Asymmetric Learning for Stereo Matching Cost Computation

Zhongjian MA†,††a), Student Member, Dongzhen HUANG†, Baoqing LIb), and Xiaobing YUAN†c), Nonmembers

SUMMARY Current stereo matching methods benefit a lot from the precise stereo estimation with Convolutional Neural Networks (CNNs). Nevertheless, patch-based siamese networks rely on the implicit assumption of constant depth within a window, which does not hold for slanted surfaces. Existing methods for handling slanted patches focus on post-processing. In contrast, we propose a novel module for matching cost networks to overcome this bias. Slanted objects appear horizontally stretched between stereo pairs, suggesting that the feature extraction in the horizontal direction should be different from that in the vertical direction. To tackle this distortion, we utilize asymmetric convolutions in our proposed module. Experimental results show that the proposed module in matching cost networks can achieve higher accuracy with fewer parameters compared to conventional methods.

key words: stereo matching, asymmetric convolutions, feature extraction, CNN

1. Introduction

Binocular stereo vision is quite charming as it works like the human visual system. The depth of an object in an image pair can be calculated directly from its disparity. Therefore, the core task of the stereo method is to find the one-to-one correspondence between pixels. Methods that solve the stereo matching problem can be divided into two categories: cost filtering methods and energy minimization methods[1]. Scharstein and Szeliski[2] summarized the steps of the stereo matching algorithms as follows: matching cost computation, cost aggregation, disparity computation/optimization, and disparity refinement. Local stereo methods follow the pipeline[3],[4], while global stereo methods skip the cost aggregation step and formulate the stereo matching problem in terms of energy minimization[5]–[8]. Global stereo methods compute disparities of all pixels simultaneously, achieving higher accuracy but consuming more time.

Nowadays, CNNs have been widely applied in many image processing fields, including stereo vision. To compute the matching cost, Zbontar et al. [9] employed a siamese network with several traditional post-processing steps and obtained impressive results. Taking advantage of the CNN similarity computation[9], many advanced stereo methods have been proposed[10]–[12]. In contrast to the patch-based methods, end-to-end networks[13],[14] are proposed to estimate per-pixel disparity straightforwardly, without any post-processing or regularization. However, end-to-end networks require higher computational resources, especially memory space. It’s hard for the current embedded platforms like NVIDIA Jetson to run these end-to-end networks while patch-based siamese networks are more affordable. In this paper, we focus on patch-based networks.

Similar to the window-based stereo matching algorithms, patch-based siamese networks take a stereo image pair of fixed size as input and output the matching cost of them. This architecture makes an implicit assumption that an object or a part of an object keeps its shape in a stereo pair, which does not apply to the case of a slanted surface. Figure 1 intuitively illustrates the problem mentioned above.

A classic local method to relax this assumption is to estimate a local plane in each pixel instead of its disparity[15]. This technique can also be used to build a data term for
global stereo methods [10], [11]. However, current patch-
based CNN methods can not effectively handle slanted sur-
faces by integrating this technique. In this paper, we pro-
pose a novel module that takes slanted surfaces into account.
Motivated by the directional asymmetry in the slanted sur-
faces, we leverage a horizontally asymmetric convolution in
the proposed module to cope with the horizontal distort-
tions. To verify the effectiveness of the proposed module,
we compare our module against the residual blocks [16]
on the KITTI 2012 [17] and KITTI 2015 datasets [18]. Both
modules are stacked properly on the MC-CNN-fst archite-
ture [9] with the same depth. The experimental results show
that both modules are effective, while the proposed module
can achieve better results with fewer parameters.

The main contributions of this paper are as follows: (1)
we leverage asymmetric convolutions to deal with the distor-
tion caused by slanted surfaces without considering the im-

cipient fronto-parallel assumption of patch-based CNN

do
t
t
t
t

methods; (2) we integrate asymmetric convolutions into the resid-

d

d

d

ual framework to enjoy the accuracy gains from increased

network depth effectively.

2. Asymmetric Learning

The entire network takes a stereo image pair as input and
outputs the matching cost of them. In the following subsec-
tions, the special phenomena about directional asymmetric
image warp are first introduced. Detailed descriptions of the
proposed module, deepened network architecture, and post-
processing methods are presented in the remaining subsections.

2.1 Horizontal Feature Pollution

We divide the matching cost computation into two steps:
feature extraction and feature matching. In MC-CNN [9],
the convolutional part acts as a feature extractor, and the
inner product of MC-CNN-fst or the fully-connected part of
MC-CNN-act acts as a feature matcher.

In contrast to the 4P module [19], which works in the
feature matcher, our work focuses on the descriptor. A pre-
cise feature descriptor is essential for obtaining an accurate
matching cost. However, due to the shared weights of the
siamese network, it becomes harder for convolutional lay-
ers to handle the simple image warp between stereo pairs.
Given a horizontal one-dimensional (1D) vector pair ex-
tracted from input patches for brevity, the simplest way to
transform one vector into another is to perform an affine
transformation: \( f(x) = Mx + b \). But if the operation is per-
formed on both vectors and in each pixel, as siamese net-
works do, it may become much more complex. Inevitably,
pixels that appear in only one patch will pollute the feature
vectors during convolution computation. Since slanted sur-
faces only lead to horizontal image warp for stereo pairs,
the pollution occurs solely on the horizontal vectors. This
nature, which differs in direction, is not utilized in current
patch-based networks and motivates our design. Experi-
mental results show that the proposed module in the match-
ing cost networks achieves more competitive performance.

2.2 Asymmetric Blocks

Inception-v2 [20] introduced the spatial factorization into
asymmetric convolutions. Replacing \( n \times n \) convolutions with
a pair of stacked convolutions \( n \times 1 \) and \( 1 \times n \) could re-
duce the computational cost \( 2/n \) of the original. Inspired
by the spatial factorization, we try to tackle the horizontal
feature pollution by utilizing asymmetric convolutional lay-
ers. Considering that a plain two-dimensional (2D) convolu-
tion function \( F(x) \) can be approximately factorized into
a horizontal 1D convolution and a vertical 1D convolution
\( F(x) := v_1(v_2x + b_2) + b_1 \), we hypothesize that asymmetric
convolutions can be used to approximate the directionally
asymmetric transformation. Thus, instead of using asym-
metric convolutions in pairs, we only leverage horizontal
asymmetric convolutions in our module to deal with the im-

age distortion in the horizontal direction. We can compute a
typical output element \( m_{i,j} \) of an asymmetric convolutional
layer according to Eq. (1):

\[
m_{i,j} = v \cdot x_{i,j,k} + b,
\]

where \( v \) indicates the \( k \times 1 \) convolution kernel, \( \cdot \) is the inner
product, \( x_{i,j} \) is the pixel of the input patch, and \( b \) is a bias


ter. Further, we formally define the asymmetric block as

\[
y = \sigma(w_1v(x + b_1) + b_2) + x,
\]

where \( \sigma \) denotes ReLU (Rectified Linear Unit) [21], \( w \) de-
notes the plain \( n \times n \) convolution kernel, \( v \) indicates the \( n \times 1 \)

convolution kernel, \( b_1 \) and \( b_2 \) are the biases.

As a part of preprocessing, we integrate asymmetric
convolutions into the residual architecture [16] to tackle the
degradation problem in which the network with stacked
asymmetric convolutional layers also suffer. Besides, the
preprocessing consisting solely of asymmetric convolutions
performed worse than those consisting of residual blocks
due to the limitation of a single receptive field dimension.
Therefore, we further integrate plain convolutions into our
blocks to widen the receptive field. As shown in Fig. 2 (d),
an asymmetric block contains one asymmetric convolutional
layer and one plain convolutional layer for simplicity. As
mentioned in [9], the number of convolution layers implicit-
ly controls the size of image patches. We set the input
matching patch size to \( 9 \times 9 \) and the convolution kernel size
to \( 3 \times 3 \), which is the same as the setting in [9]. Other
combinations may work better; the concise combination works
well enough to prove the superiority of asymmetric blocks.

As shown in Fig. 3, although the training error of the
matching cost network with asymmetric blocks is higher,
the validation error is lower, that is, the higher validation
accuracy is obtained, which indicates that the asymmetric
network architecture is more robust to overfitting for stereo
matching. We can get similar results with the other number
2.3 Deepened Matching Cost Networks

As shown in Fig. 2, we deepen the matching cost networks with residual/asymmetric blocks to verify the effectiveness of asymmetric learning. We choose MC-CNN-fst[9] as the basic framework of networks for its lightweight feature matcher in the experiments. The first layer of the additional module is a plain $3 \times 3$ convolution, as the skip connections will introduce many repeated elements when the dimensions increase from 1 to 64. Then, several residual/asymmetric blocks are stacked to extract more precise features. Batch normalization is discarded in both blocks to save running time. The last part of the extractor is composed of 4 layers of unpadded $3 \times 3$ convolution, reducing the size of the feature map from $9 \times 9$ to $1 \times 1$ gradually. Each convolutional layer in the networks is followed by a ReLU expect the last one, and the number of the feature maps in each layer is 64. The cosine similarity between two feature vectors extracted is finally computed as the matching cost by normalization and an inner product.

In our experience, stacking residual/asymmetric blocks on the feature map of different sizes also improves the feature extraction, but the accuracy gain gradually attenuates as the size reduces. On the $3 \times 3$ feature maps, adding additional blocks can barely improve the performance. Instead of obtaining the higher accuracy by stacking as many blocks as possible, we employ additional module only on the $9 \times 9$ feature maps to compare the performance of residual/asymmetric blocks.

2.4 Post-Processing Methods

We follow the post-processing methods in [9]: cross-based cost aggregation [22], Semi-Global Matching (SGM) [23], interpolation, sub-pixel enhancement, median filtering, and bilateral filtering.

As mentioned in [24], stacking residual blocks does not improve the quality of CNN output. But if full post-processing methods are applied, the output results of networks with stacked residual blocks are observably better. Our experiments demonstrate accuracy gains with the stacked blocks. We note that the error rates of networks with asymmetric blocks are higher than those with residual blocks when the post-processing methods, especially the SGM, are removed. That is, traditional post-processing steps are essential for our method.

3. Experimental Analysis

3.1 Baseline Network Architectures

We construct the matching cost networks based on the MC-CNN-fst architecture [9] and apply the same post-processing methods with their best parameters as set in [9]. We deepen the networks by adding several blocks. The performance of networks is measured on the KITTI 2012 [17] and KITTI 2015 datasets [18] by the percentage of mismatched pixels, and the runtime for inference. We implemented all experiments on the computer with an NVIDIA 1080Ti GPU and an Intel Xeon E5-2620 CPU.

The submissions to the online test server are limited...
to significantly different methods. Thus we evaluate these networks by validation error rates instead of test error rates. We follow the training strategies presented by [9] along with its data augmentation. All the networks are trained from scratch by random initialization. Considering that there are more parameters to be learned, we properly increase the training epochs from 14 to 16. We compute the validation error rate at the end of each epoch and record the lowest one.

The KITTI 2012 and KITTI 2015 datasets contain 194 and 200 training image pairs, respectively. Both of them are further randomly split into the validation set of 40 image pairs and the rest as the training set.

### 3.2 Experiments on the KITTI 2012 Dataset

Table 1 presents the performance comparisons of the networks with different additional modules on the KITTI 2012 dataset. The “Number of additional blocks” indicates the number of the stacked residual/asymmetric blocks on the MC-CNN-fst architecture. The “Res” indicates the matching cost networks with residual blocks containing the 3 × 3 convolution kernel, the “Asy-H” indicates the matching cost networks with horizontal asymmetric blocks containing the 3 × 1 convolution kernel, and the “Asy-V” indicates the matching cost networks with vertical asymmetric blocks containing the 1 × 3 convolution kernel. The “Error rate” reports the percentage of misclassified pixels, and the “Runtime” measures the time, in seconds, spent on processing one pair of images without post-processing.

<table>
<thead>
<tr>
<th>Number of additional blocks</th>
<th>Error rate (%)</th>
<th>Runtime (s)</th>
<th>Error rate (%)</th>
<th>Runtime (s)</th>
<th>Error rate (%)</th>
<th>Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.82</td>
<td>0.222</td>
<td>2.79</td>
<td>0.220</td>
<td>2.83</td>
<td>0.220</td>
</tr>
<tr>
<td>2</td>
<td>2.76</td>
<td>0.240</td>
<td>2.75</td>
<td>0.236</td>
<td>2.74</td>
<td>0.236</td>
</tr>
<tr>
<td>3</td>
<td>2.74</td>
<td>0.257</td>
<td>2.72</td>
<td>0.251</td>
<td>2.73</td>
<td>0.251</td>
</tr>
<tr>
<td>4</td>
<td>2.73</td>
<td>0.274</td>
<td>2.71</td>
<td>0.266</td>
<td>2.73</td>
<td>0.266</td>
</tr>
<tr>
<td>5</td>
<td>2.72</td>
<td>0.291</td>
<td>2.71</td>
<td>0.281</td>
<td>2.72</td>
<td>0.281</td>
</tr>
<tr>
<td>6</td>
<td>2.71</td>
<td>0.308</td>
<td>2.69</td>
<td>0.296</td>
<td>2.71</td>
<td>0.297</td>
</tr>
</tbody>
</table>

Compared to the original validation error rate of 3.02% [9] on the KITTI 2012 dataset, the deepened networks achieve observably higher accuracy. With the same depth, networks with horizontal asymmetric blocks have lower validation error rates than networks with residual blocks, while networks with vertical asymmetric blocks do not, demonstrating that the direction of asymmetric blocks is crucial for the task to undertake. Instead of performing a horizontal affine transformation directly, another way to handle the horizontal image distortion is to warp the images vertically and then scale the stereo pair to the same size. Therefore, we further design the vertical asymmetric block to approximate such a transformation. Experimental results in Table 1 show that the indirect method is harder to approximate. The networks with vertical asymmetric blocks have similar error rates to the networks with residual blocks. In summary, networks with horizontal asymmetric blocks can achieve the highest accuracy, which verifies the effectiveness of the proposed blocks.

### 3.3 Experiments on the KITTI 2015 Dataset

The strategy of stacking blocks in the experiments on the KITTI 2015 dataset is a little different. We observe that the network gains considerable accuracy from the first several stacked residual blocks. But when we replace residual blocks with asymmetric blocks, the accuracy gain becomes much less. We argue that this is because the network is too shallow to learn strong representations. Therefore, we first stack four residual blocks on MC-CNN-fst [9] as the initial network. Then the additional module composed of residual/asymmetric blocks and a plain convolutional layer is added to the initial network for comparison.

In Table 2, the “Res” indicates the matching cost networks with residual blocks, and the “Asy-H” indicates the matching cost networks with horizontal asymmetric blocks. The “5 additional blocks” indicates the network deepened with one plain convolutional layer, one residual/asymmetric block, and four residual blocks. The “6 additional blocks” means the number of residual/asymmetric block increases from 1 to 2, and so on. Compared to the original validation error rate of 3.99% [9] on the KITTI 2015 dataset, the experimental results show that the deepened networks with horizontal asymmetric blocks can achieve higher accuracy, which also verifies the effectiveness of the proposed blocks.

### Table 1

The performance comparisons of different network architectures on the KITTI 2012 dataset. Compared to the original validation error rate of 3.02% [9] on the KITTI 2012 dataset, the deepened networks achieve observably higher accuracy. The networks with horizontal asymmetric blocks can achieve the highest accuracy.

<table>
<thead>
<tr>
<th>Number of additional blocks</th>
<th>Error rate (%)</th>
<th>Runtime (s)</th>
<th>Error rate (%)</th>
<th>Runtime (s)</th>
<th>Error rate (%)</th>
<th>Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.82</td>
<td>0.222</td>
<td>2.79</td>
<td>0.220</td>
<td>2.83</td>
<td>0.220</td>
</tr>
<tr>
<td>2</td>
<td>2.76</td>
<td>0.240</td>
<td>2.75</td>
<td>0.236</td>
<td>2.74</td>
<td>0.236</td>
</tr>
<tr>
<td>3</td>
<td>2.74</td>
<td>0.257</td>
<td>2.72</td>
<td>0.251</td>
<td>2.73</td>
<td>0.251</td>
</tr>
<tr>
<td>4</td>
<td>2.73</td>
<td>0.274</td>
<td>2.71</td>
<td>0.266</td>
<td>2.73</td>
<td>0.266</td>
</tr>
<tr>
<td>5</td>
<td>2.72</td>
<td>0.291</td>
<td>2.71</td>
<td>0.281</td>
<td>2.72</td>
<td>0.281</td>
</tr>
<tr>
<td>6</td>
<td>2.71</td>
<td>0.308</td>
<td>2.69</td>
<td>0.296</td>
<td>2.71</td>
<td>0.297</td>
</tr>
</tbody>
</table>

### Table 2

The error rates (%) of different network architectures on the KITTI 2015 dataset. Compared to the original validation error rate of 3.99% [9] on the KITTI 2015 dataset, the experimental results show that the deepened networks with horizontal asymmetric blocks can achieve higher accuracy.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Number of additional blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Res</td>
<td>3.61</td>
</tr>
<tr>
<td>Asy-H</td>
<td>3.60</td>
</tr>
</tbody>
</table>
3.4 Some Qualitative Results on the KITTI 2012 Validation Set

To the best of our knowledge, most scene objects are more or less slanted, and many weak texture regions and repetitive texture regions of the fence, eaves, etc. may simultaneously contain slanted surfaces. From the error maps shown in Fig. 4, we observe that the strategy of deepening networks can improve matching performance in the weak texture regions and the repetitive texture regions containing slanted surfaces. Moreover, the deepened network with horizontal asymmetric blocks outperforms the original MC-CNN-fst [9] and the deepened network with residual blocks.

4. Discussion

The fronto-parallel assumption in window-based stereo matching does not hold for dealing with the slanted surfaces. Analogously, the existing patch-based CNN methods also rely on this implicit assumption, and conventional methods mainly focus on post-processing. In this paper, we propose a novel module with asymmetric blocks as a part of preprocessing to tackle the distortion problem caused by the slanted surface in the matching cost computation of the stereo matching. Experimental results on the KITTI 2012 and KITTI 2015 datasets have shown the effectiveness of the proposed module quantitatively and qualitatively. With the same depth, networks with horizontal asymmetric blocks can achieve higher accuracy and can reduce computational complexity. In future work, we will conduct more experiments on the Middlebury dataset [25] and other larger datasets to verify the effectiveness of the proposed method and compare it with the classic approaches to handle the stereo matching cost.

References


