Multi-Scale Cost Volumes Cascade Network for Stereo Matching

Xiaogang Jia, Wei Chen*, Zhengfa Liang, Mingfei Wu, Xuehui Wang, Chen Li

Abstract—Stereo matching is essential for robot navigation. However, the accuracy of current widely used traditional methods is low, while methods based on CNN need expensive computational cost and running time. This is because different cost volumes play a crucial role in balancing speed and accuracy. Thus we propose MSCVNet, which combines traditional methods and CNN to improve the quality of cost volume. Concretely, our network first generates multiple 3D cost volumes with different resolutions and then uses 2D convolutions to construct a novel cascade hourglass network for cost aggregation. Meanwhile, we design an algorithm to distinguish and calculate the loss for discontinuous areas of disparity result. According to the KITTI official website, our network is much faster than most top-performing methods (24×than CSPN, 44×than GANet, etc.). Meanwhile, compared to traditional methods (SPS-St, SGM) and other real-time stereo matching networks (Fast DS-CS, DispNetC, and RTSNet, etc.), our network achieves a big improvement in accuracy, demonstrating the effectiveness of our proposed method.

I. INTRODUCTION

Stereo matching can recover the depth information of the scene from planar pictures. Since lidar is expensive, many robot applications rely on it to obtain depth for odometry and navigation. For example, the traditional SGM algorithm [1] is well known and widely used in real-time robot applications. However, the SGM algorithm is sensitive to uneven illumination and textureless areas, which leads to low accuracy. Therefore, most recent methods use CNN to construct stereo matching networks.

According to the dimension of cost volumes and aggregation networks, stereo matching networks can be divided into the following two types: 2D convolution networks based on 3D cost volume [2]–[4] and 3D convolution networks based on 4D cost volume [5]–[7]. Recent top-performing methods mainly adopt the latter, such as GCNet [5], PSMNet [6], CSPN [7], etc. However, 3D convolution networks require a lot of memory and computational cost, so these methods often require many GPUs for training and long-running time for testing. Besides, complex network structures also lead to low generalization, far from replacing the SGM algorithm [1] and applying it to real robot applications.

In contrast, 2D convolution networks [3], [4] have great advantages in memory and running time, but they lose a lot of feature information when generating 3D cost volume (all channel values of two pixels are converted into one value).

This is very similar to traditional methods [8] (the RGB channel values of two pixels are converted into one value). Therefore, both traditional methods and 2D convolution networks have low accuracy.

Considering that a single 3D cost volume will lose a lot of feature information, and the 4D cost volume is far from being applied to real-time robot applications in terms of running time, we integrate traditional methods and CNN to generate multiple 3D cost volumes to replace the expensive 4D cost volume. At present, the running time of traditional methods and 2D convolution networks are both around 0.05s. Combining these two methods is still far less than 3D convolution networks, where the latter are mainly between 0.32s to 2s. At the same time, by combining the two methods, the accuracy will be greatly improved. Therefore, it is worthwhile to integrate traditional methods and 2D convolution networks.

Our contributions are summarized as follows:

- We propose a matching cost computation method based on traditional methods and CNN.
- We propose a fast and accurate stereo matching network, which is useful for real-time robot applications.
- We construct a novel 2D cascade hourglass network, which can effectively aggregate traditional methods and CNN features.

II. RELATED WORK

Stereo matching algorithms generally perform a four-step pipeline [9]: matching cost computation, cost aggregation, disparity computation, and refinement. This section reviews the most relevant works to ours.

A. Traditional method for Matching cost computation

Traditional methods use the color, brightness, and gradient information of the images for stereo matching, including mutual information [10], Census transform [12], [13], Rank transform [14], Birchfield and Tomasi [15], etc. These methods can quickly generate a cost volume and rough disparity map without any training process, but the accuracy is low. Take Census transform as an example, Census transform is widely used to calculate the cost volume and has many variants. It compares the surrounding pixel values with the center in a rectangular area (the size is \( n \times m \), \( n \) and \( m \) are odd numbers) and maps the result into a sequence. As
shown in Equation 1
\[
\begin{aligned}
C_s(u, v) &= \sum_{i=-n'}^{n'} \sum_{j=-m'}^{m'} \zeta(I(u, v), I(u+i, v+j)) \\
\zeta(x, y) &= \begin{cases} 
0 & \text{if } x \leq y \\
1 & \text{if } x > y 
\end{cases}
\end{aligned}
\]
(1)

where \(n'\) and \(m'\) are the largest integers not greater than half of \(n\) and \(m\) respectively, \(\otimes\) is the bitwise concatenation operation.

After obtaining the census sequence of all pixels in the left and right images, traditional methods calculate the Hamming distance of the census sequences to measure the similarity between the reference pixel and candidate pixels, which may simply require counting the number of 1s in the bitwise XOR of the two sequences. As shown in Equation 2
\[
C(d, u, v) = \text{Hamming}(C_{sl}(u, v), C_{sr}(u-d, v))
\]
(2)

Census transform [12], [13] is well proven for its robust matching, especially along object boundaries, outliers, and radiometric differences. But it abandons the original image information, so it is necessary to use other traditional methods to supplement it. Considering that Census transform has matching ambiguities in repeated or textureless areas, and AD (absolute difference) [16] is sensitive to differences, Mei et al. propose the ADCensus algorithm [8] in combination with AD and Census transform. This method can trade-off speed and accuracy, also ranked first on the Middlebury official website for a long time. Recent Fast DS-CS [2] achieves real-time and accurate effects by combining ADCensus with Unet [17].

B. Neural network for Matching cost computation

MC-CNN [18] proves that only using CNN instead of traditional methods in the matching cost computation step can significantly improve the accuracy of stereo matching. Thus most of the current stereo matching methods use neural networks to calculate the cost volume.

Mayer et al. propose DispNetC [3], a representative matching cost computation method, and use the 1D Correlation layer to generate the cost volume. Concretely, after extracting the left and right feature vectors by a neural network, DispNetC uses a dot product style operation to decimate the feature dimension and generate a cost volume of dimensionality \(H^*W^*D(H, W\) is the height and width of the image, \(D\) is the max disparity value). The 1D Correlation layer is defined as:
\[
C(d, x, y) = \frac{1}{N} (f_l(x, y), f_r(x, y-d))
\]
(3)

where \(f(x, y)\) is the feature vector extracted by neural network at location \((x, y)\). \(\langle,\rangle\) denotes the inner product of two feature vectors, and \(N\) denotes the number of channels.

Although DispNetC [3] provides an efficient way for measuring feature similarities, it loses much information because it produces only a single correlation value for each disparity level. GCNet [5] takes a different approach by concatenating left unary feature with corresponding right feature cross disparity range, and packing these into a 4D volume, which is defined as:
\[
C(d, x, y) = \text{Concat} \{f_l(x, y), f_r(x, y-d)\}
\]
(4)

However, 4D cost volumes need many 3D convolutions for cost aggregation. These networks [5], [7], [19] have expensive computational cost and running time, which is far from meeting the need for real-time applications.

C. Multi-Scale Cost Volumes

Most current stereo matching networks generate one cost volume in matching cost computation. However, 3D cost volumes lose much feature information, while 4D cost volumes require many 3D convolutions, leading to a significant increase in GPU memory and running time. Therefore, it isn’t easy to a trade-off between accuracy and speed by using only a single cost volume. Several works adopt a coarse-to-fine architecture that leverages multi-scale cost volumes to improve accuracy to mitigate this problem. Yang et al. [20] design a hierarchical coarse-to-fine network that uses potential correspondences to gradually build up a pyramid of cost volumes that increase resolution. Gu et al. [21] propose a memory and time efficient cost volume calculation method with gradually higher cost volume resolution and adaptive adjustment of disparity intervals. Similarly, Xu et al. [4] propose AANet, which constructs multi-scale 3D cost volumes by correlating left and right image features at corresponding scales and then aggregates them with several stacked Adaptive Aggregation Modules. All these methods demonstrate that multi-scale cost volumes can achieve competitive accuracy while maintaining fast speed.

III. Method

We present MSCVNet, which mainly contains the following two modules: Multi-scale 3D cost volumes and Guided cascade hourglass network. Unlike previous multi-scale cost volume methods, our network combines traditional methods and CNN to generate cost volumes with different resolutions, thus avoiding feature redundancy caused by a single method, which will be discussed in the ablation study. Meanwhile, our guided cascade hourglass structure can fully extract features of cost volumes and effectively integrate the traditional method and CNN to improve accuracy. The architecture of MSCVNet is illustrated in Fig. 1

A. Multi-scale 3D cost volumes

1) Traditional Method: We first introduce the process of generating cost volume from the traditional method. To reduce the computational cost, we use mean-pooling to adjust the stereo images to 1/2 resolution. Then the images are converted from RGB to YUV. We construct three cost volumes \(C_i\) \(i=1,2,3\) based on YUV color space. For the Y channel data, we use the 5*5 Census transform to
compute the corresponding census sequence of the stereo images. Then we calculate the hamming distance within the maximum disparity range in the same horizontal direction, as mentioned in Equation 2. Afterward, we can get the census cost volume \( C_1 \) with a size of \( 1/2H \times 1/2W \times 96 \). Since the census transform can’t match occlusion and textureless areas well and the AD(absolute difference) method is sensitive to the pixels’ difference, we use AD on the U and V channels to construct the other two cost volumes \( C_2 \) and \( C_3 \). When getting all the cost volumes, we concatenate the feature values across each channel into one 3D tensor of \( 1/2H \times 1/2W \times 288 \). For each pixel \( (x, y) \), the cost can be expressed as follows:

\[
C(x, y) = [C_1(x, y, 0), C_2(x, y, 0), ..., C_3(x, y, 95)]
\]

To speed up the network training, we normalize the above cost volume to satisfy the distribution with zero mean and unit variance. Afterward, we use four 1*1 convolution layers to reduce dimensions to 144, 72, 36, 32. Furthermore, we concatenate the left input image to the succinct low-dimensional cost volume and use three 3*3 convolution layers for feature harvesting. Finally, we can get the traditional cost volume with a size of \( 1/2H \times 1/2W \times 32 \).

2) Neural Network: We use 1D Correlation to construct the other two cost volumes. First, we use a feature extractor to capture multi-scale feature vectors. Our feature extractor is an Unet structure [17], an encoder-decoder with skip connections and learnable parameters. More concretely, we implement downsampling by a 3*3 convolution followed by a 2*2 convolution with the stride of 2. Our up-sampling block is realized by 2*2 deconvolution with the stride of 2. Features of the same resolution are concatenated with skip-connection, followed by a 1*1 convolution for dimensionality reduction and 3*3 convolution for feature harvesting. Each convolution layer uses batch normalization and Relu for non-linearities. We only use the last feature map of each resolution for correlation and further upsample to generate a higher resolution feature map. Thus the high-resolution feature map contains part of spatial context information. Finally, we can obtain multi-scale feature maps of the left and right images.

We only use feature maps with 1/2 and 1/4 resolution for correlation. Their max disparity ranges are 96 and 48, respectively. Therefore, we can obtain the other two cost volumes with a size of \( 1/2H \times 1/2W \times 96 \) and \( 1/4H \times 1/4W \times 48 \). Similar to the traditional method, we use 1*1 convolution to convert the 1/2 resolution cost volume into a low-dimensional vector of \( 1/2H \times 1/2W \times 32 \).

B. Guided cascade hourglass network

Our cost aggregation network mainly consists of the following two modules: Multi-scale guided feature encoder and Cascade hourglass network. The specific implementation of our network is as follows:

1) Multi-scale Guided Feature Encoder: As shown in Fig. 1, we first capture multi-scale guidance features based on the traditional cost volume. Our down-sampling block applies two 3*3 convolutions with the stride of 2 and 1 separately. The deepest feature map reaches a 1/16 spatial resolution after three layers of down-sampling convolutions. Therefore, the traditional cost volume can provide four different scale guide features for the subsequent cascade network.

2) Cascade Hourglass Network: We use an encoder to extract feature vectors from the other two cost volumes, leveraged by a decoder to cooperate with the features generated from traditional cost volume. The architecture of one cascade hourglass network is shown in Fig. 2.

The first hourglass network uses a series of ResNet [22] blocks with a stride of 2 to gradually downsize the feature resolution to 1/16. Then the decoder consists of three up-sampling blocks to gradually increase the feature resolutions.
which can be expressed as follows: to the entire disparity map. The disparity map and coordinate value. Due to occlusion areas, warped disparity image discontinuous areas of any disparity map. We first generate changing the weights. Our network can improve the accuracy of different areas by extracting deep features.

After the two cascade hourglass networks, we can obtain an optimized cost volume with a size of $1/2H \times 1/2W \times 32$. Then we use a 1*1 convolution to generate the disparity map directly and use bilinear interpolation to upsample this map to the full resolution.

C. Training Loss

We propose a novel algorithm, which divides the disparity map into two parts: monotonically increasing areas (including occlusion), and discontinuous areas, as shown in Fig. 3 Thus our network can improve the accuracy of different areas by changing the weights.

Our algorithm uses the following steps to calculate the discontinuous areas of any disparity map. We first generate warped disparity image $Warp(D)$ according to the left disparity map and coordinate value. Due to occlusion areas, $Warp(D)$ isn’t monotonically increasing, and discontinuous areas will appear at the edge of some objects. Simply put, we analyze a row of pixels in $Warp(D)$, which can be extended to the entire disparity map.

First, we assume that $Y$ is a row of pixels in $Warp(D)$, which can be expressed as follows:

$$Y = \{Y_1, Y_2, ..., Y_m, y_1, y_2, ..., y_n, Y_k, Y_{k+1}, ...\}$$

where $Y_1 < Y_2 < ... < Y_m < Y_k, y_1 < y_2 < ... < y_n < Y_m$. As shown in Fig. 3 for large objects, such as trees, cars, houses, etc. the disparity of these areas will suddenly decrease rather than suddenly increase. Their discontinuous disparity areas are $Y_m, y_1$. For small objects, such as railings, grass and telephone poles, etc. the disparity of these areas suddenly decreases and increases. Their discontinuous disparity areas are $Y_m, y_1$ and $y_n, Y_k$. We distinguish these two cases by comparing the value of $Y_k - y_n$ and $\epsilon$.

The output of our algorithm is:

$$Output = \begin{cases} 0, 0, ..., 1, 1, 0, ..., 1, 1, 0, ... & Y_k - y_n \leq \epsilon \\ 0, 0, ..., 1, 1, 0, ..., 0, 0, 0, ... & Y_k - y_n > \epsilon \end{cases}$$

We first calculate the maximum pixel values of the current coordinate:

$$Y_{max} = \{Y_1, Y_2, ..., Y_m, Y_m, Y_m, ..., Y_m, Y_k, Y_{k+1}, ...\}$$

Then we calculate the absolute difference with $Y$ and analyze the pixels greater than 0 and $\epsilon$:

$$Mask(Y_{max} - Y) = \{0, 0, ..., 0, 1, 1, ..., 1, 0, 0, ...\}$$

After that, we move Mask to the left and right by one bit and calculate the absolute difference with the original Mask.

$$|Mask_{>1} - Mask| = \{0, 0, ..., 0, 1, 0, ..., 0, 1, 0, ...\}$$

$$|Mask_{<1} - Mask| = \{0, 0, ..., 1, 0, 0, ..., 1, 0, 0, ...\}$$

Finally, we can get the Output by adding the two results together.

Based on Fast DS-CS [2], our loss function is defined as:

$$L(d_{GT}, \hat{d}) = \max \left( \tau, (d_{GT} - \hat{d}) * (1 - \lambda * Output) \right)^{1/8}$$

(5)

where $\hat{d}$ and $d_{GT}$ denote the final disparity map from our network and the ground truth, respectively. $Output$ denotes
TABLE I
Performance comparison on the Scene Flow and KITTI.

| Method      | Scene Flow | KITTI2012 | KITTI2015 |
|-------------|------------|-----------|-----------|
|             | EPE        | >3px      | >3px      | All pixels | Non-Occluded pixels |
|             | Out-Noc    | Out-All   | D1-bg     | D1-fg      | D1-all    | D1-bg | D1-fg   | D1-all |
| MC-CNN [18] | 3.79       | 2.43      | 3.63      | 1.64       | 2.39      | 2.89  | 8.88    | 3.89   | 2.48 | 7.64 | 3.33 |
| GC-Net [5]  | 2.51       | 1.77      | 2.30      | 1.12       | 1.46      | 2.21  | 6.16    | 2.87   | 2.02 | 5.58 | 2.61 |
| PSMNet [6]  | 1.09       | 1.49      | 1.89      | 0.90       | 1.15      | 1.86  | 4.62    | 2.32   | 1.71 | 4.31 | 2.14 |
| GANet [19]  | 0.84       | 1.19      | 1.60      | 0.76       | 1.02      | 1.48  | 3.16    | 1.81   | 1.34 | 3.11 | 1.63 |
| SPS-St [23] | -          | 3.39      | 4.41      | 2.33       | 3.00      | 3.84  | 12.67   | 5.31   | 3.5  | 11.61| 4.84 |
| FADNet [24] | 0.83       | 2.42      | 2.86      | 1.34       | 1.62      | 2.68  | 3.50    | 2.82   | 2.49 | 3.07 | 2.59 |
| RTSNet [25] | -          | 2.43      | 2.90      | 1.42       | 1.72      | 2.86  | 6.19    | 3.14   | 2.67 | 5.83 | 3.19 |
| Fast DS-CS [2] | -       | 2.61      | 3.20      | 1.46       | 1.85      | 2.83  | 4.31    | 3.08   | 2.53 | 3.74 | 2.73 |
| DispNetC [3] | 1.68     | 4.11      | 4.65      | 2.05       | 2.39      | 4.32  | 4.41    | 4.34   | 4.11 | 3.72 | 4.05 |
| MSCVNet     | 1.32       | 2.25      | 2.81      | 1.37       | 1.74      | 2.31  | 5.41    | 2.82   | 2.12 | 5.02 | 2.60 |

IV. EXPERIMENTS
We perform extensive experiments on popular benchmarks to verify our network’s effectiveness, including a comparison to related works and ablation studies. We first conduct pre-training on the Scene Flow [3], and then fine-tuning on KITTI2012 [26] and KITTI2015 [27] respectively. After that, we pass the final model through the test set and submit the results to the KITTI official website for comparison. Meanwhile, we perform ablation studies using KITTI2015 to evaluate the influence on performance and running time made by different cost volumes, the number of cascade hourglass networks, and different loss functions. The experimental settings and network details are presented in Section 4.1, followed by the evaluation results on the three popular stereo datasets.

A. Network Details
We implement our network in Tensorflow and use Adam as optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$). We use an NVIDIA GTX 1080Ti GPU for training with a batch size of 4. Since many convolutions in our network don’t use the BN layer, we set the maximum learning rate to $10^{-4}$. We begin by training for 500k iterations on Scene Flow with a learning rate of $10^{-4}$. For KITTI, we use the final model pre-trained on Scene Flow and then fine-tune for 100k iterations and another 50k with a learning rate of $10^{-4}$ and $10^{-5}$ respectively. We combine the KITTI2012 and KITTI2015 datasets and train them together. We train the final model for 50k with a learning rate of $2 \times 10^{-5}$ on KITTI2012 and KITTI2015 separately. We also conduct data augmentation in the following ways: For Scene Flow and KITTI2015, since these two datasets provide ground-truth of both left and right images, we can get additional training data by flipping the stereo images and the right disparity map horizontally. Meanwhile, we use random cropping and vertical flipping for the training data of KITTI and Scene Flow.

B. Results on Scene Flow
We first test on the Scene Flow [3], Scene Flow is a large scale synthetic dataset which consists of 35454 training and 4370 testing images with $H = 540$ and $W = 960$. This dataset provides dense disparity maps as ground truth. The end-point error (EPE) and 1-pixel error are regarded as the metrics in our experiment, where EPE is the mean disparity error in pixels, and 1-pixel error is the percentage of pixels whose EPE is bigger than 1 pixel. After training for 500k iterations, we compare the performance with other latest stereo matching methods. The final results are shown in Table I. As shown in the table, our
MSCVNet can achieve a competitive result at a fast speed. Since we only pre-train on the Scene Flow without fine-tuning, the performance can be further improved. We mainly conduct a detailed comparison and analysis on KITTI2012 and KITTI2015.

C. Results on KITTI

KITTI is a real-world dataset with street scenes from a driving car. This dataset provides sparse but accurate dense disparity maps as ground truth. Image size is $H = 376$ and $W = 1240$. For KITTI2012 [26], it consists of 200 stereo images with ground-truth disparities for training and another 200 image pairs without ground-truth disparities for testing. For KITTI2015 [27], there are 194 stereo images for training and 195 for testing. During training, we combine KITTI2012 and KITTI2015 and divide the whole training images into a training set with 354 image pairs and a validation set with 40 image pairs (20 from KITTI2012 and 20 from KITTI2015). We evaluate our method using official metrics. For KITTI2012, we use Out-Noc and Out-All as metrics. They denote the percentage of erroneous pixels in non-occluded areas (Out-Noc) and total areas (Out-All). For KITTI2015, we use D1-bg, D1-fg, D1-all as metrics. They compute the percentage of stereo disparity outliers with errors greater than 3 pixels for the background (D1-bg), foreground (D1-fg), and all (D1-all) pixels, respectively. After fine-tuning for 50k iterations, we report the official results along with running time in Table II.

As shown in the table, our method can make a trade-off between speed and accuracy. Although our MSCVNet is little less accurate than top-performance methods, such as GCNet [5], PSMNet [6], GANet [19], it is our advantage that we can achieve real-time at 41ms. Meanwhile, compared with the traditional method (SPS-St [23]) and other fast networks, we achieve significant performance improvement. In particular, compared with Fast DS-CS [2] based on ADCensus and DispNetC [3] based on 1D Correlation, our MSCVNet obviously better than them in all evaluation metrics on KITTI2012, and only a little worse than Fast DS-CS in D1-fg on KITTI2015, demonstrating that integrating ADCensus and 1D Correlation is effective. Our network can achieve better performance than FADNet [24] while using less time. We also visualize the results in Fig. 4 to further prove our method’s effectiveness.

D. Ablation Study on KITTI2015

We test the impact of different settings on the performance from the following aspects. First, we compare the performance difference between Multi-scale cost volumes and a single cost volume. Then we only use 1D Correlation to construct the Multi-scale cost volumes and compare the performance with our method. Finally, we test the performance difference caused by the number of cascade hourglass networks and different loss functions.

As shown in Table II by comparing 1, 2 with 3, we can find that combining ADCensus and 1D Correlation can significantly improve accuracy while only increases 10ms. By comparing 3 with 4, we find that using two hourglass networks is better than one. This is obviously due to the increase of network layers. From the results of 4 and 5, it can be seen that using different methods to generate cost volume is significantly better than using only one method. We infer that this is due to redundancy in the network. Finally, by comparing 4 with 6, we can prove that calculating the loss for different areas can improve accuracy. All these results confirm that our network is efficient.

V. Conclusion

This paper proposes a fast and accurate stereo matching method for real-time robot applications. We integrate the traditional method and CNN to improve the quality of the 3D cost volume and construct a novel cascade hourglass network for cost aggregation. We also design a novel algorithm for loss function by distinguishing discontinuous disparity areas. Results on Scene Flow and KITTI demonstrate the effectiveness of our MSCVNet. We can achieve a competitive result with a significantly fast speed at 41ms. We attempt to expand our work to multi-view stereo and combine depth completion to achieve a more accurate effect in future work.

Acknowledgement

This work was supported by the National Key Research and Development Program of China (No. 2018YFB0204301), General Program of National Natural Science Foundation of China (81973244), Science and Technology Program Projects of Shenzhen (JCYJ20170818110101726).

| Cost Volume         | Cost Aggregation | Loss  | KITTI2015 |
|--------------------|------------------|-------|-----------|
| ADCensus           | 1D Correlation    | $\lambda$ | PC3(%), EPE, time(s) |
| 1                  | ✓                | ✓     | 97.34, 0.78, 0.021 |
| 2                  | ✓                | ✓     | 97.78, 0.73, 0.023 |
| 3                  | ✓                | ✓     | 98.05, 0.64, 0.035 |
| 4                  | ✓                | ✓     | 98.31, 0.61, 0.041 |
| 5                  | ✓                | ✓     | 97.93, 0.74, 0.037 |
| 6                  | ✓                | ✓     | 98.54, 0.59, 0.041 |

This work was supported by the National Key Research and Development Program of China (No. 2018YFB0204301), General Program of National Natural Science Foundation of China (81973244), Science and Technology Program Projects of Shenzhen (JCYJ20170818110101726).
