Computer Vision Final Project

MTK - Depth Map Generation on More Realistic Scenes

B04901190 范晟祐 R05945052 曾子家
R07945026 林炳彰 R06922029 昝亭甄

Methods

- Simple method (derived from hw4, traditional method without deep learning)
    - Cost computation
        - Matching cost
            - Squared difference (SD):
    - Cost aggregation
        - Box filter
            - Aggregate costs within certain window size (10)
    - Disparity Optimization
        - winner-take-all
    - Disparity Refinement
        - Left-right consistency check
        - Hole filling
        - Weighted median filtering

- GCNet
    - architecture: preprocessing + GCNet
        - preprocessing
            - simple CNN for recignize Real ro Synthetic
                - 5 layer convolution
                - assure the test data type
            - synthetic data
                - Histogram matching
            - real data
                - Histogram matching
                - GaussianBlur (kernel = 3×3)
                - Threshold (OTSU)
                - Erode
                - Dilate
        - GCNet
            - Dataset: Sceneflow Flyingthings3D cleanpass
            - Train epoch: 10 (due to computing power)
            - Input Size: 512×384
            - Maxdisp: 64

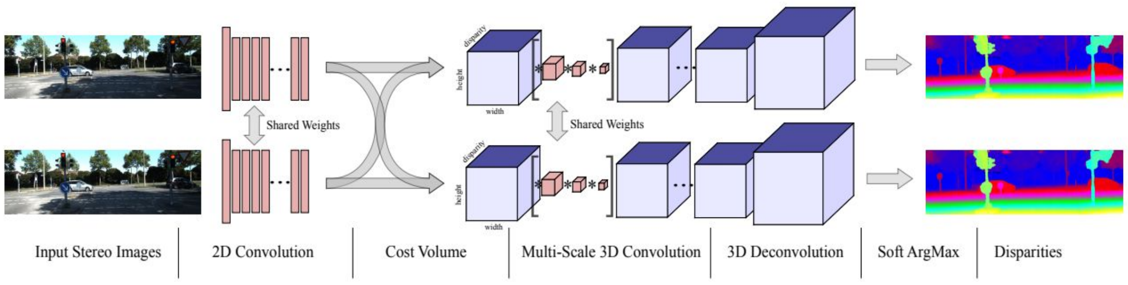
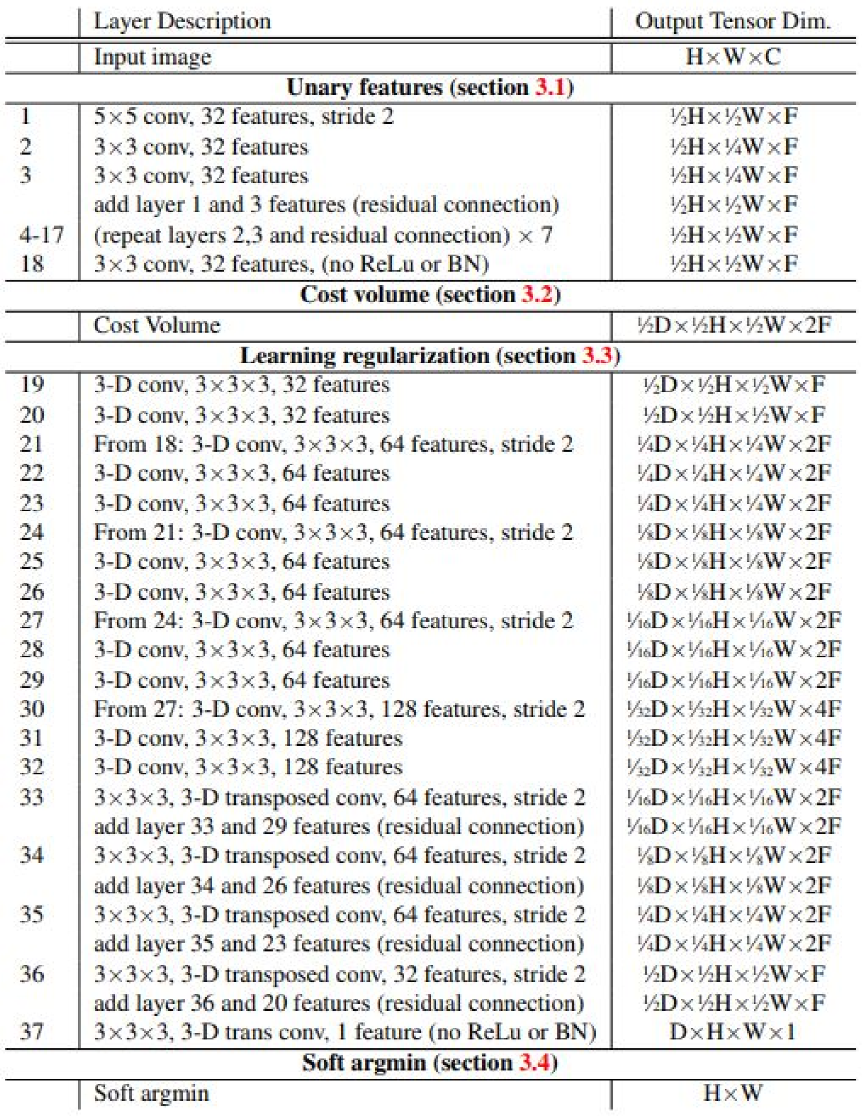


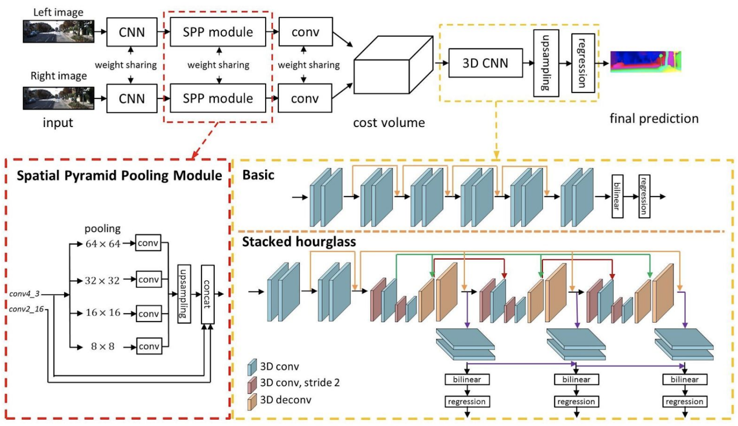
Figure 1: **Our end-to-end deep stereo regression architecture, GC-Net** (Geometry and Context Network).

- network configurations



	Layer Description	Output Tensor Dim.
	Input image	H×W×C
	Unary features (section 3.1)	
1	5×5 conv, 32 features, stride 2	½H×½W×F
2	3×3 conv, 32 features	½H×¼W×F
3	3×3 conv, 32 features	½H×¼W×F
	add layer 1 and 3 features (residual connection)	½H×½W×F
4-17	(repeat layers 2,3 and residual connection) × 7	½H×½W×F
18	3×3 conv, 32 features, (no ReLu or BN)	½H×½W×F
	Cost volume (section 3.2)	
	Cost Volume	½D×½H×½W×2F
	Learning regularization (section 3.3)	
19	3-D conv, 3×3×3, 32 features	½D×½H×½W×F
20	3-D conv, 3×3×3, 32 features	½D×½H×½W×F
21	From 18: 3-D conv, 3×3×3, 64 features, stride 2	¼D×¼H×¼W×2F
22	3-D conv, 3×3×3, 64 features	¼D×¼H×¼W×2F
23	3-D conv, 3×3×3, 64 features	¼D×¼H×¼W×2F
24	From 21: 3-D conv, 3×3×3, 64 features, stride 2	⅛D×⅛H×⅛W×2F
25	3-D conv, 3×3×3, 64 features	⅛D×⅛H×⅛W×2F
26	3-D conv, 3×3×3, 64 features	⅛D×⅛H×⅛W×2F
27	From 24: 3-D conv, 3×3×3, 64 features, stride 2	1/16D×1/16H×1/16W×2F
28	3-D conv, 3×3×3, 64 features	1/16D×1/16H×1/16W×2F
29	3-D conv, 3×3×3, 64 features	1/16D×1/16H×1/16W×2F
30	From 27: 3-D conv, 3×3×3, 128 features, stride 2	1/32D×1/32H×1/32W×4F
31	3-D conv, 3×3×3, 128 features	1/32D×1/32H×1/32W×4F
32	3-D conv, 3×3×3, 128 features	1/32D×1/32H×1/32W×4F
33	3×3×3, 3-D transposed conv, 64 features, stride 2	1/16D×1/16H×1/16W×2F
	add layer 33 and 29 features (residual connection)	1/16D×1/16H×1/16W×2F
34	3×3×3, 3-D transposed conv, 64 features, stride 2	⅛D×⅛H×⅛W×2F
	add layer 34 and 26 features (residual connection)	⅛D×⅛H×⅛W×2F
35	3×3×3, 3-D transposed conv, 64 features, stride 2	¼D×¼H×¼W×2F
	add layer 35 and 23 features (residual connection)	¼D×¼H×¼W×2F
36	3×3×3, 3-D transposed conv, 32 features, stride 2	½D×½H×½W×F
	add layer 36 and 20 features (residual connection)	½D×½H×½W×F
37	3×3×3, 3-D trans conv, 1 feature (no ReLu or BN)	D×H×W×1
	Soft argmin (section 3.4)	
	Soft argmin	H×W

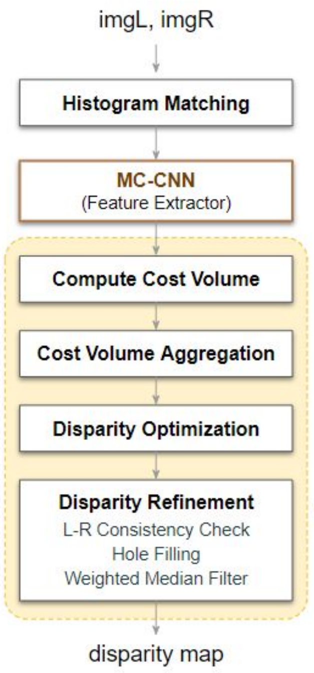
- PSMNet
    - architecture: PSMNet

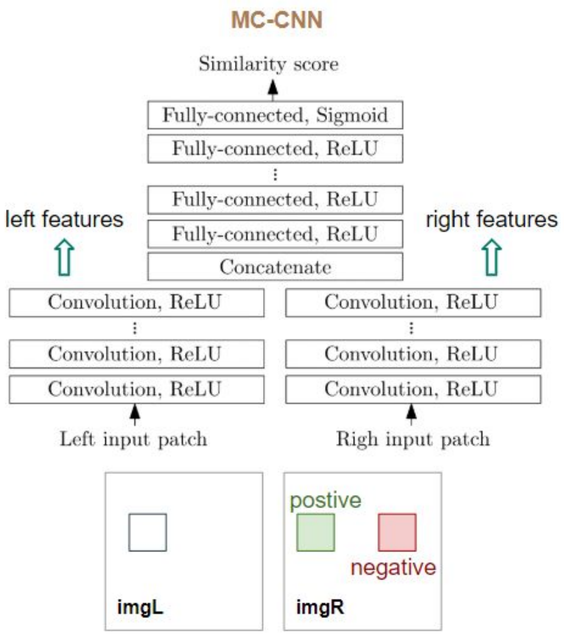


    - Training detail:
        - Use PSMNet pre-trained model (FlyingThings3D dataset)
        - Random rotate the image (from $10^\circ \sim 360^\circ$)
        - Fix the image size to Training (H: 512, W: 384)
        - Because the image in real world may have different brightness, i try to random adjust the brightness and the saturation in the training data.
        - Training detail:
            - Use PSMNet pre-trained model (FlyingThings3D dataset)
            - Fix the image size to Training (H: 512, W: 384)
            - Data Augmentation:
                - Because the image in real world may have different brightness, i try to random adjust brightness and saturation of the training data.
                - Besides, i also use the random rotation to expend the variety of data.
                - In the real scene image, the brightness of the foreground is close to dark, and the background is close to white. So i try to add the negative film to let the model to learn this saturation.
            - Because the learning rate will decay in the Adam Optim, we reset the learning rate of the model every 100 epochs.
    - Hyper Parameters:
        - Maxdisp: 64, Batch Size: 3
    - Discover:
        - I train the model in 1000 epochs, but there are something terrible happened. The model can fit the training data will, but the result on real scene is too bad.
        - the real data have too many repeated pattern, but the training sample not have the this kind of pattern. If we can find some dataset have the characteristic, maybe we can approve the result on real scene.

- Traditional Method with MC-CNN
    - architecture: Traditional Method + MC-CNN

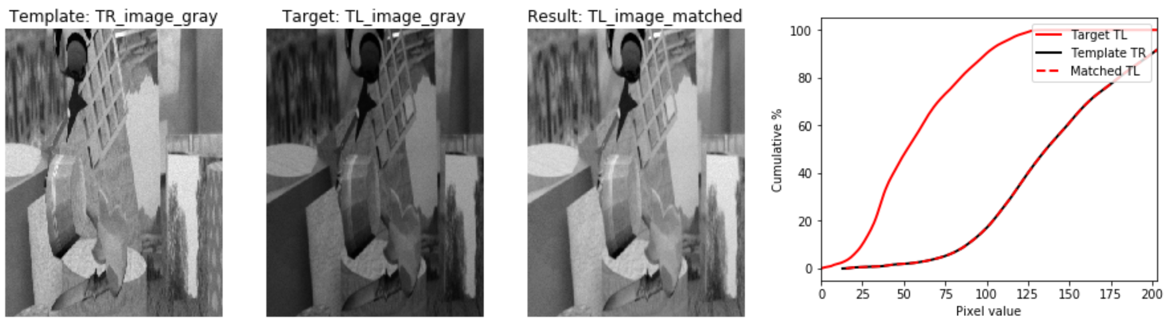
      We use MC-CNN as the feature extractor and compute the cost volume.





    - Preprocessing -- Histogram Matching

      We use right images as the template image for histogram matching because we observe that the left images of the synthetic data are darker or more blurred than the right images.



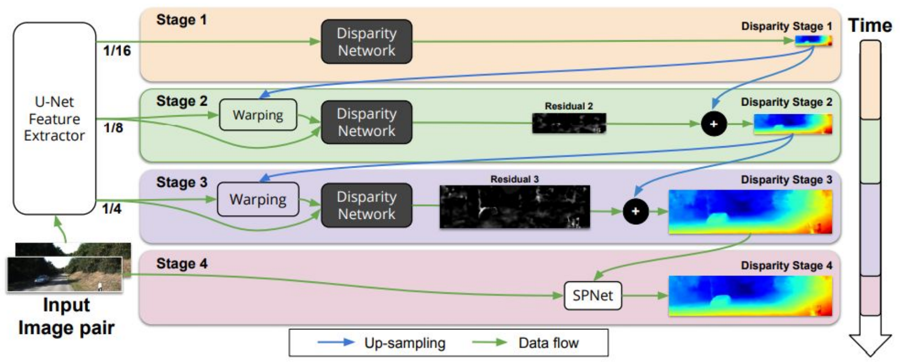
    - Detail for MC-CNN
        - Dataset: MiddEval3, Sceneflow Flyingthings3D cleanpass

          We first use the MiddEval3 dataset to pre-train the model. Then, we found that the characteristic of data in Flyingthings3D cleanpass dataset are similar to the synthetic data, and we think it can improve the performance of the model and still keep generalization (not overfitting to synthetic data).

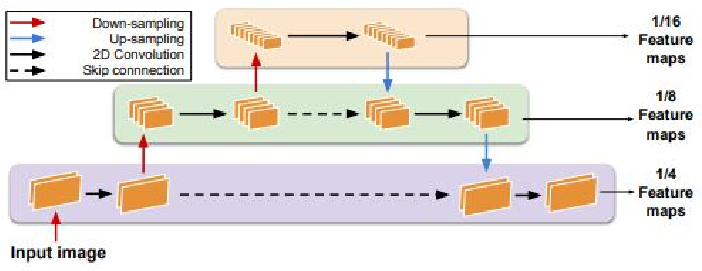
        - Patch Size: 11×11
        - Batch Size: 128
        - Disparity Range: -10 ~ 64

- Cost computation
  - compute the matching cost between left features and right features
- Cost aggregation
  - compute union region in cross-based cost aggregation
  - compute average match cost using union regions
- Disparity Optimization
  - winner-take-all
- Disparity Refinement
  - Left-right consistency check
  - Hole filling
  - Weighted median filtering

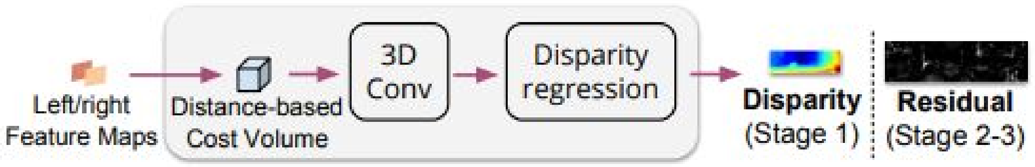
- Anytime Stereo Image Depth Estimation on Mobile Devices
  - application
    - robot vision
  - features
    - generate accurate disparity maps in real time under significant computational constraints (power- or memory-constrained devices.)
    - end-to-end learned approach can trade off computation and accuracy at inference time and output its current best estimate
    - can process 1242×375 resolution images within a range of 10-35 FPS on an NVIDIA Jetson TX2 module with only marginal increases in error – using two orders of magnitude fewer parameters than the most competitive baseline
  - whole architecture



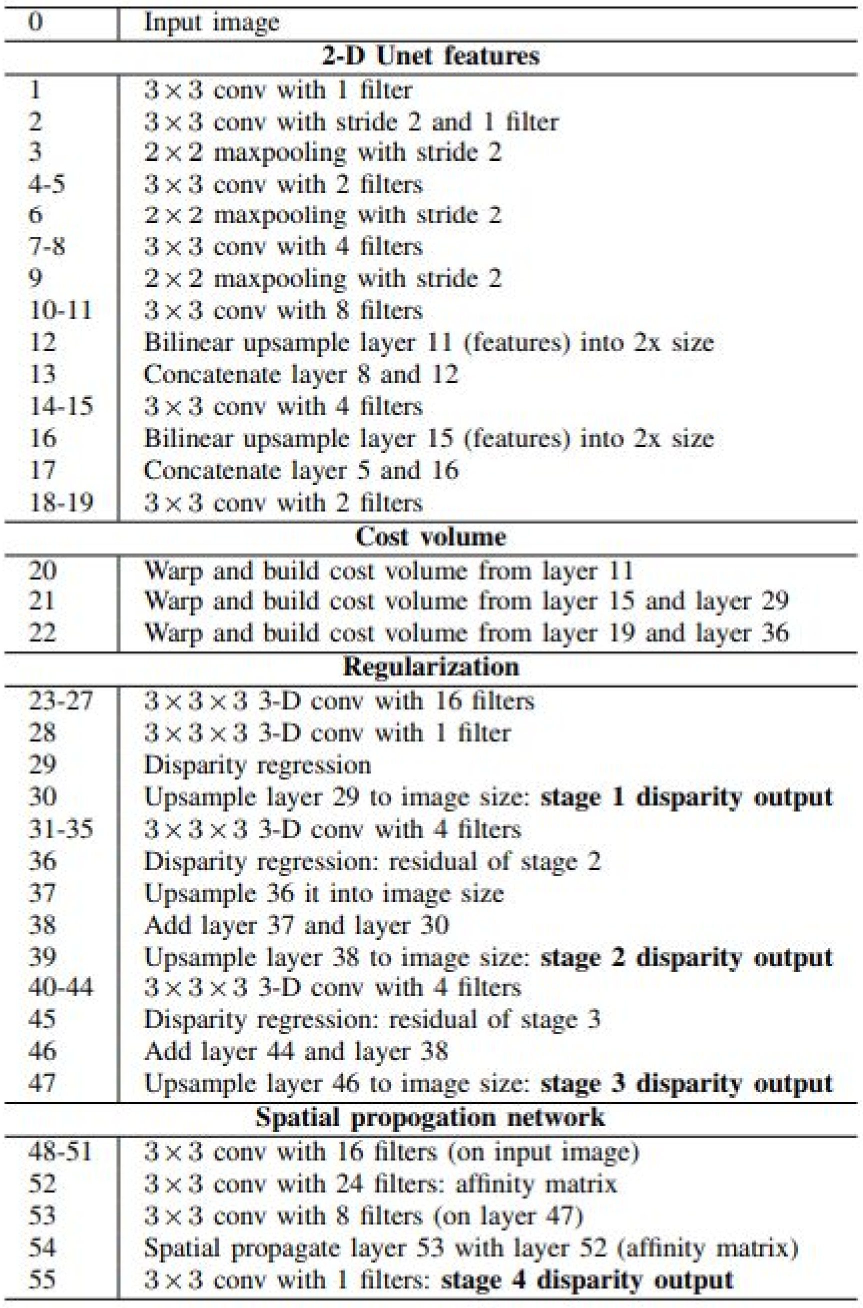
  - U-Net



- disparity network



- network configurations



0	Input image
	2-D Unet features
1	3×3 conv with 1 filter
2	3×3 conv with stride 2 and 1 filter
3	2×2 maxpooling with stride 2
4-5	3×3 conv with 2 filters
6	2×2 maxpooling with stride 2
7-8	3×3 conv with 4 filters
9	2×2 maxpooling with stride 2
10-11	3×3 conv with 8 filters
12	Bilinear upsample layer 11 (features) into 2x size
13	Concatenate layer 8 and 12
14-15	3×3 conv with 4 filters
16	Bilinear upsample layer 15 (features) into 2x size
17	Concatenate layer 5 and 16
18-19	3×3 conv with 2 filters
	Cost volume
20	Warp and build cost volume from layer 11
21	Warp and build cost volume from layer 15 and layer 29
22	Warp and build cost volume from layer 19 and layer 36
	Regularization
23-27	3×3×3 3-D conv with 16 filters
28	3×3×3 3-D conv with 1 filter
29	Disparity regression
30	Upsample layer 29 to image size: **stage 1 disparity output**
31-35	3×3×3 3-D conv with 4 filters
36	Disparity regression: residual of stage 2
37	Upsample 36 it into image size
38	Add layer 37 and layer 30
39	Upsample layer 38 to image size: **stage 2 disparity output**
40-44	3×3×3 3-D conv with 4 filters
45	Disparity regression: residual of stage 3
46	Add layer 44 and layer 38
47	Upsample layer 46 to image size: **stage 3 disparity output**
	Spatial propogation network
48-51	3×3 conv with 16 filters (on input image)
52	3×3 conv with 24 filters: affinity matrix
53	3×3 conv with 8 filters (on layer 47)
54	Spatial propagate layer 53 with layer 52 (affinity matrix)
55	3×3 conv with 1 filters: **stage 4 disparity output**

- this network wasn’t built successfully :(

Experiment Results

- Machine Specs

  CPU: Intel® Core™ i7-8700

  GPU: GTX 1080ti

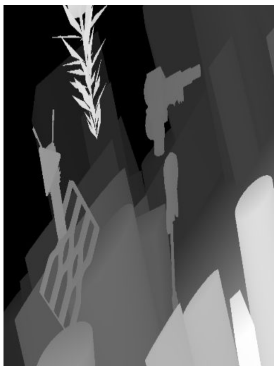
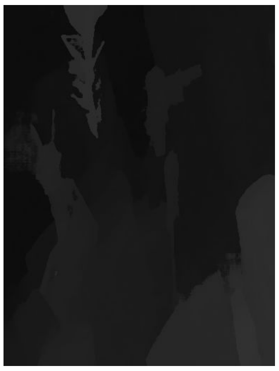
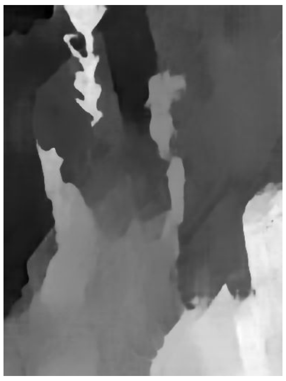
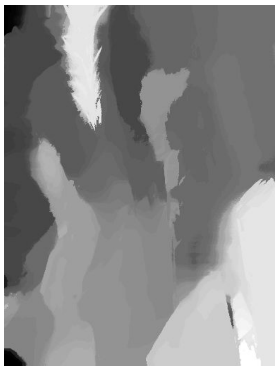
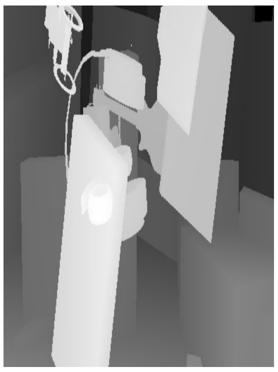
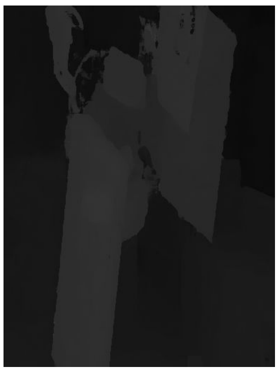
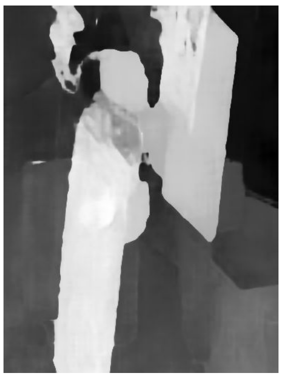
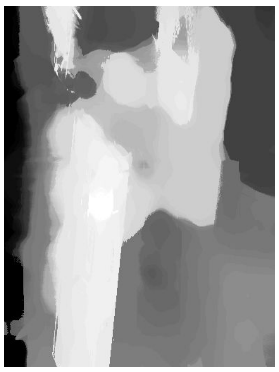
- Average Computation Time

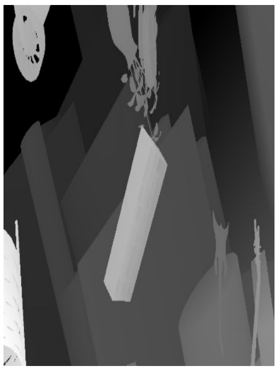
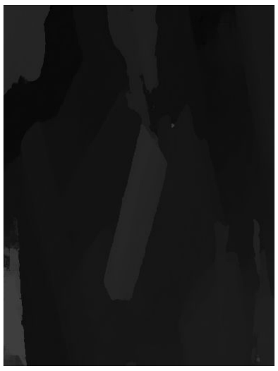
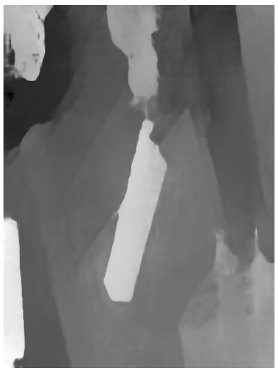
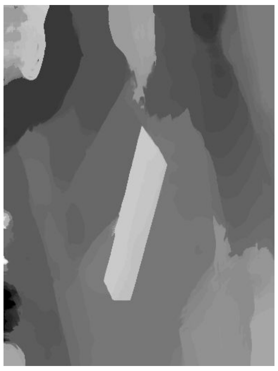
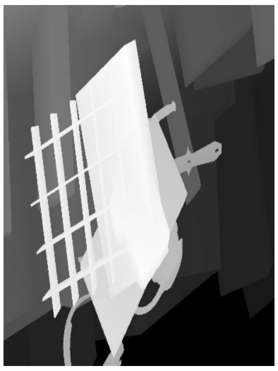
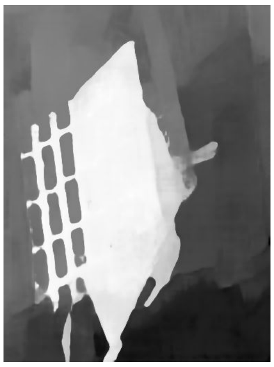
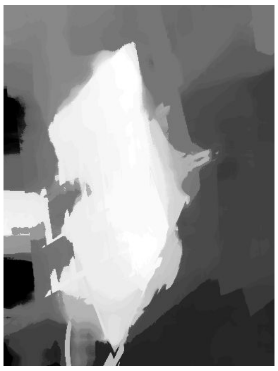
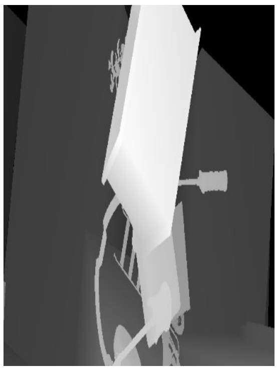
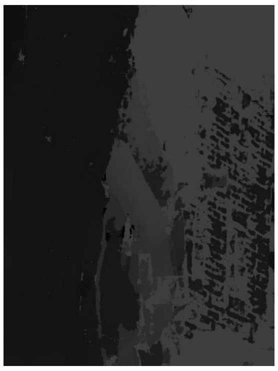
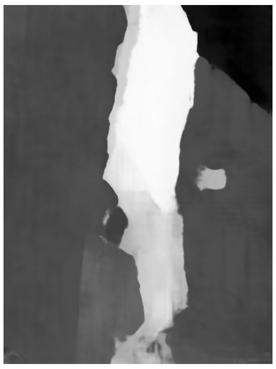
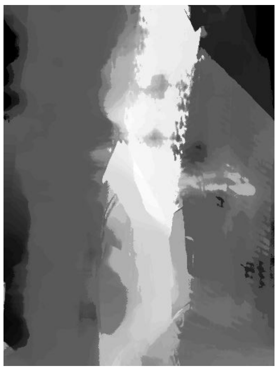
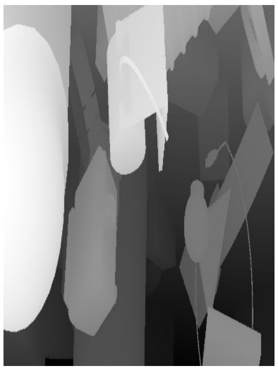
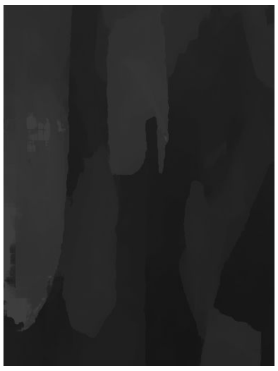
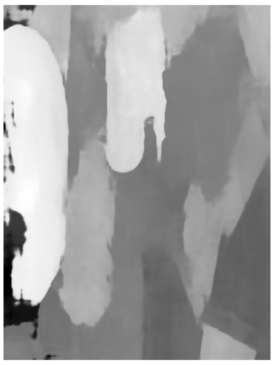
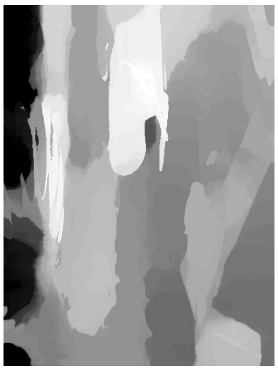
	GCNet	PSMNet	Traditional Method with MC-CNN
Computation Time	4.01s	1.2s	36.79s

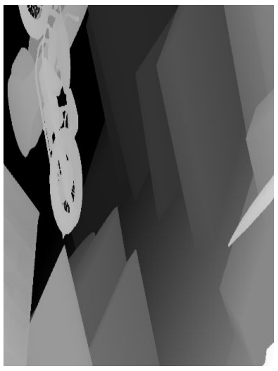
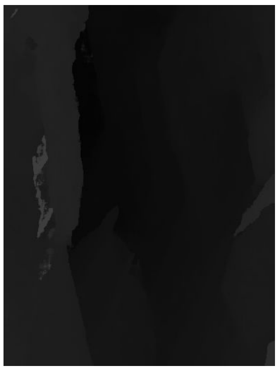
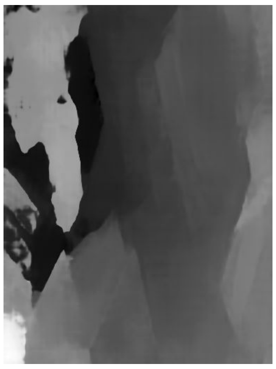
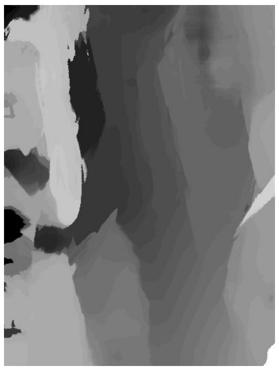
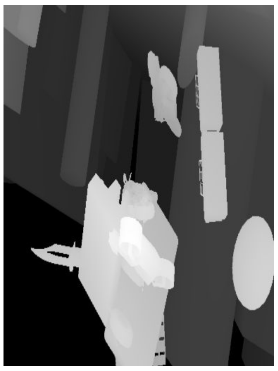
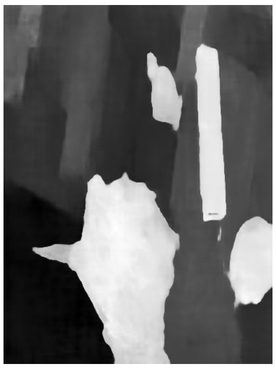
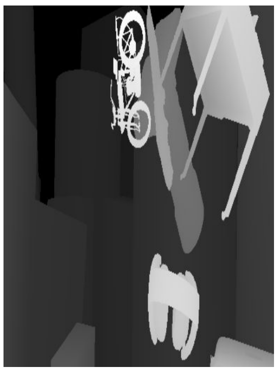
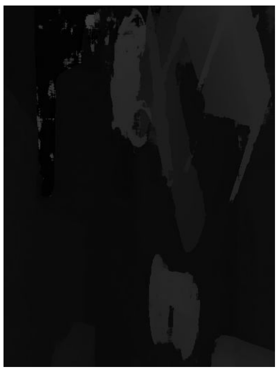
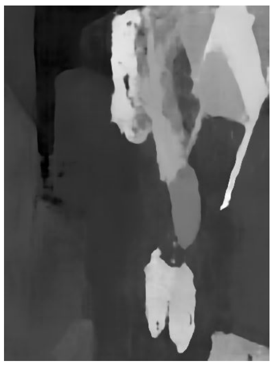
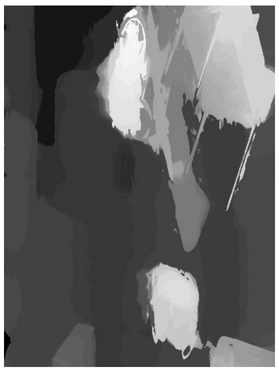
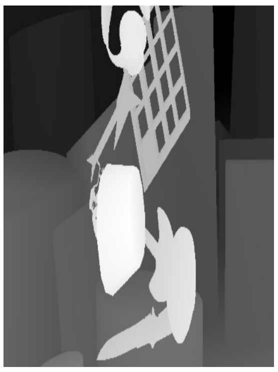
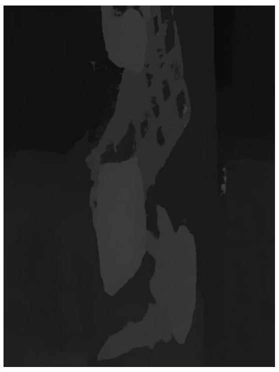
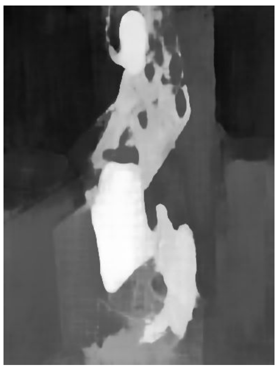
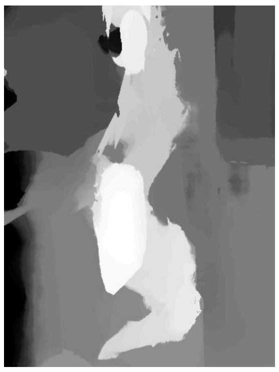
- Average Error for Synthetic Data

	GCNet	PSMNet	Traditional Method with MC-CNN
Average Error	5.13601	2.8076	3.6277

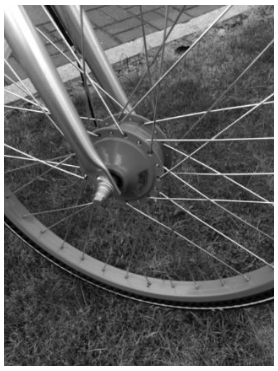
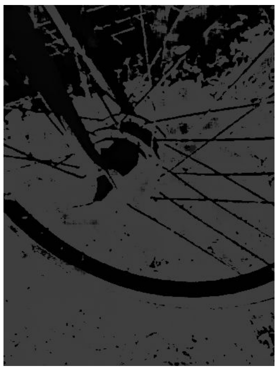
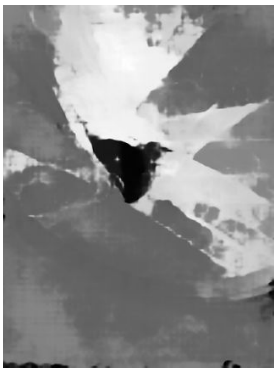
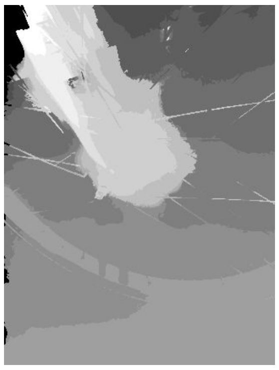
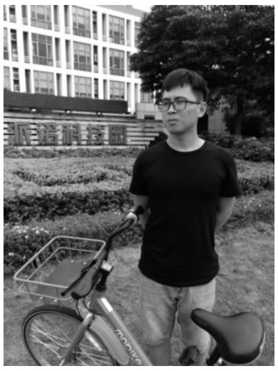
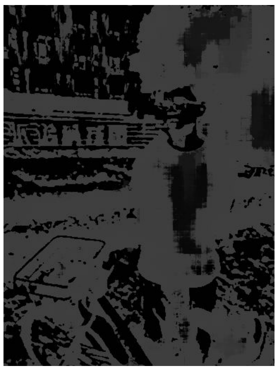
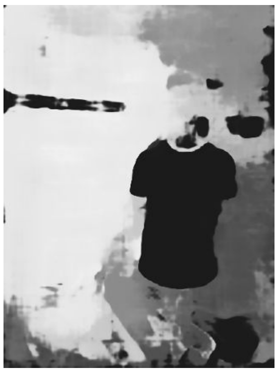
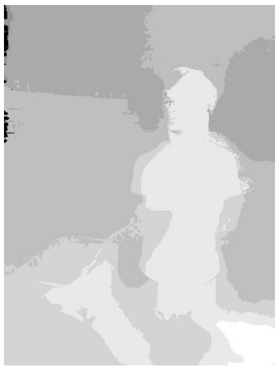
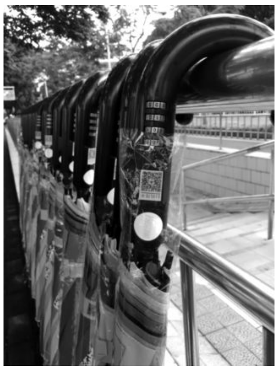
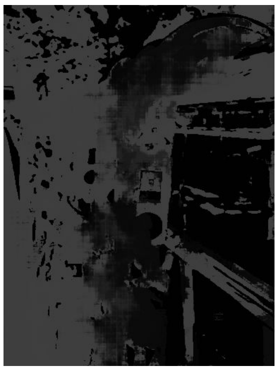
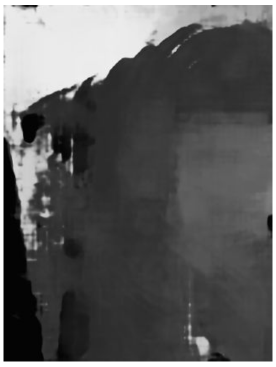
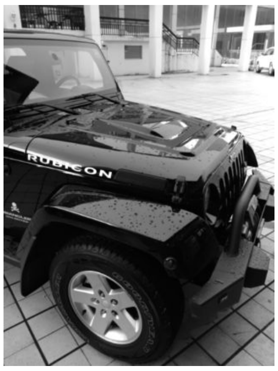
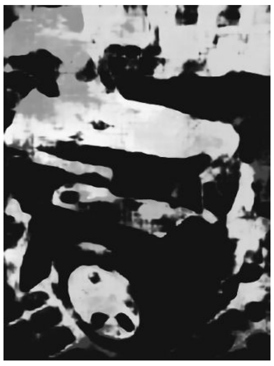
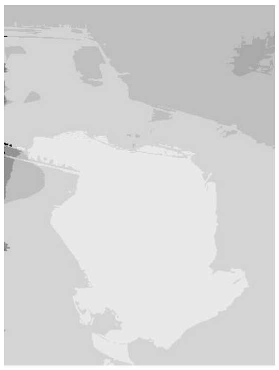
- Synthetic scene data

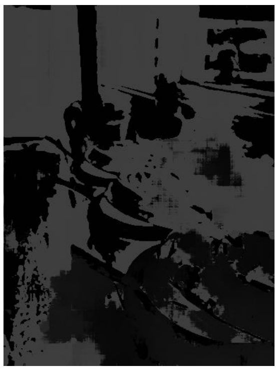
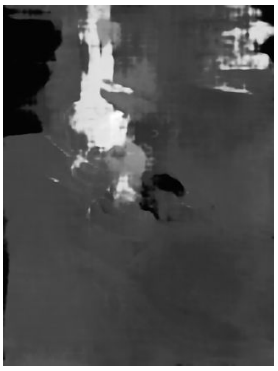
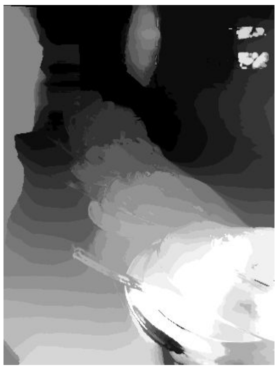
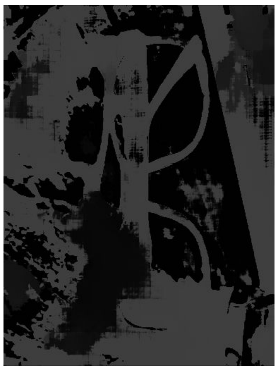
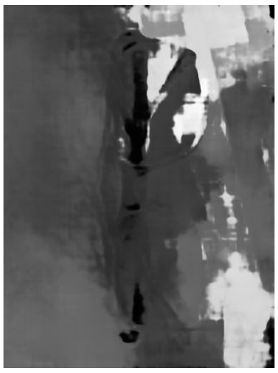
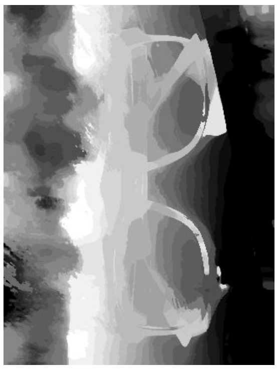
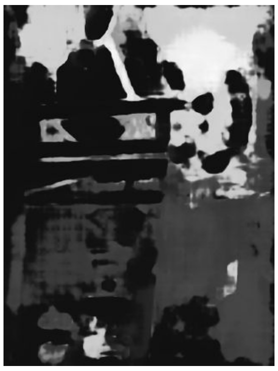
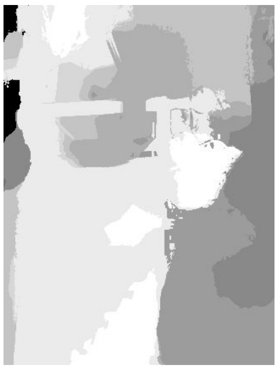
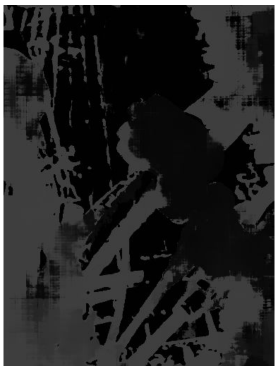
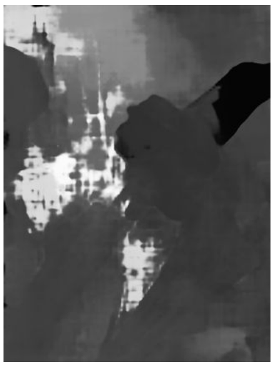
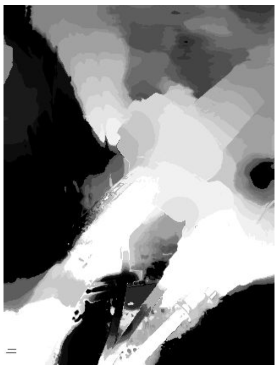
Ground Truth	GCNet	PSMNet (1000 epochs)	Traditional Method with MC-CNN
 TLD0.pfm	 error: 1.67	 error: 2.3561	 error: 1.76
			

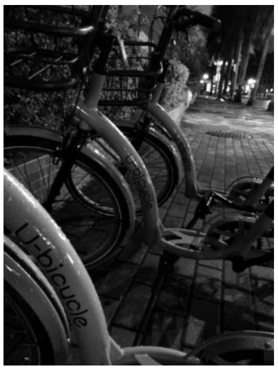
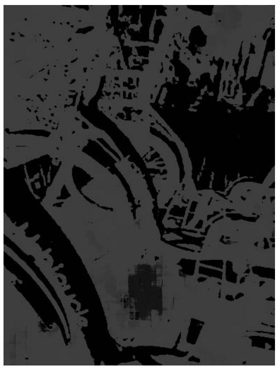
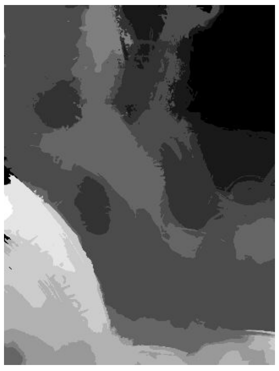
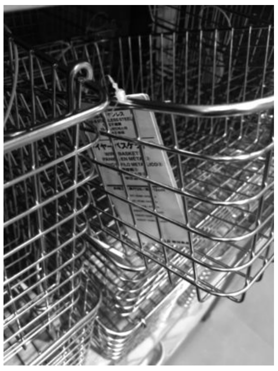
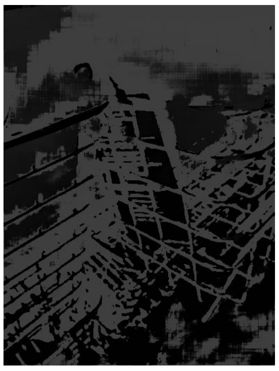
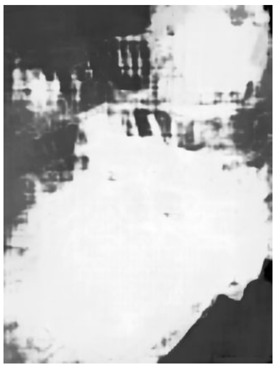
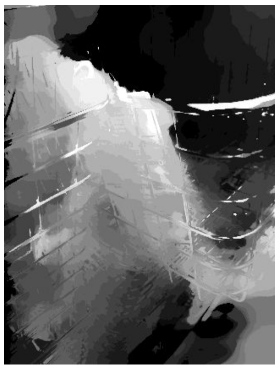
TLD1.pfm	error: 1.84	error: 2.3204	error: 3.01
 TLD2.pfm	 error: 2.15	 error: 1.9695	 error: 2.62
 TLD3.pfm	 error: 4.25	 error: 3.0592	 error: 4.14
 TLD4.pfm	 error: 20.7	 error: 2.2573	 error: 3.49
 TLD5.pfm	 error: 1.87	 error: 3.905	 error: 9.65

 TLD6.pfm	 error: 4.08	 error:2.9487	 error:1.89
 TLD7.pfm	error: 9.46	 error: 2.8102	error: 3.44
 TLD8.pfm	 error: 3.15	 error: 2.6949	 error: 2.34
 TLD9.pfm	 error: 2.18	 error: 3.7512	 error: 3.92

○ Real Scene Data

Left Image	GCNet	PSMNet	Traditional Method with MC-CNN
 TL0.bmp			
 TL1.bmp			
 TL2.bmp			
 TL3.bmp			

TL4.bmp			
TL5.bmp			
TL6.bmp			
TL7.bmp			

 TL8.bmp			
 TL9.bmp			

Implement

GCNet: https://drive.google.com/drive/folders/1UXw23qkgOfLqSJdz59lG-x_VCiKe9DWK?usp=sharing

PSMNet: https://drive.google.com/file/d/1GxHX6COpFvD-BjwKJbDDax7Odi9lfDFW/view?usp=sharing

MC-CNN: https://drive.google.com/open?id=14YHytQk4J8UpRQbEHP3QVt5n3jvq2ker

(We uploaded the "Traditional Method with MC-CNN" code on CEIBA without the model files. The model files are in the google drive.)

References

[1] A. Kendall et al. "End-to-End Learning of Geometry and Context for Deep Stereo Regression," CVPR 2017, https://arxiv.org/abs/1703.04309.

[2] J. Žbontar and Y. LeCun, "Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches," CVPR 2016, https://arxiv.org/abs/1510.05970.

[3] J. R. Chang and Y. S. Chen, "Pyramid Stereo Matching Network," CVPR 2018, https://arxiv.org/abs/1803.08669.

[4] Wang et al. "Anytime Stereo Image Depth Estimation on Mobile Devices," arXiv preprint arXiv:1810.11408, 2018, https://arxiv.org/pdf/1810.11408.pdf.

[5] A python implementation of MC-CNN - Github, https://github.com/Jackie-Chou/MC-CNN-python.

[6] Middlebury Stereo Evaluation - Version 3 (MiddEval3), http://vision.middlebury.edu/stereo/submit3/.

[7] Sceneflow Flyingthings3D cleanpass, https://lmb.informatik.uni-freiburg.de/resources/datasets/.