Depth Map Generation on More Realistic Scenes

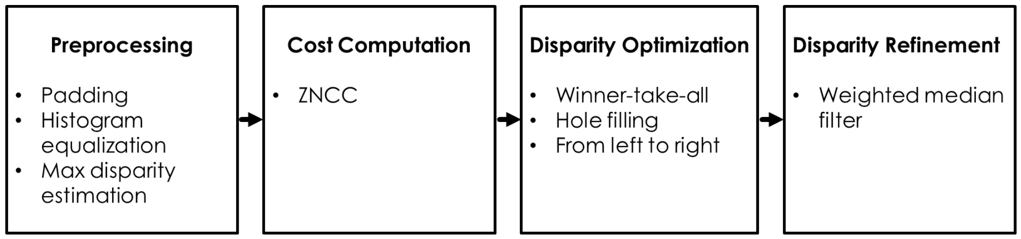
R07942154 林暄富 R08942112 羅政捷 R08942073 吳柏潤 R07942073 吳奕萱

1. Introduction

The goal of the project is to generate depth map for images of synthetic and real scenes. The performance of stereo matching algorithms for generating depth maps depends on the max disparity of the input images, yet this information is not directly provided, which suggests that a reliable estimation of the max disparity is needed. The nature of real data also poses challenges. The disparity between left and right input data may be negative, perhaps because the camera did not move horizontally. Moreover, repetitive patterns in real images may cause confusion, which leads to details not shown or blurred contours in the output disparity map.

To solve these problems, first, feature matching is implemented to estimate max disparity. Then, stereo matching is performed with traditional methods (enhancement of HW4) and deep learning models (PSMNet). Results of stereo matching methods are compared, and the non-learning approach shows advantage by having lower average error, faster computation speed, and better disparity maps of real scenes.

2. Architecture



3. Algorithm

(1) Preprocessing

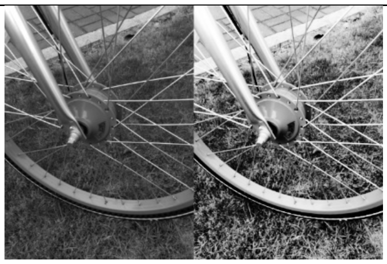
a. Data preparation

Padding (reflect type, size 32) and color conversion from rgb to gray

b. Histogram equalization

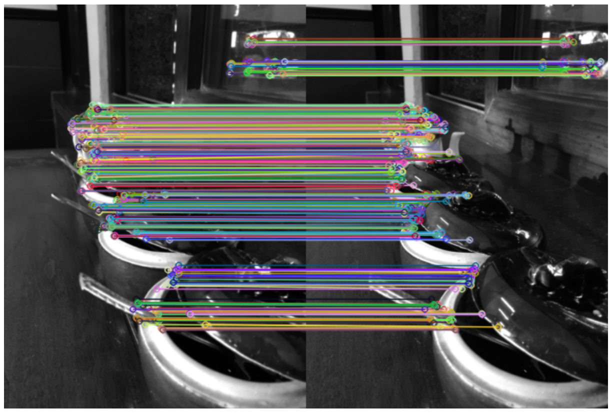
When the intensity and contrast of an image pair are not comparable, cost computation based on intensity may be erroneous. Thus, histogram equalization is performed on each image, to normalize the intensity values as well as enhance contrast of some images. The figure below shows processed image pairs.

RD0	RD1	RD2

		
RD3	**RD4**	**RD5**
RD6	**RD7**	**RD8**
RD9		

c. Max disparity estimation by feature matching

Max disparity estimation was implemented so that the program doesn't have to go through every displacement until it reaches a given upper limit. The method we use is to match the features of two images, and pick out the pair with maximum horizontal displacement. We utilize Feature descriptor Oriented FAST and Rotated BRIEF (ORB), as it is a binary detector that can detect feature points invariant to scale and rotation [1]. We then use brute-force matcher to find the feature pairs. Next, Random Sample Consensus (RANSAC) was applied to get rid of the outliers. Finally, we find the pair with the maximum horizontal displacement.



Feature matching with TL0/TL1. (maximum horizontal displacement: 8)

Since the feature detection method might not pick out the feature pair with the max disparity, we added a safety margin (16) to the maximum horizontal displacement, and that's the estimated max disparity.

(2) Disparity map computation

a. Cost computation

Template matching with Zero mean normalized cross-correlation (ZNCC), size 17x17, is applied to find similar points in the two images. ZNCC shows better performance than simply taking the sum of squared distances of pixel values.

b. Cost aggregation

We tried to integrate information of neighboring pixels with similar values by applying filters. However, applying guided filter (with the left image as guide), median blur, and box filter did not show better results, as the contours became too blurred.

c. Disparity optimization

Disparity is optimized with a winner-take-all approach. Since the goal is to generate the disparity of the left image, regions on the left side of objects are prone to problems. To deal with issues of these regions, we fill the values from left to right. If two pixels in the left image correspond to the same pixel in the right image, we take the value of the left pixel. For holes with no corresponding points, the first non-zero disparity value on the left is taken.

While generating disparity maps on real data, we notice that the disparity between the left and right image pairs may have negative values, perhaps because the camera did not move horizontally when the two images are taken. Assuming that the negative values mostly occur at regions with greater depth, we fill them with a small positive value that approaches zero.

d. Disparity refinement

Weighted Median filter has the ability to remove outliers while preserving edges and structures of the image [2]. Here we apply weighted Median filter with radius 31.

4. Results

(1) Computation time (for 10 images)

	Real	**Synthetic**

Time (sec)	132.69 sec	238.01 sec

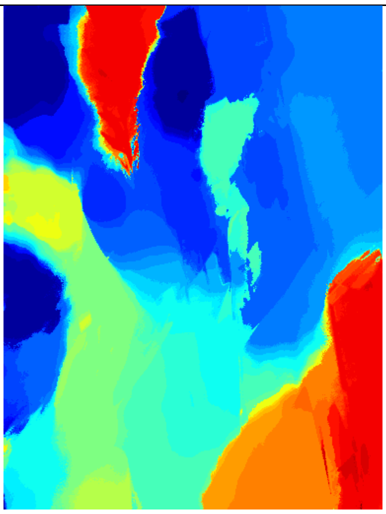
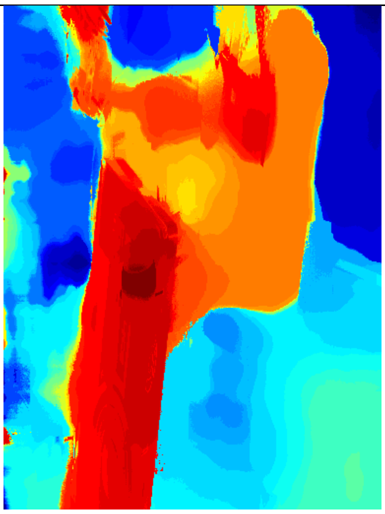
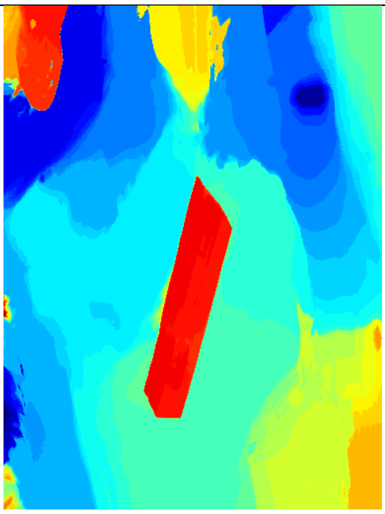
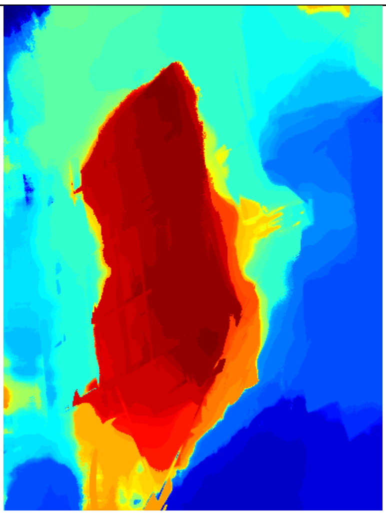
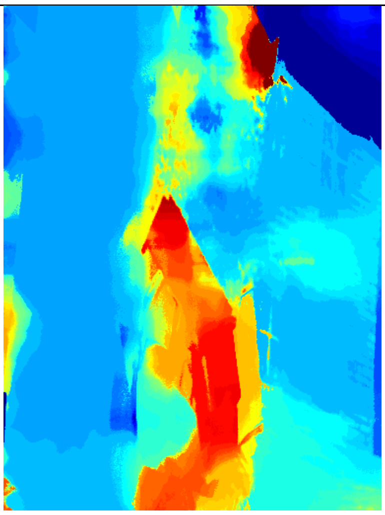
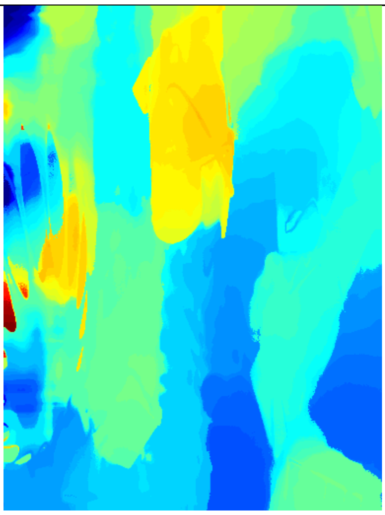
Machine specs: MacBook Pro 2017 (2.3GHz dual- core Intel Core i5, Turbo Boost up to 3.6GHz, with 64MB of eDRAM)

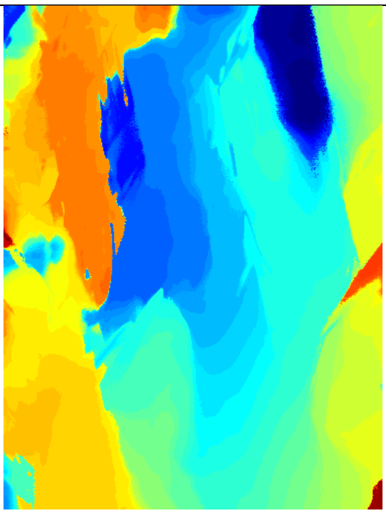
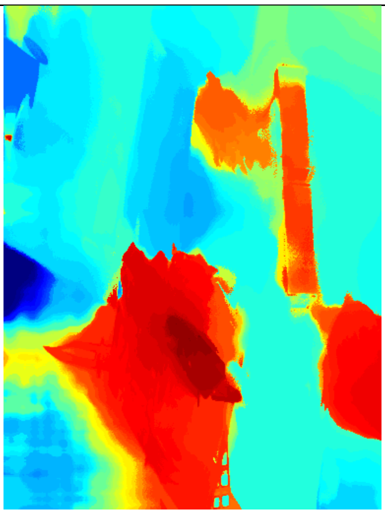
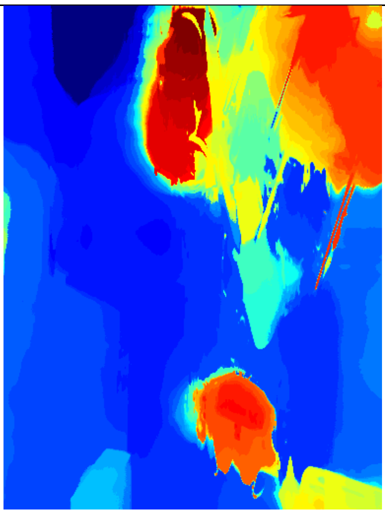
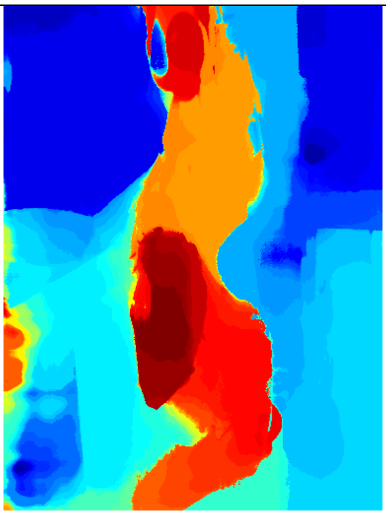
(2) Average error of the synthetic scenes

	SD0	**SD1**	**SD2**	**SD3**	**SD4**	**SD5**	**SD6**	**SD7**	**SD8**	**SD9**	**Avg**
Error	2.96	3.21	2.07	4.26	5.83	4.53	3.53	4.44	3.22	3.34	3.74

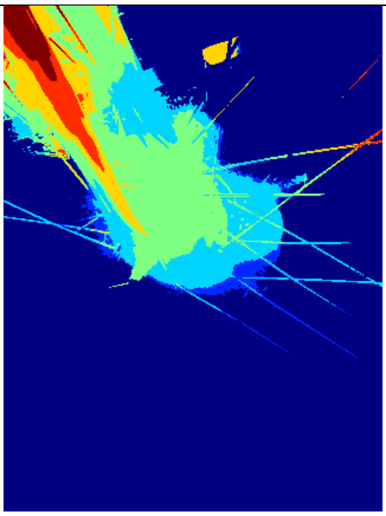
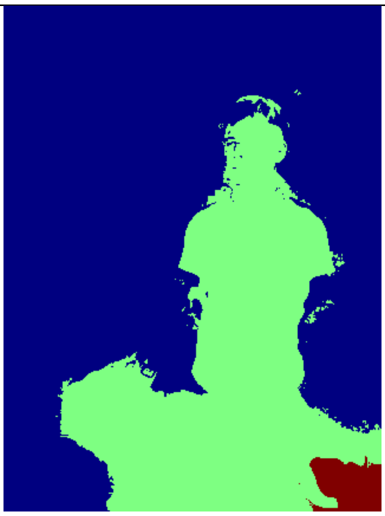
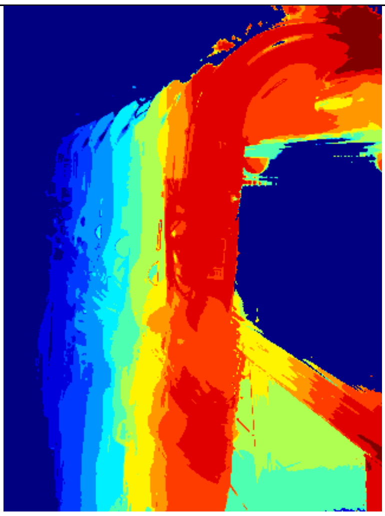
(3) Disparity map

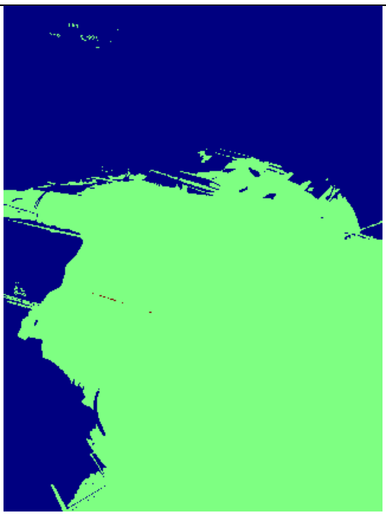
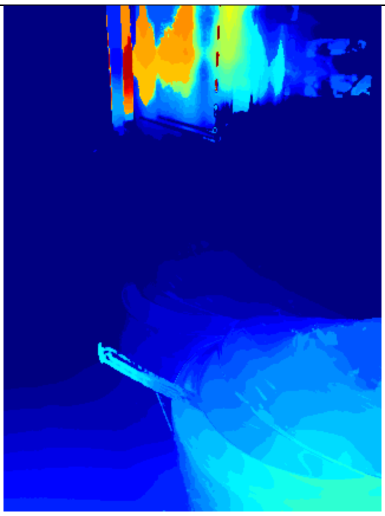
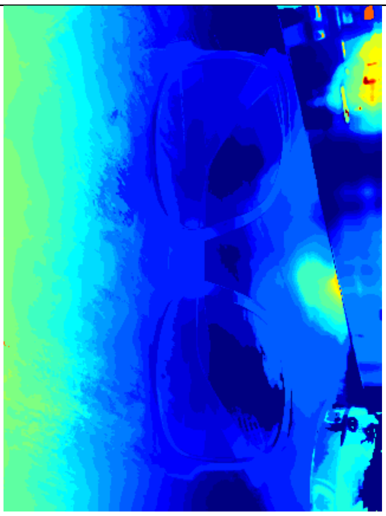
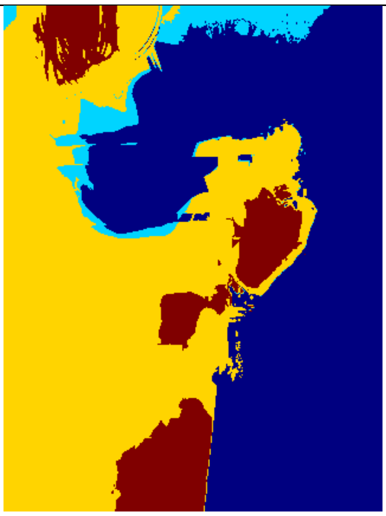
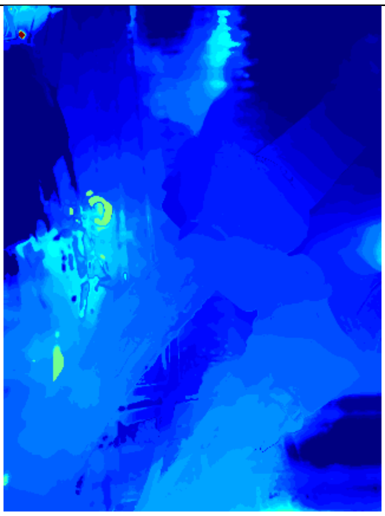
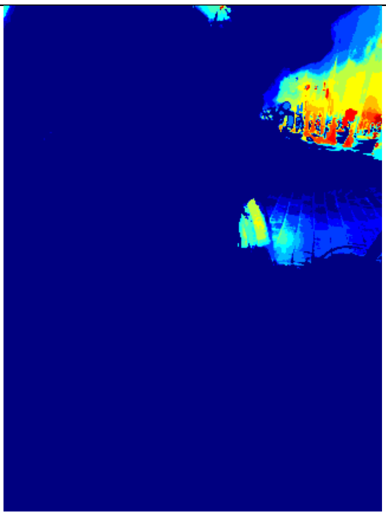
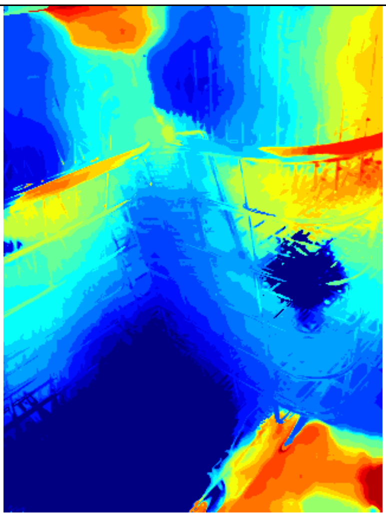
Synthetic scenes

SD0	**SD1**	**SD2**
		
SD3	**SD4**	**SD5**
		
SD6	**SD7**	**SD8**

		
SD9		
		

Real scenes

RD0	**RD1**	**RD2**
		

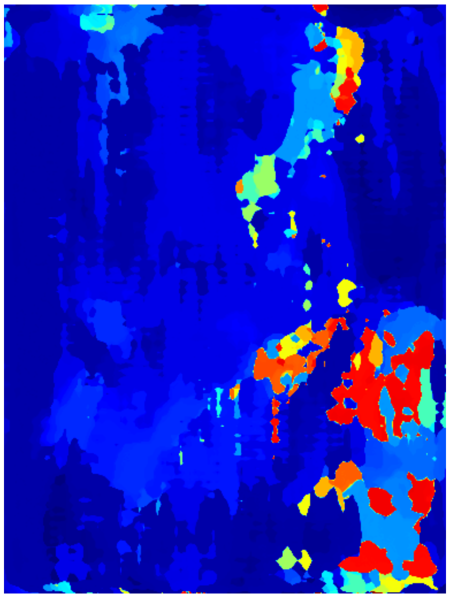
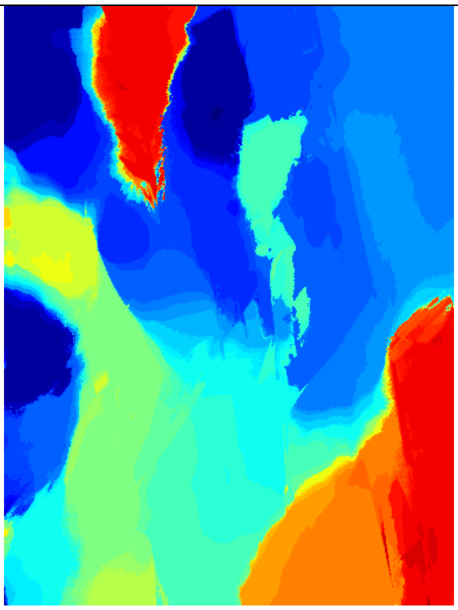
RD3	RD4	RD5
		
RD6	**RD7**	**RD8**
		
RD9		
		

5. Discussion

(1) PSMNet vs. our method

Pyramid Stereo Matching Network (PSMNet) has shown advantages in capturing global context information. The network consists of convolutional neural networks (CNN) for a larger receptive field, spatial pyramid pooling (SPP) modules to capture spatial information of different scales, and stacked hourglass 3D CNN networks to process the cost volume [3]. PSMNet is expected to help tackle repetitive patterns in images, a major issue we have with real data.

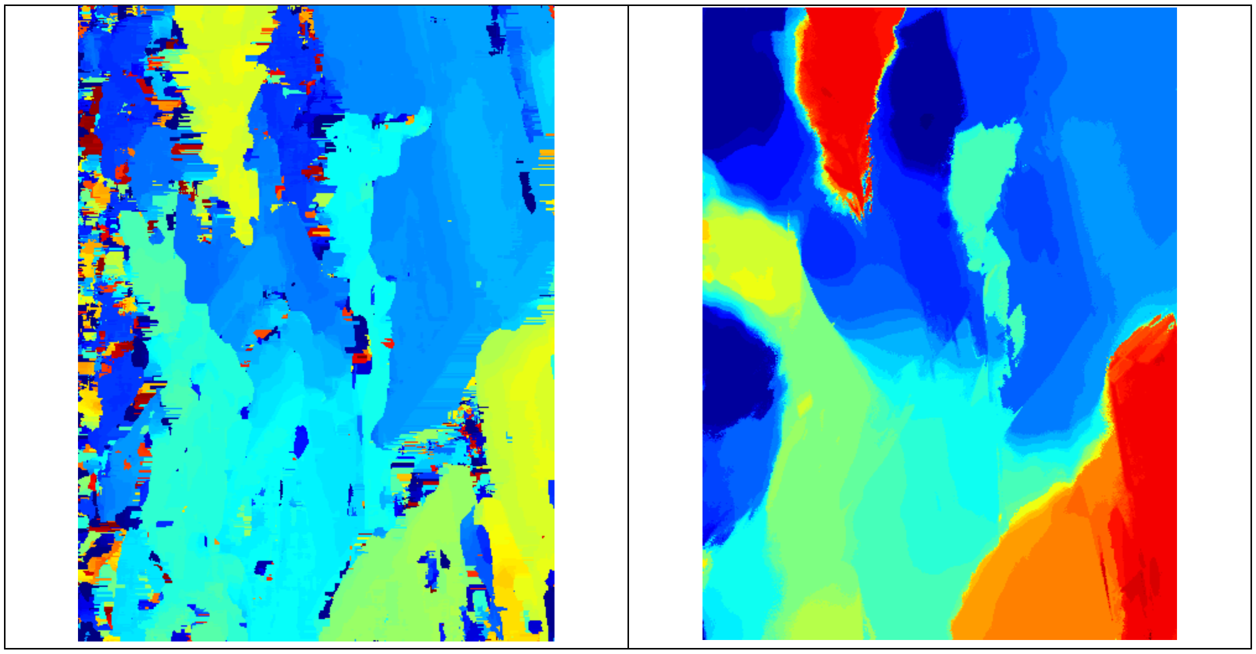
However, from the results below, we can see that pre-trained PSMNet model (trained on rgb data, so we first applied it on synthetic data) did not preserve edges in the original image, perhaps due to issues with our data pre-processing.

PSMNet	**Our Method**
	

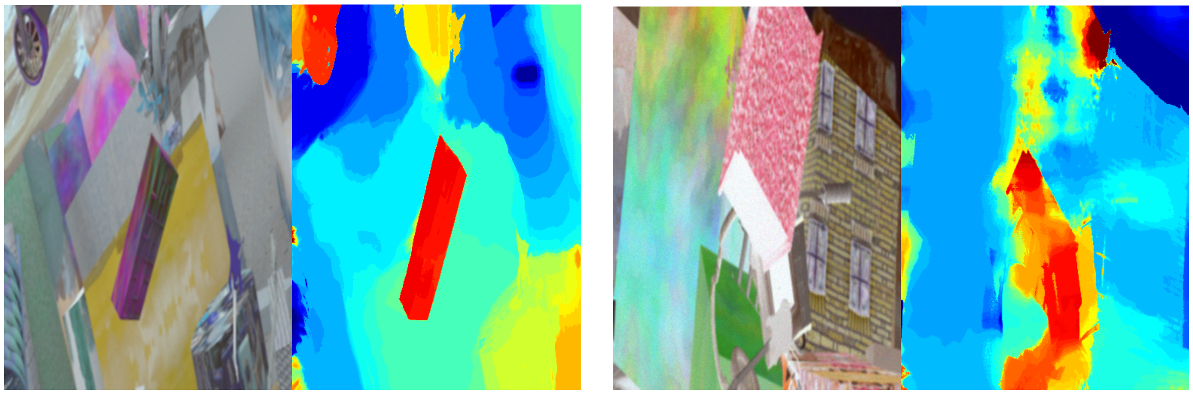
(2) With vs. without weighted median filter

Disparity refinement with weighted median filter helps preserve edges by removing noise, especially in the left region from the result shown below.

Without Weighted Median Filter	**With Weighted Median Filter**



(3) Performance on synthetic data



TL2 / SD2 (error: 2.07) vs. TL4 / SD4 (error: 5.83)

We are curious about the scenarios where our method performs the best and the worst, so we take a closer look at results with the lowest (data: TL2) and highest (data: TL4) error. TL2 has easily distinguishable patterns, so our method shows reasonable result in most regions except with the wheel on the upper left corner. As for TL4, much improvement is needed for obscure images and regions with complex patterns.

6. References

[1] Shaharyar Ahmed Khan Tareen and Zahra Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK." iCoMET 2018

[2] Ziyang Ma, Kaiming He, Yichen Wei, Jian Sun and Enhua Wu, "Constant Time Weighted Median Filtering for Stereo Matching and Beyond" ICCV 2013

[3] Jia-Ren Chang and Yong-Sheng Chen, "Pyramid Stereo Matching Network." CVPR 2018

7. Usage

Run Example

```
python3 main.py --input-left <Lpath> --input-right <Rpath> --output <Outputpath>
```

- `<Lpath>` (e.g. `.data/Synthetic/TL0.png`)
- `<Rpath>` (e.g. `.data/Synthetic/TR0.png`)
- `<Outputpath>` (e.g. `.output/Synthetic/SD0.pfm`)

Visualize

```
python3 run_visualize.py
```

Test

```
python3 run_main.py <Scene>
```

- `<Scene>` (e.g. `Synthetic`)

Evaluation

```
python3 cal_err.py
```

Calculate average error on synthetic scenes.