Mini Project #2: 3D Stereo in the Dark

Alex Yu (yualex@pdx.edu)

1 Introduction

A stereo 3D system is based on binocular vision. This in a way mimics the way human eyes perceive depth. Using two cameras-which are placed at a fixed distance apart from each other-captures the same scene from slightly different angles. These two images contain disparities in the positions of the objects relative to each other. By analyzing these disparities through stereo matching techniques, the system constructs a disparity map, which represents the depth information for each pixel in the scene. Using algorithms, the disparity map is converted into 3D coordinates which provides a reconstruction of the scene that is aware of depths.

It is challenging to do stereo in low-light conditions for various of reasons. Many of the issues were actually encountered during this project such as a low signal-to-noise ratio, where low light introduced significant noise in the images; reduced scene information; and poor feature matching, where the algorithm struggled to identify corresponding points between the left and right images.

Despite all of these challenges, we are motivated in achieving stereo 3D vision in low light for several reasons. Some could be how some environments, such as nighttime, underwater, or any poorly lit areas, cannot be artificially brightened due to numerous constraints that could apply. Another reason is how depth information in these environments is essential for tasks where decisions must be made in real time or where accuracy is crucial.

Some real-world applications where this is particularly important include self-driving cars operating at night, nighttime surveillance and security systems, and search and rescue operations in low-light conditions.

2 Methods

The approach I took to finding the optimal stereo 3D algorithm parameters for this project was a step-by-step process to fine-tune the settings. The training set I used consisted of selected images from the 2021 Middlebury Stereo Dataset recommended to us.

To start, I simulated a low-light environment on my test set for both left and right images. I chose not to include much noise to first observe the effects of lighting alone. After this, I brightened the images and added noise to compare the results.

Next, I implemented ORB (Oriented FAST and Rotated BRIEF) for feature detection and matching to evaluate its performance in identifying corresponding points between the left and right stereo pairs. The Brute-Force matcher with Hamming distance was used to match descriptors and assess disparity consistency.

After that, I explored two block-matching techniques: StereoBM, a fast, hardware-friendly algorithm for initial disparity estimation, and StereoSGBM, a more advanced approach incorporating cost aggregation for smoother, more accurate disparity maps. I used a grid search to optimize key parameters, including numDisparities, blockSize, and P1 and P2. However, I ultimately focused on StereoCNN, as it provided the best performance through deep learning, as suggested by the assignment.

For StereoCNN, I implemented it to predict disparity maps directly from the stereo pairs. The network architecture followed an encoder-decoder design. Like StereoBM and StereoSGBM, I used a grid search to optimize hyperparameters such as learning rate, batch size, and number of epochs. For the loss function, I tested several options to improve disparity prediction, including MSE, Smooth L1 Loss, and Edge-Aware Smoothness Loss.

Finally, before calculating the RMSE value between the predicted disparity map and ground truth, I displayed all disparity maps to visually assess their quality. After that, I calculated the RMSE for all images and took the average, which is shown below in the results.

3 Results

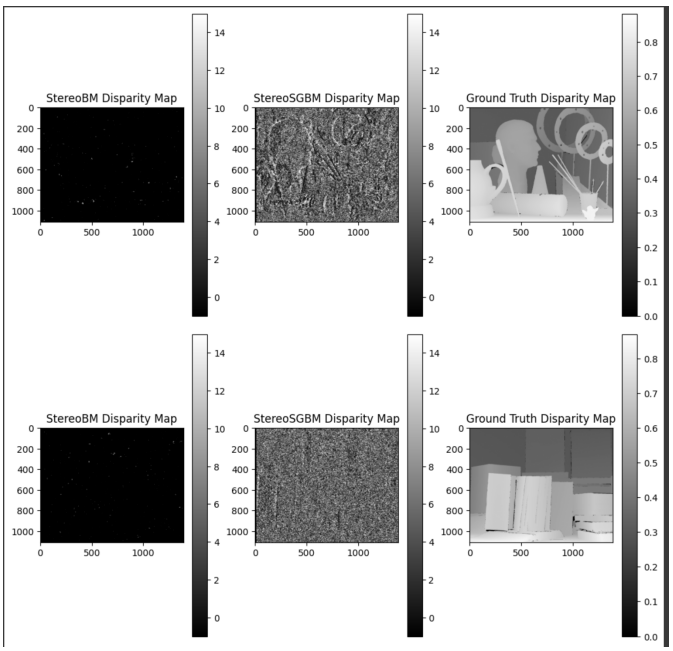
The results for this project were produced through the following steps:

- Train 3D reconstruction algorithm using a separate image dataset to find the best parameters for that specific algorithm.
- Apply and display our separate image dataset to visually see our results along with its average RMSE value.
- Using the same trained algorithms above, apply and display our official test images to finally view our end results.

3.1 Feature Matching

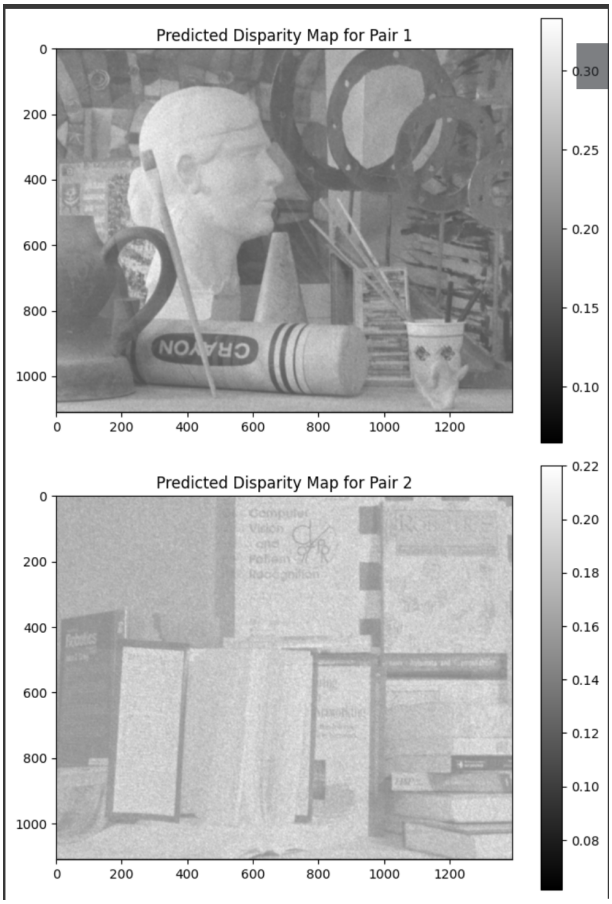
(a) Comparison between less noise and light vs. more noise and light

3.2 StereoBM and StereoSGBM

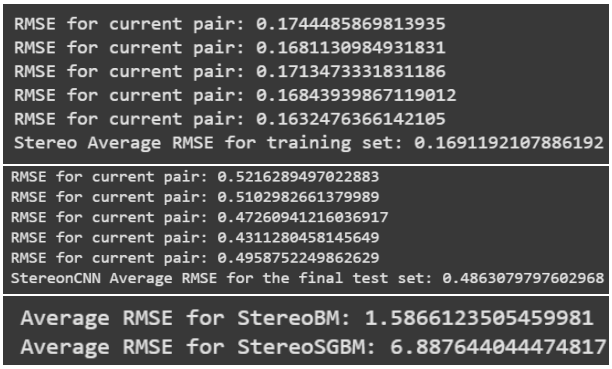


(a) More can be found in "results/stereoBM and SGBM" folder

3.3 StereoCNN



(a) Image results can be found in "restormer/restored" folder along with its comparison in "restormer/degraded



(a) Average RMSE values between StereoCNN training image set, StereoCNN final image set, and StereoBM and SGBM final image set

4 Conclusion

Out of all the images and algorithms tested throughout this entire project, the best results shown was from the Stereo custom neural network (StereoCNN). With the best average RMSE throughout the board of 0.486. This could also be directly seen in the predicted disparity maps displayed above where they are significantly more clear compared to the other algorithms used.

After that, StereoBM and StereoSGBM provided values that were no where close to what CNN provided. This can also be directly seen in the visualization as well. This could possibly be a result of how I tuned the parameters or possibly missed a normalization somewhere within my code.

When using less light and noise, I noticed that some feature mapping did because it could not find any matching features. This can be seen in comparison to more light and noise in section 3.1. I believe that this could have made an impact on some of the RMSE values.

I was not surprised that the best results came from StereoCNN but I was shocked to see how much better it was. While neural networks are considered the state-of-the-art algorithms right now, meaning they're likely the best we got right now, I did not predict that it could be this significant of a difference. However, I do believe that there are some errors with the way I possibly implemented of the other algorithms.

Early on in the project, I focused much of my time constructing the custom neural network. I also had access to google colab GPU hardware accelerator during this time. This drastically improved the processing, training, and overall accuracy of the disparity predictions specifically for StereoCNN. It wasn't until after I started the implementation of other algorithms that I realized I lost access to GPU hardware acceleration. Because of this, training and fine tuning the parameters for the other algorithms took a significantly more time and I wish I had put more time into those.

I learned a lot during the project since it exposed me to a lot of resources needed to reconstruct a 3D scene. Similar to mini-project 1, it seems to be very difficult to actually do this perfectly. There are so many different approaches you can take in designing algorithms.

Some things that didn't work as expected were a lot of conversions that needed to be made to much of the files in order for it to be able to compute. Many sizes of all the images were not the same so it made computing RMSE and disparity maps difficult.

One of the things I wanted to try more was to really narrow down the parameters of other 3D constructing algorithms with GPU hardware acceleration. Having this tool made a drastic difference in this project. I believe having and using this tool more would've drastically effected my approach to the project.