

Computer Vision Final Project Report

電機三 阮明皓 B05901062 | 電機三 潘彥銘 B05901182

A. Algorithm

We introduced two algorithms because we tried both. But binary stereo matching (BSM) performs better. MC-CNN needs much more time, which we were running out of, to fine-tune.

We only run the BSM code when executing. Both methods are bundled in the file that we uploaded though.

a. Binary stereo matching

i. Cost Computation

We implemented a local method introduced in [1]. First, we introduce BRIEF descriptor.

$$B(x) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(p_i, q_i) \quad (1)$$

Each pair (p_i, q_i) is sampled by an isotropic Gaussian distribution in an $S \times S$ window, which is centered on pixel x . And $\tau(p_i, q_i)$ is a binary function which is defined as:

$$\tau(p_i, q_i) = \begin{cases} 1 & : I(p_i) > I(q_i) \\ 0 & : I(p_i) \leq I(q_i) \end{cases} \quad (2)$$

where $I(x)$ denotes the intensity of pixel x . After calculating the descriptor, i.e. a binary string for each pixel, the cost volume is constructed as:

$$C(x, d) = \| B(x) \mathbf{XOR} B(x_d) \|_1 \quad (3)$$

where x_d is the corresponding pixel of x with disparity d in another view, XOR is a bitwise xor-operation. In short, $C(x, d)$ measures the hamming distance between two binary strings.

ii. Cost Aggregation

Firstly, we define a weight function for pixel pair (p_i, q_i) in (1) as:

$$w(x, p_i, q_i) = \max(SAD(x, p_i), SAD(x, q_i)) \quad (4)$$

where $SAD(x, y) = \sum_{c \in [L, A, B]} |I_c(x) - I_c(y)|$ is the sum of absolute difference between two pixels in the CIELAB color space. Then we get our bitwise mask function for a given pair (p_i, q_i) as:

$$\delta(x, p_i, q_i) = \begin{cases} 1 & : w(x, p_i, q_i) \leq T \\ 0 & : w(x, p_i, q_i) > T \end{cases} \quad (5)$$

where T is set to be the quarter smallest value in the sequence $w(x, p_1, q_1)$, $w(x, p_2, q_2)$, ..., $w(x, p_n, q_n)$. Finally, we can use this mask function to compose a binary mask:

$$\Phi(x) = \sum_{1 \leq i \leq n} 2^{i-1} \delta(x, p_i, q_i) \quad (6)$$

Consequently $\Phi(x)$ is the proposed binary mask for cost aggregation. Incorporating the binary mask into (3), the new cost volume is defined as:

$$C(x, d) = \|B(x) \text{ XOR } B(x_d) \text{ AND } \Phi(x)\|_1 \quad (7)$$

According to the definition of the binary mask, it will preserve those pixel pairs who have similar depth with window's center.

iii. Disparity Optimization

We implemented the Winner-Take-All method mentioned in class.

iv. Disparity Refinement

Firstly, a left/right check using two-view depth maps is performed to classify depth results into two categories: valid and invalid. For an invalidated pixel p , we search its closest valid pixel to the left and to the right. We select the lower of the two as p 's refined disparity [2].

Then, we apply 5x5 median and bilateral filter to get our final disparity map.

We have also tried different refinement method, like subpixel enhancement and eight direction interpolation. But they don't work well on the binary stereo matching. The performance is not stable.

v. More Optimizations for Fine-tuning

We found that real data given may have negative disparities. So we pad the left side of the input left image and the right side of the right image. We will still get two equal-sized input images, but with right image shifted left. There will be no negative disparities afterwards.

However, the synthetic data, which have no negative disparities originally, now gets a wider range of choices of possible disparities. This leads to higher

error rate. To handle this problem, we set a threshold on the amount of points whose disparities are smaller than the padding size. If the amount is above the threshold, we take the image as the one with more negative disparities. Otherwise, we take the image as the one with no negative disparities and see the negative disparities as invalid in the refinement step.

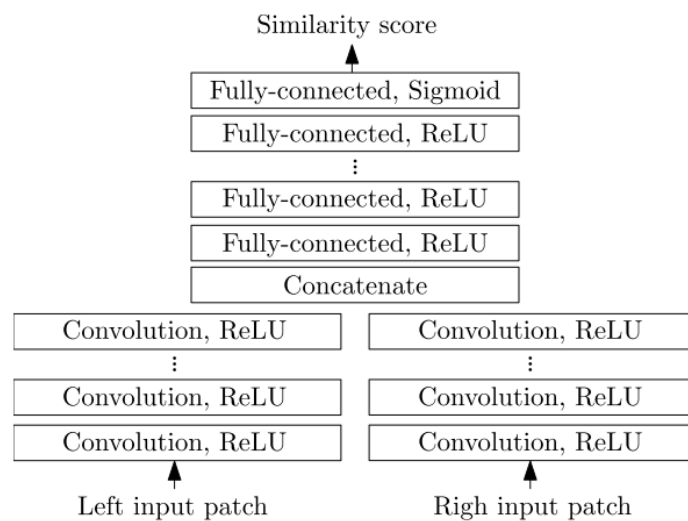
After a few other tuning, we get a much better result on the real data, and remain good on the synthetic data.

B. MC-CNN

a. Cost Computation

We build a binary classification dataset first, which consists of lots of pair of patches from images. These pair are divided into two categories: positive and negative example. Positive example means the match of the pair is correct and negative example means the mismatch of the pair.

Our model architecture is like below:



The input of this network is two image patches and the output is a similarity score, which is between 0 and 1. We hope the network will output high/low similarity score when input positive/negative examples after our training.

b. Cost Aggregation

We use the cross-based aggregation to build the support region in both left and right images, whose intersection will be the final support region.

Compute the matching cost in the final support region, like (8):

$$\begin{aligned}
C_{\text{CBCA}}^0(\mathbf{p}, d) &= C_{\text{CNN}}(\mathbf{p}, d), \\
C_{\text{CBCA}}^i(\mathbf{p}, d) &= \frac{1}{|U_d(\mathbf{p})|} \sum_{\mathbf{q} \in U_d(\mathbf{p})} C_{\text{CBCA}}^{i-1}(\mathbf{q}, d),
\end{aligned} \tag{8}$$

where $U_d(\mathbf{p})$ is the final support region, $|U_d(\mathbf{p})|$ is its size and i is the iteration time.

Then we perform the semi-global matching. The goal is to minimize the energy function $E(D)$:

$$\begin{aligned}
E(D) = \sum_{\mathbf{p}} \left(C_{\text{CBCA}}^4(\mathbf{p}, D(\mathbf{p})) + \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} P_1 \cdot 1\{|D(\mathbf{p}) - D(\mathbf{q})| = 1\} \right. \\
\left. + \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} P_2 \cdot 1\{|D(\mathbf{p}) - D(\mathbf{q})| > 1\} \right),
\end{aligned} \tag{9}$$

where P_1 and P_2 are penalization coefficients and D is disparity image. But to minimize $E(D)$ is a NP-complete problem. So we reduce $E(D)$ to four direction minimization problem, like (10):

$$\begin{aligned}
C_r(\mathbf{p}, d) = C_{\text{CBCA}}^4(\mathbf{p}, d) - \min_k C_r(\mathbf{p} - \mathbf{r}, k) + \min \left\{ C_r(\mathbf{p} - \mathbf{r}, d), C_r(\mathbf{p} - \mathbf{r}, d - 1) + P_1, \right. \\
\left. C_r(\mathbf{p} - \mathbf{r}, d + 1) + P_1, \min_k C_r(\mathbf{p} - \mathbf{r}, k) + P_2 \right\}.
\end{aligned} \tag{10}$$

where \mathbf{r} denotes the direction, we solve this problem by using dynamic programming. After computing the $C_r(\mathbf{p}, d)$ in four direction, the matching cost will be:

$$C_{\text{SGM}}(\mathbf{p}, d) = \frac{1}{4} \sum_{\mathbf{r}} C_r(\mathbf{p}, d). \tag{11}$$

After the semi global matching, we perform the cross-base aggregation again to get our final matching cost.

c. Disparity optimization

We implemented the Winner-Take-All method.

d. Disparity refinement

We first apply the following rules to divide the pixels into three categories:

correct if $|d - D^R(\mathbf{p} - \mathbf{d})| \leq 1$ for $d = D^L(\mathbf{p})$,
mismatch if $|d - D^R(\mathbf{p} - \mathbf{d})| \leq 1$ for any other d ,
occlusion otherwise.

where $D^L(\mathbf{p})$ and $D^R(\mathbf{p})$ is the left and right disparity map.

For occlusion pixels, we search its left side to find the first correct pixel and choose its disparity for the mismatch pixel.

For mismatch pixels, we search its eight directions to find the first correct pixel in each direction. We compute the median of these eight pixels' disparity and the median will be the disparity of mismatch pixel. The disparity map after this step is denoted $D_{INT}(p)$.

Then we perform the subpixel enhancement like (12):



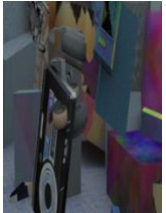
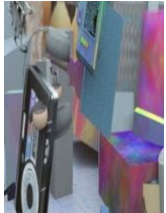


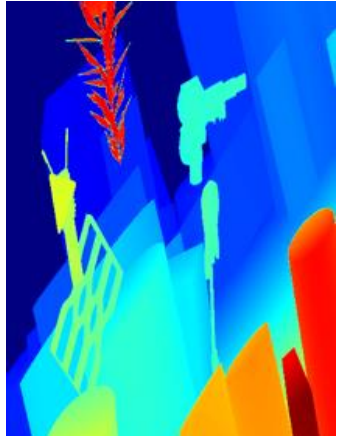
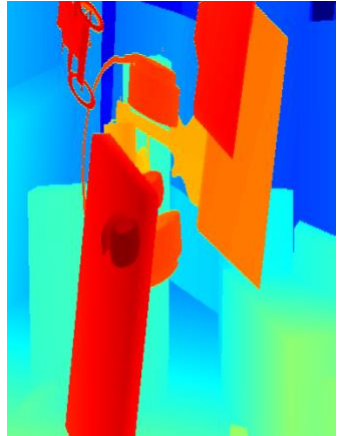
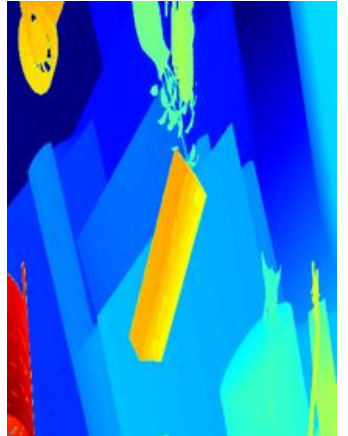
$$D_{SE}(p) = d - \frac{C_+ - C_-}{2(C_+ - 2C + C_-)}, \quad (12)$$

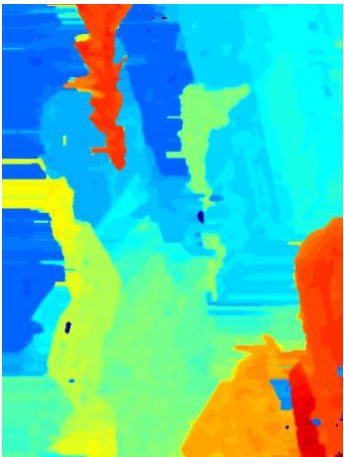
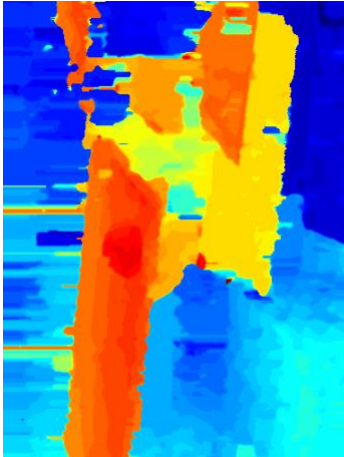
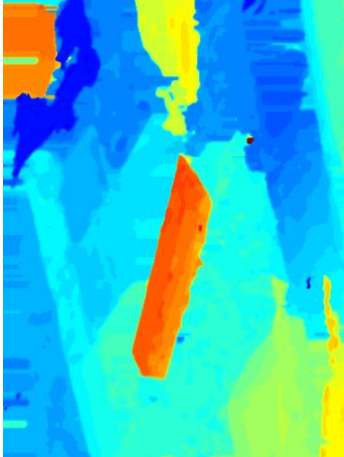


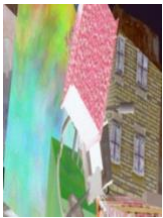




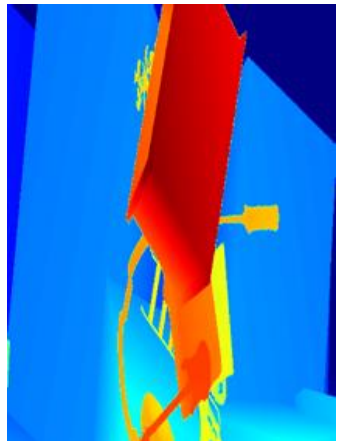
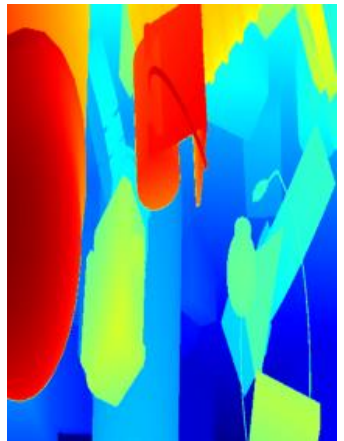
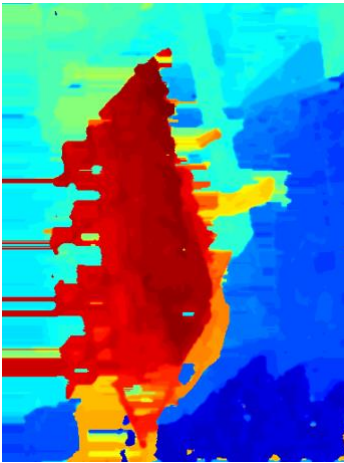
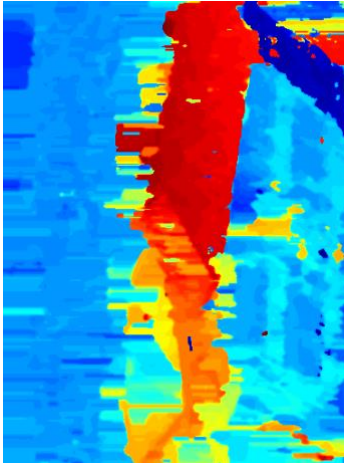
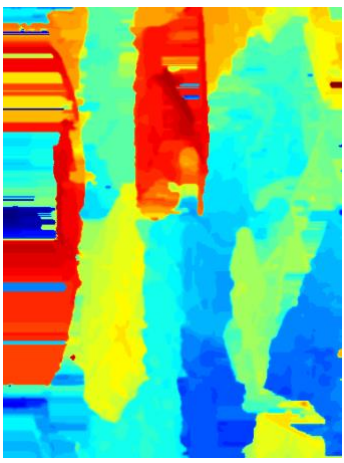
where $d = D_{INT}(p)$, $C_- = C_{SGM}(p, d - 1)$, $C = C_{SGM}(p, d)$, and $C_+ = C_{SGM}(p, d + 1)$.


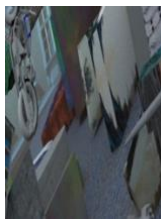
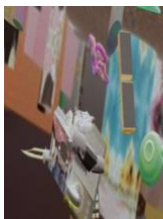



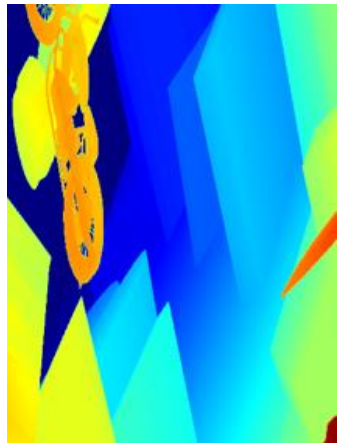
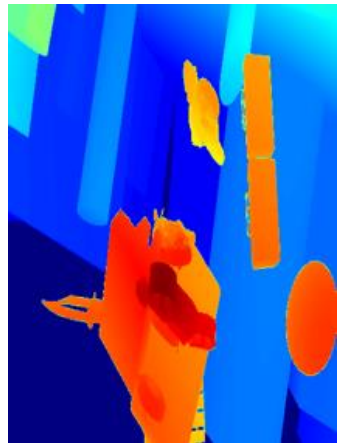
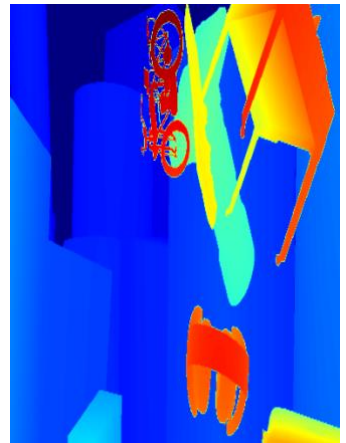
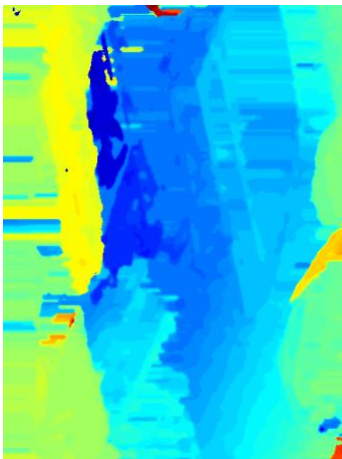
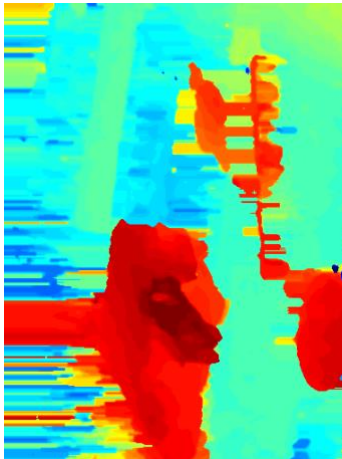
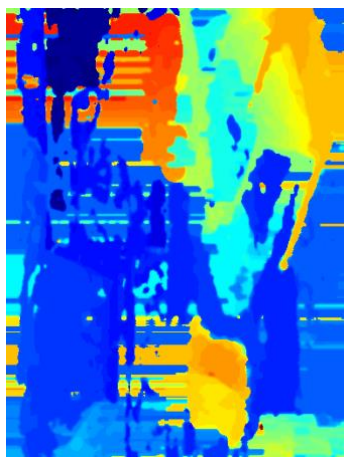


Finally we apply 5x5 median and bilateral filter to $D_{SE}(p)$ to get our final disparity map.

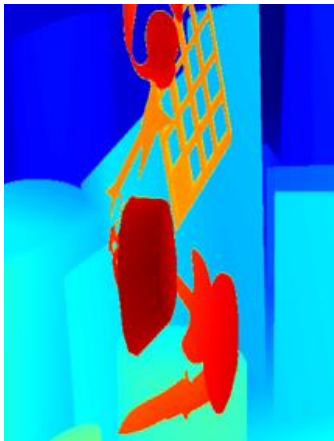
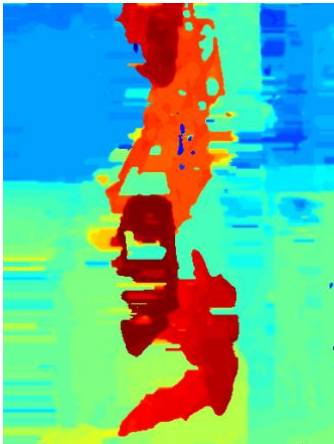
C. Results

a. Synthetic Images







	0		1		2	
L/R						
GT						

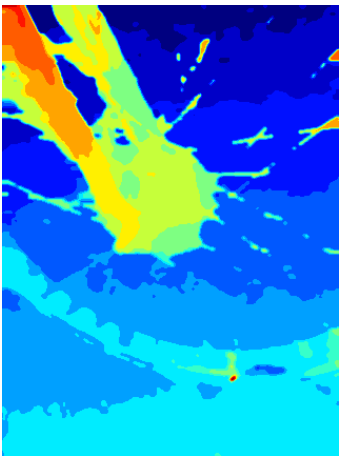

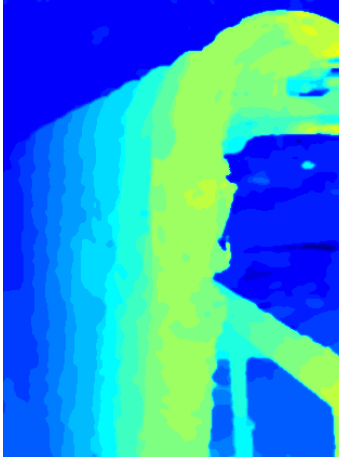






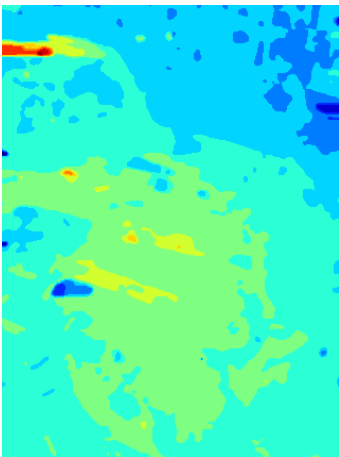
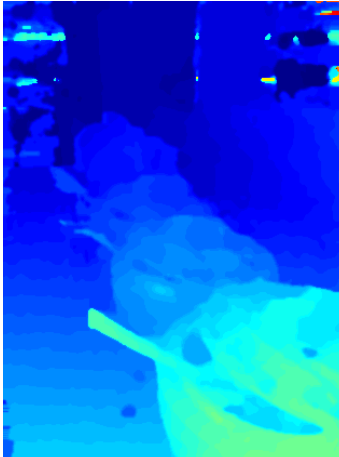
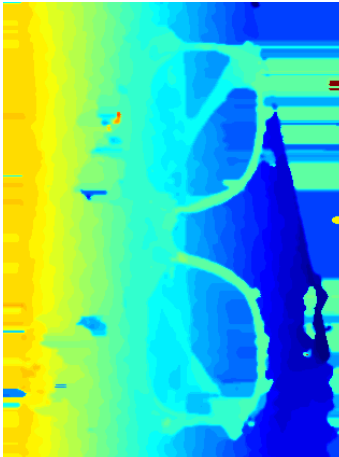






Result						
Error	1.70		1.99		2.61	
	3		4		5	
L/R	 		 		 	
GT						
Result						
Error	3.94		3.34		4.10	

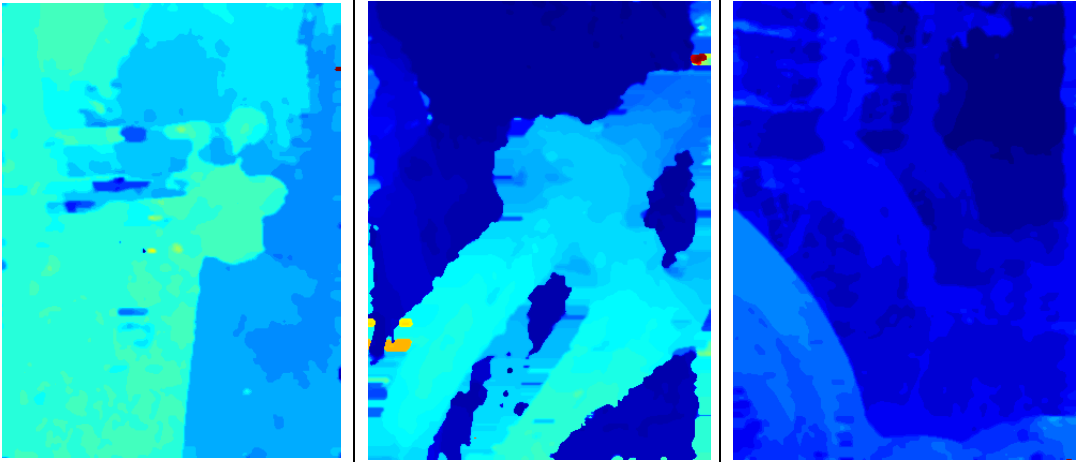

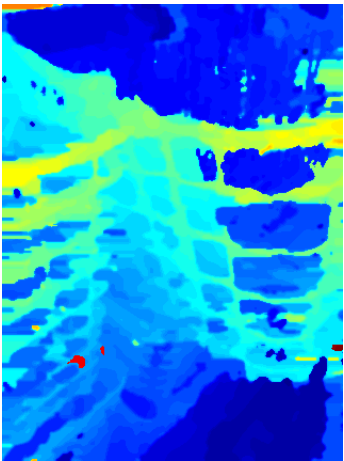
	6		7		8	
L/R						
GT						
Result						
Error	1.65		3.47		3.56	
	9		<p>Average Error: 2.93 Runtime of Image 0: 243 sec (Machine Spec: Intel Core i7 4-core CPU)</p>			
L/R						

GT		
Result		
Error	2.96	

b. Real Images

	0		1		2	
L/R						

Result						
	3		4		5	
L/R						
Result						
	6		7		8	
L/R						

Result			
	9		
L/R			
Result		<p>Runtime of Image 0: 165 sec (Machine Spec: Intel Core i7 4-core CPU)</p>	

C. References

- [1] Kang Zhang, Jiyang Li, Yijing Li, WeiDong Hu, Lifeng Sun, and Shiqiang Yang. Binary stereo matching. CoRR, abs/1402.2020, 2014.
- [2] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo - stereo matching with slanted support windows. In BMVC, January 2011.
- [3] J. Zbontar~ and Y. LeCun, “Stereo matching by training a convolutional neural network to compare image patches,” The Journal of Machine Learning Research, vol. 17, no. 1, pp. 2287–2318, 2016.