun, "Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2019, doi: 10.1109/tpami.2019.2954885.

[3] C. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction," Apr. 06, 2016. https://arxiv.org/pdf/1604.00449.pdf.

[4] “Camera Calibration and 3D Reconstruction¶,” *OpenCV*. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_table_of_contents_calib3d/py_table_of_contents_calib3d.html. [Accessed: 18-Dec-2020].

[5] N. Nesic, “2014 Stereo Datasets with Ground Truth,” *Middlebury*, 2014. https://vision.middlebury.edu/stereo/data/scenes2014/ (accessed Dec. 16, 2020).

[6] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. New York: Cambridge University Press, 2017.

[7] O. Padierna, “Stereo 3D reconstruction with openCV using an iPhone camera. Part I.,” *Medium*, 10-Jan-2019. [Online]. Available: https://becominghuman.ai/stereo-3d-reconstruction-with-opencv-using-an-iphone-camera-part-i-c013907d1ab5. [Accessed: 18-Dec-2020].