Evaluate depth estimation technologies (Stereo and LiDAR) - Zhejin Huang

Team member: Zhejin Huang

Contents:

1. Background

Depth estimation technologies enable devices to better understand the surrounding environment and provide information for Augmented Reality use cases. For those use cases, an ideal depth estimation technology should have good quality (low error, high recall), high performance (low latency) and low power consumption. To enhance industry experience, it would be a good start to evaluate and understand the strengths and limitations of Stereo and LiDAR based depth estimation technologies with respect to quality, performance and power.

2. Technologies and Metrics

Stereo and LiDAR based depth estimation are widely used in the industry. In this project I evaluated state-of-the-art Stereo algorithms from OpenCV and LiDAR based pipeline in iPhone. These pipelines are mature enough for the study purpose.

2.1 Stereo evaluation

2.1.1 Stereo algorithms

For Stereo based depth estimation, I evaluated OpenCV's StereoBM and StereoSGBM algorithms [1]. Both algorithms take two rectified gray scale images from left and right cameras and output a disparity map from Left image point of view. StereoBM implements a basic block matching algorithm with SAD-based (Sum-of-Absolute-Differences) loss function. StereoSGBM (Semi Global block matching), which is a more advanced algorithm than StereoBM, assumes a disparity map to have spatial continuity. It adds to the loss function the disparity map's discontinuity and uses dynamic programming to search for the best results. With this assumption, the accuracy is improved while it takes more time to search. Below are diagrams to explain the pipeline of StereoBM and StereoSGBM.

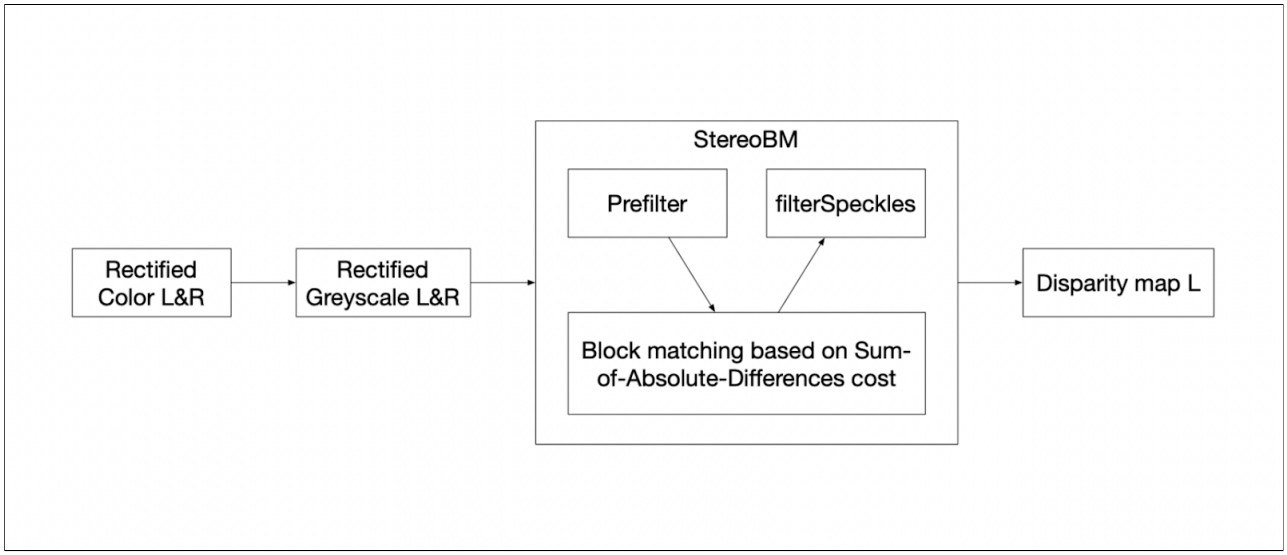


Figure 1. A diagram to explain OpenCV StereoBM algorithm's pipeline.

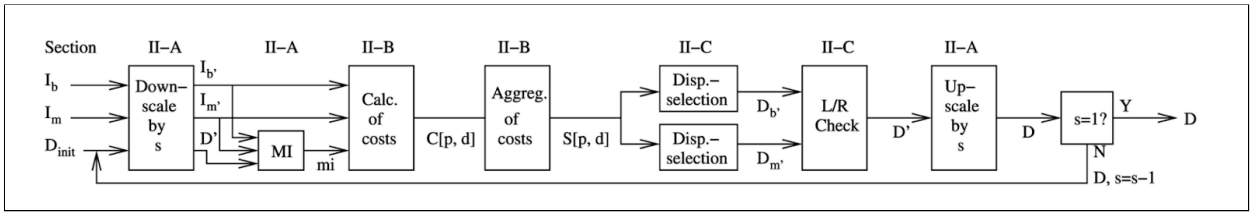


Figure 2. A Diagram to explain OpenCV StereoSGBM algorithm. [2]

2.1.2 Dataset

The middlebury 2021 dataset [3] has 24 pairs of left and right rectified color images and ground truth disparity maps. Loading the color images as grayscale from OpenCV will get the inputs for the algorithms. To evaluate the outputs of the algorithms, which are of 4-bit fixed point format, each disparity value should be divided by 16. To decode the ground truth disparity map by using the header that contains endianness and scale factor, I reorganized a script [4] I found online to pmf2tiff.py in my repo. It decodes the pmf ground truth disparity maps and saves them as directly usable tiff files. After these pre-processings, a ground truth disparity map's value is invalid if it is infinity, otherwise valid; and an output disparity map's value is invalid if it is negative, otherwise valid.

2.1.3 Metrics

To evaluate the quality, I use the relative (absolute) error averaged on pixels that are both valid on output and ground truth disparity maps. I also calculated the recall (#output & ground truth valid / #ground truth valid) to make sure the output covers enough area. Note that it is also important to look at the output disparity map to see how reasonable the results are.

To evaluate the performance, I use latency (wall time) in milliseconds.

2.1.4 Device

The algorithms are running on an Apple M1 Max device with macOS.

2.2 LiDARDepth App

For LiDAR depth estimation, I'm using the LiDARDepth App [5] on iPhone 14 Pro Max to get a depth map. A screenshot of the app is shown below. The depth map is rendered by the app with blue color representing near and red color representing far.

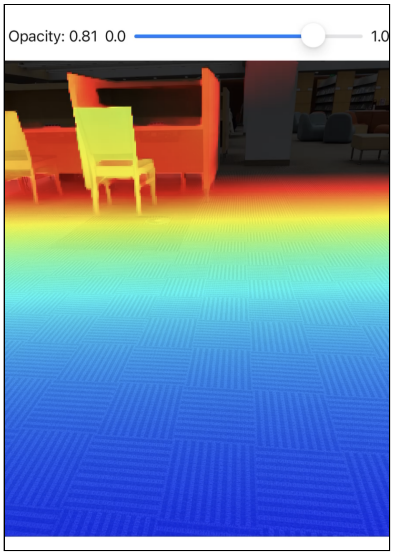


Figure 3. The heat map from LiDARDepth App.

For iPhone LiDAR accuracy evaluation, an efficient way to evaluate is to set up several scenes and use ruler measurement as ground truth. An experiment is done by IT-JIM and the results are shown below. We can see that the error rate is below 0.5% within 4.5 meters and goes up to ~5% at 5 meters.

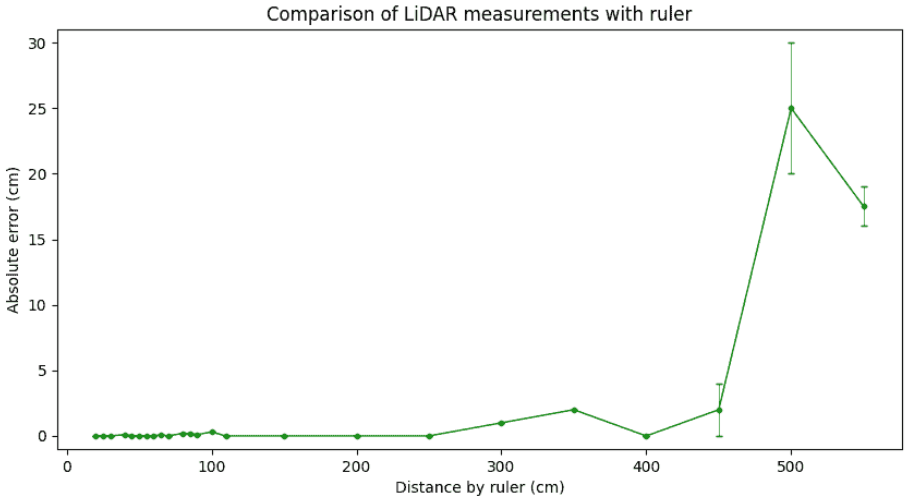


Figure 4. iPhone 12's LiDAR accuracy measurement done by IT-JIM [7]

For performance measurement it is much harder with limited instrumentation and disclosure. The developer of this app sets up the LiDAR camera configuration and outputs configuration through AVFoundation [6] framework (a framework that handles audio and vision streams on Apple devices) one time during app setup time and receives the depth data per-frame and renders it as the heat map shown above. Given the fact that the implementation of AVFoundation is hidden, it is hard to measure the latency of each component. However it is easy to measure the frame rate from the application side (by adding a timer), which is 30fps, meaning that the maximum latency of each component individually is less or equal than 33ms (assuming sequential execution and no latency hiding technologies).

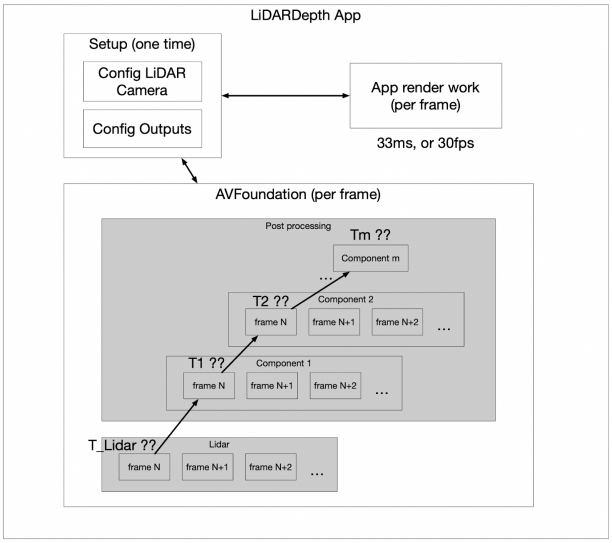


Figure 5. LiDARDepth App pipeline.

3. Experiments and Results

3.1 Stereo Experiments

Block size (the window size for loss function) and number of disparities (the number of pixels to search for a match on the horizontal epipolar line) are 2 most important parameters for the Stereo algorithms. For block size, we want to decrease it for better recall and more details, but we also want to increase it to reduce noise. For number of disparities, we want to decrease it for less compute and lower latency but increase it for better search results. I did experiments to tune these parameters to get acceptable (overall optimal) results on all these metric we care about: latency, accuracy and recall.

3.1.1 StereoBM

Evaluating a large range of blockSize (from 5 to 241) and numDisparities (from 16 to 256) on 1 data (to save time), I got the results below as 3D plots. The result shows that latency linearly increases with numDisparities and it also has a positive correlation with blockSize. For error rate, it shows poor results on small block sizes (lower than 20). For recall, it is good to have blockSize lower than 100 and numDisparities higher than 100. The red highlighted regions for each metric are desirable.

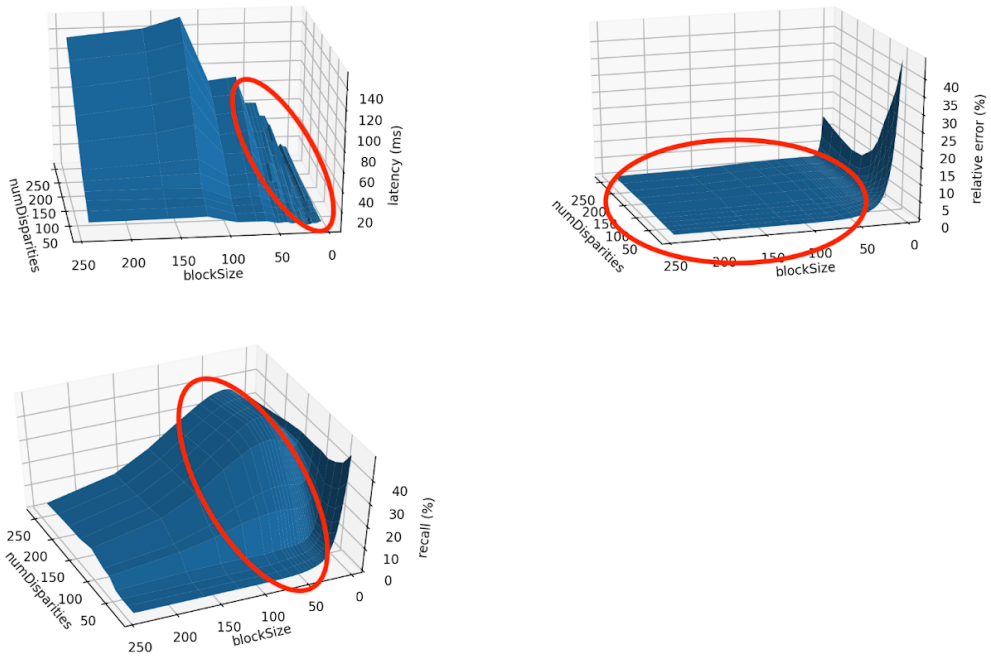


Figure 6. Plots of latency (ms), error (%) and recall (%) of StereoBM on a large region of blockSize and numDisparities. Evaluated on 1 data.

To study a bit more on the intersection of these 3 regions, I evaluated with blockSize from 15 to 25 and numDisparities from 128 to 192 on all the 24 data. From the results below, having numDisparities=176 and blockSize=21 gets relatively good accuracy, recall and performance.

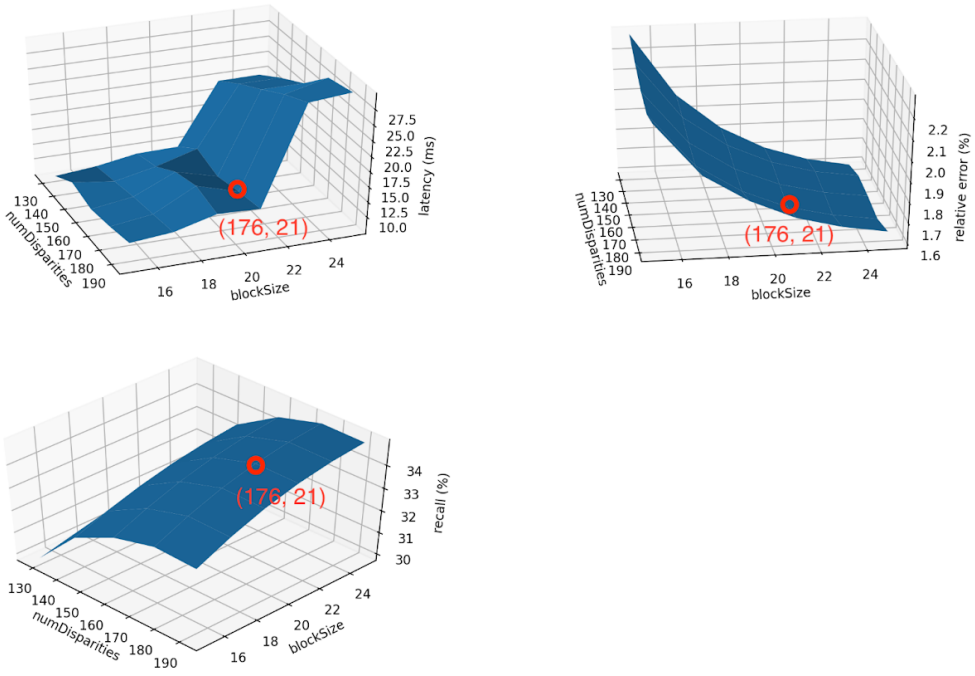


Figure 7. Plots of latency (ms), error (%) and recall (%) of StereoBM on a small region of blockSize and numDisparities. Evaluated on 24 data.

3.1.2 StereoSGBM

Similar to the experiments for Stereo_BM, I did a search on a larger range of blockSize (from 5 to 201) and numDisparities (from 16 to 320) on 1 data and got the results below. For performance, latency is linear to numDisparities and not much correlated with blockSize. For accuracy, the error rate is very sensitive to blockSize with the optimum around 20-30. For recall, we want to avoid the region where block size is around 30 and numDisparities below 100.

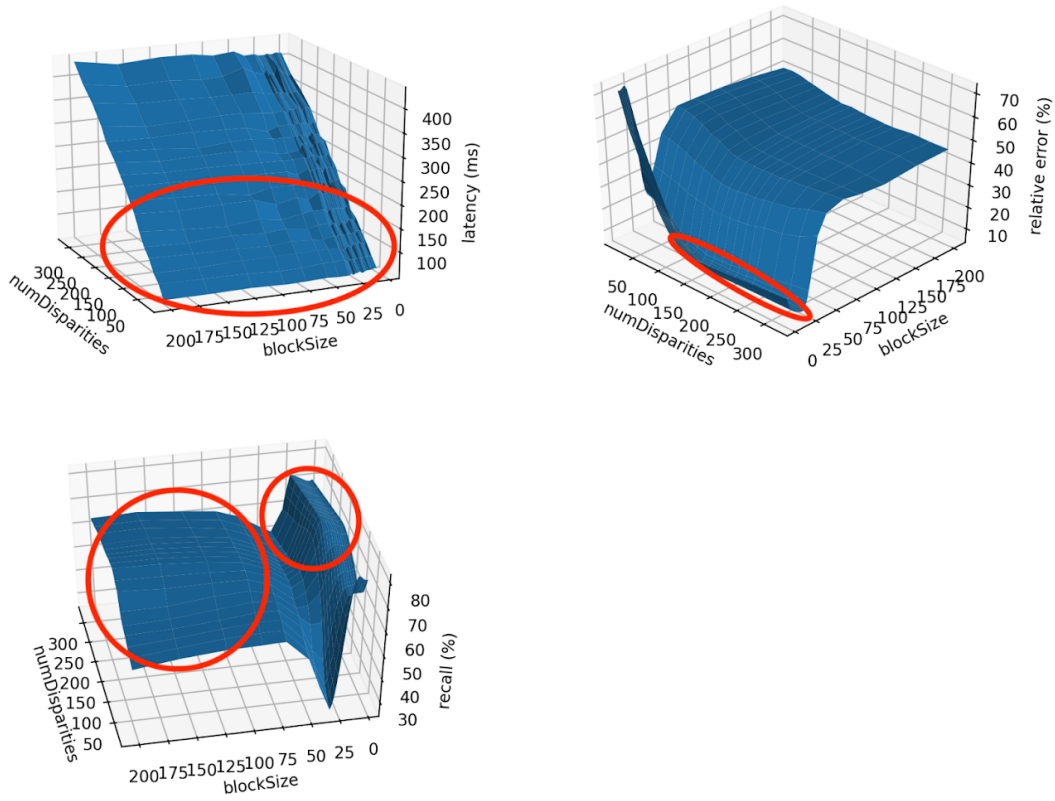


Figure 8. Plots of latency (ms), error (%) and recall (%) of StereoSGBM on a large region of blockSize and numDisparities. Evaluated on 1 data.

To get more precise results, I ran the pipeline on 24 data with numDisparities ranging from 128 to 192 and blockSize ranging from 19 to 37. And got the results below. Having numDisparities=128 and blockSize=21 gets relatively good accuracy, recall and performance.

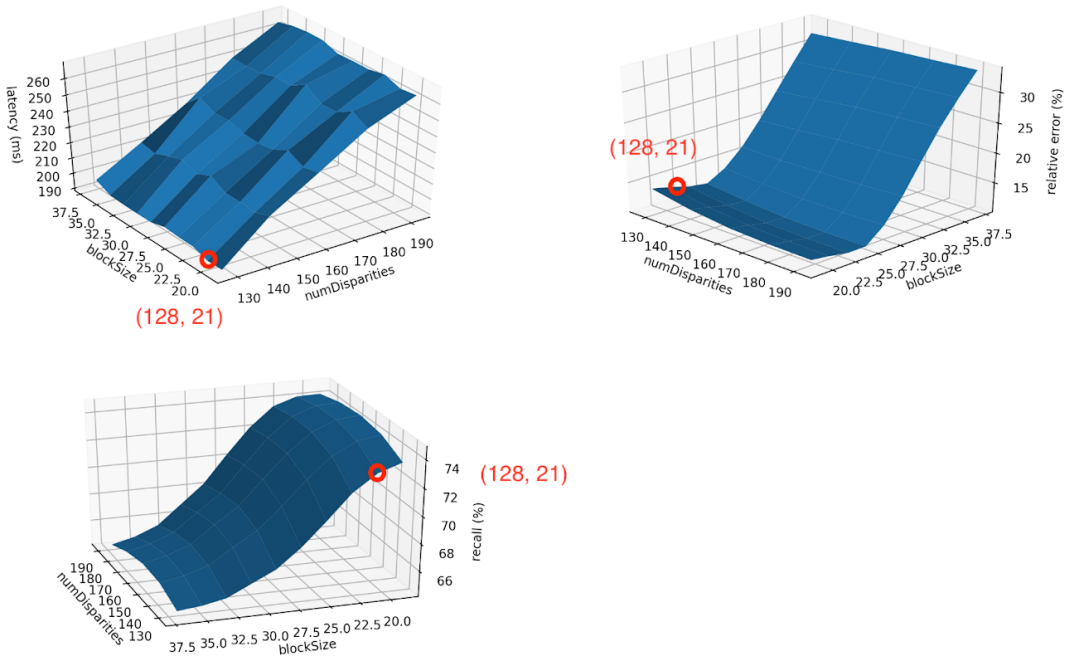


Figure 9. Plots of latency (ms), error (%) and recall (%) of StereoSGBM on a small region of blockSize and numDisparities. Evaluated on 24 data.

3.1.3 Look at disparity maps

(1) Optimal StereoBM and StereoSGBM results

Besides looking at the metrics, it is also important to look at the disparity maps to tell whether the result makes sense. By comparing BM and SGBM disparity maps, we can see that although BM is more accurate than SGBM, the recall is pretty low, missing a lot of information, especially on horizontal edges. SGBM has a better recall but less accuracy, also the latency is 16 times the latency of BM.

Figure 10&11. Left, right color image from middlebury 2021 artroom2 [3].

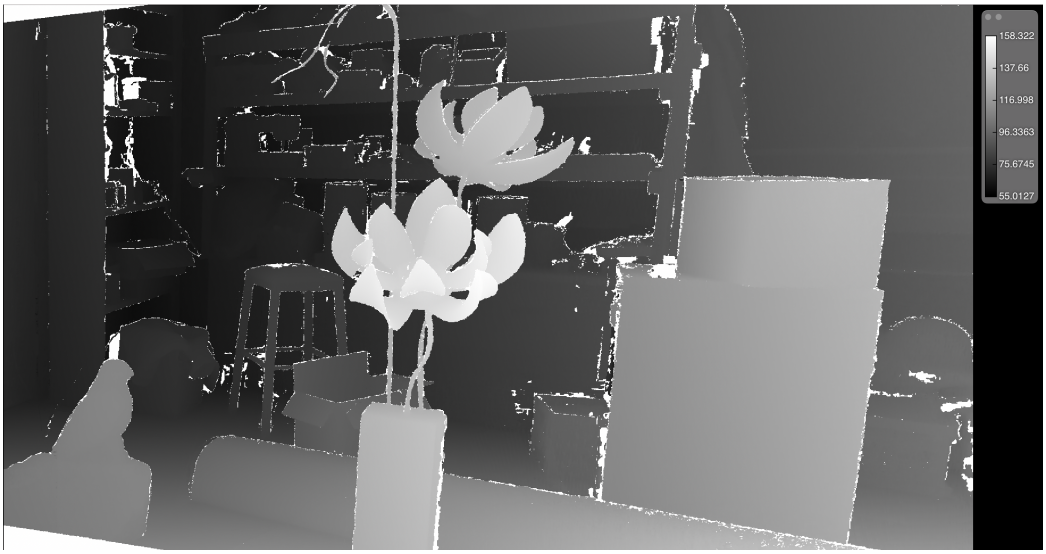


Figure 12. Ground truth disparity map from middlebury 2021 artroom2 [3] after preprocessing. Note that the disparity maps below share the same scale bar, while invalid disparities here are visualized as pure white area (the dataset use `inf` as invalid) and invalid disparities below are visualized as pure black area (the algorithms output `-1` as invalid)

Figure 13. Optimal result (with numDisparities=176 and blockSize=21) of StereoBM, with 12.0 ms latency, 1.68% error rate and 34.56% recall.

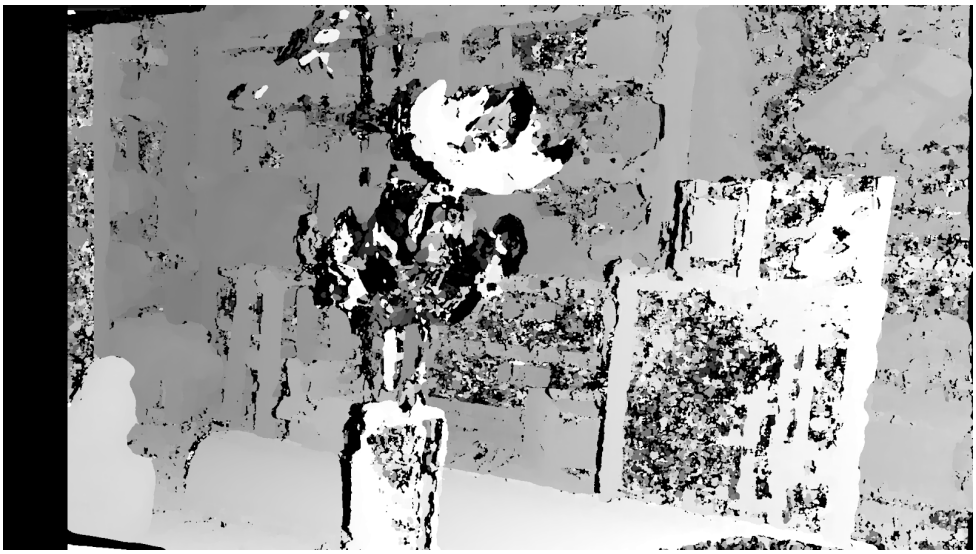


Figure 14. Optimal result (with numDisparities=128 and blockSize=21) of StereoSGBM, with 192.3 ms latency, 13.25% error rate and 73.01% recall.

(2) Non-optimal StereoBM and StereoSGBM results

By comparing the optimal results with non-optimal ones, it is clear that larger blockSize reduces local noise but loses recall and that larger numDisparities improves both noise and recall but hurts performance, and that the optimal parameter settings get reasonable results. Also by looking at the optimal results, StereoSGBM gets a more reasonable disparity map then StereoBM (low recall, missing horizontal edges...).

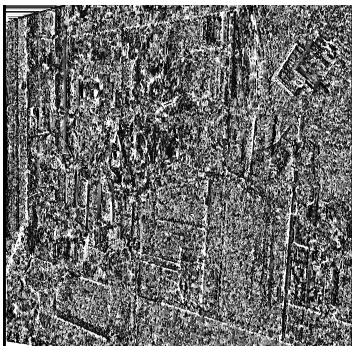
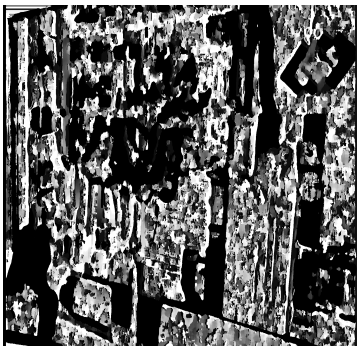
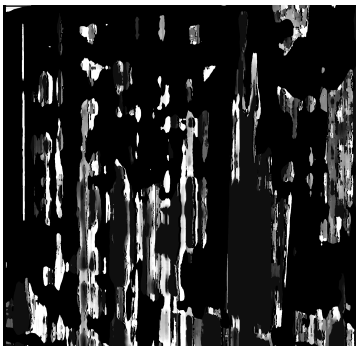
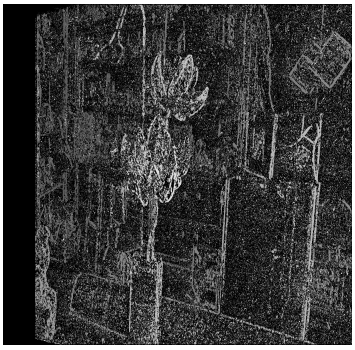
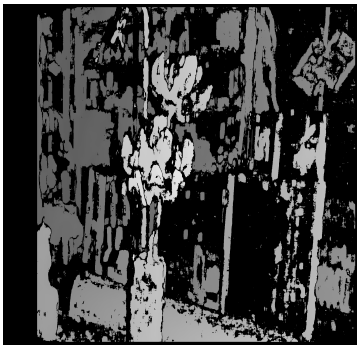
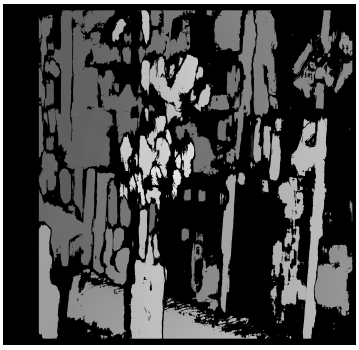
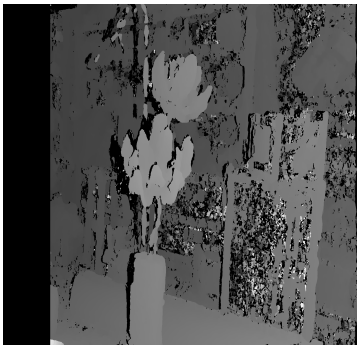
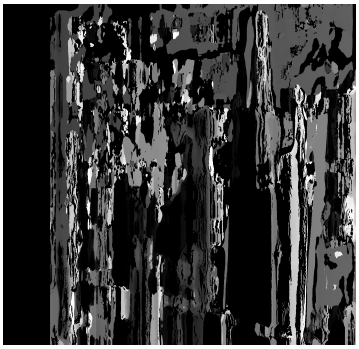
nDisparities\bSize	5	21	41
16			
176			
256			

Table 1. StereoBM disparity maps with small/optimal/large nDisparities/bSize values.

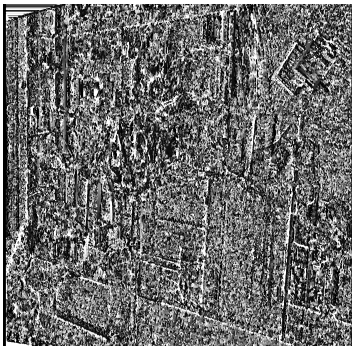
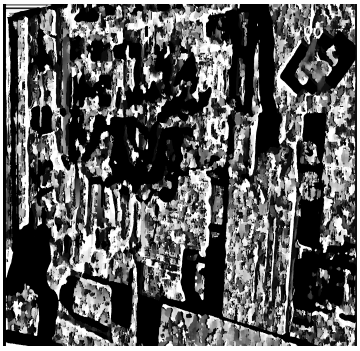
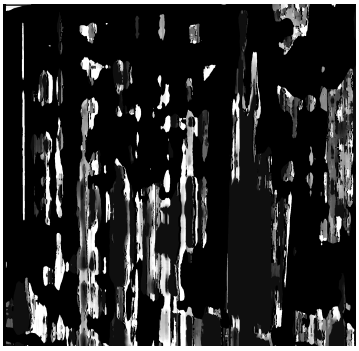
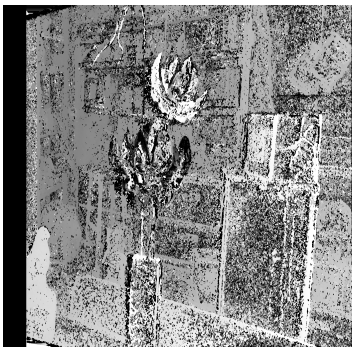
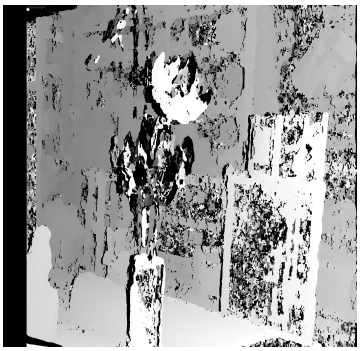
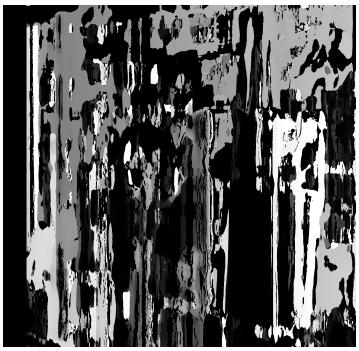
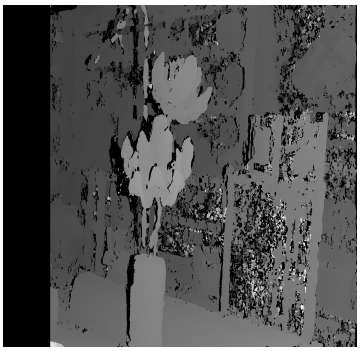
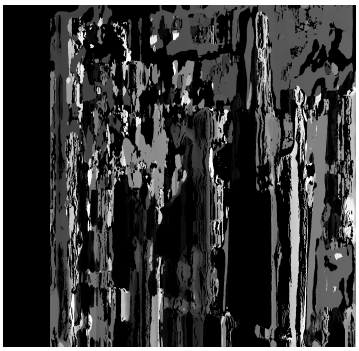
nDisparities\bSize	5	21	41
16			
128			
256			

Table 2. StereoSGBM disparity maps with small/optimal/large nDisparities/bSize values.

4. Conclusions

By comparing the Stereo results with LiDAR, we can see that LiDAR on iPhone has higher quality (lower error rate and higher recall) than StereoBM and StereoSGBM. The latency of the iPhone LiDAR pipeline is unknown but it could successfully run on a 30fps application, which is at least better than StereoSGBM.

Method \ Metric	Latency	Error rate	Recall
StereoBM	12.0 ms	1.68%	34.56%
StereoSGBM	192.3 ms	13.49%	73.01%
iPhone LiDAR	Best guess: <= 33 ms on each component	~0.5% below 5m 5% at 5m	~100%

Table 3. The comparison between 3 methods.

Note that disparity map's error rate is converted to depth map's error rate using: ((1- (1 / (1 + error_rate))) + ((1 / (1 - error_rate)) - 1)) / 2

5. Future works

5.1 Use the same scene to evaluate LiDAR and Stereo

5.2 Detailed level Performance analysis

This study measures the performance for a whole pipeline. It is better to break down to a more detailed level and focus on the bottlenecks. For Stereo pipelines, we could break it down to function level and look at how each function consumes CPU resources. For the Lidar pipeline, it involves Lidar, CPU and some components on other hardware.

5.3 Dedicated device with fixed states

To measure performance precisely, it is good to profile on a dedicated device without much noise from other processes (less contention) and dynamic hardware frequency (fixed cycle rate).

5.4 Power analysis

The study mainly focuses on accuracy and performance, whereas power consumption (measured in Watts or hardware cycles) is also a meaningful aspect to evaluate. However, this analysis is limited by hardware instrumentation.

5.5 Evaluate Stereo pipelines on iOS device

To control the platform for power and performance evaluation, it is necessary to evaluate Stereo and Lidar pipelines on iOS devices. This requires setting up OpenCV environments on iOS.

5.6 Evaluate Stereo pipelines on GPU (CUDA)

StereoBM and StereoBM are already sped up using SIMD, but this is far from ideal since these algorithms are able to be highly parallelized. From the inheritance graph below, we can see that StereoBM has a GPU version cv::cuda::StereoBM and there are two other algorithms that only runs on GPU, which are cv::cuda::StereoBeliefPropagation and cv::cuda::StereoConstantSpaceBP. Unfortunately, we cannot use CUDA on iOS devices.

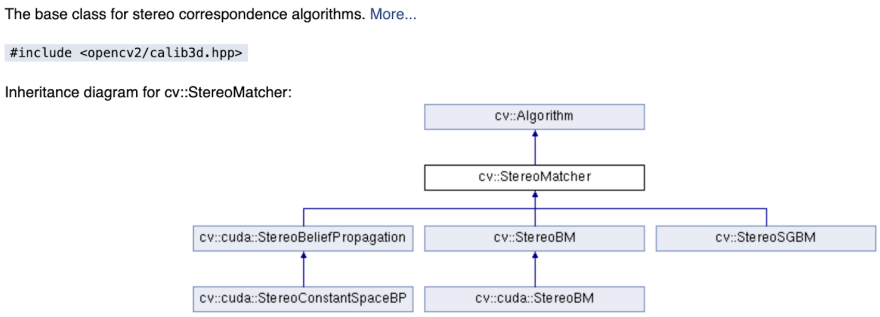


Figure 15. OpenCV Stereo Matching algorithms' inheritance tree. [8]

6. Materials

Stereo depth evaluation code: https://github.com/YHHHCF/psych221_zhejinh

A copy of slides: Zhejin_presentation.pdf

7. References

[1] https://docs.opencv.org/3.4/d9/dba/classcv_1_1StereoBM.html
[2] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 2, pp. 328-341, Feb. 2008, doi: 10.1109/TPAMI.2007.1166.
[3] https://vision.middlebury.edu/stereo/data/scenes2021/
[4] https://github.com/wu258/Middlebury-Stereo-dataset-pfm-to-png
[5] https://developer.apple.com/documentation/avfoundation/additional_data_capture/capturing_depth_using_the_lidar_camera?language=objc
[6] https://developer.apple.com/av-foundation/
[7] https://www.it-jim.com/blog/iphones-12-pro-lidar-how-to-get-and-interpret-data/
[8] https://docs.opencv.org/3.4/d2/d6e/classcv_1_1StereoMatcher.html