Deep self-guided cost aggregation for stereo matching

Williem a, In Kyu Park b,∗

aComputer Science Department, School of Computer Science, Bina Nusantara University, Jakarta, Indonesia 11480
bDepartment of Information and Communication Engineering, Inha University, Incheon 22212, Republic of Korea

A R T I C L E   I N F O

Article history:
Received 22 September 2017
Available online 6 July 2018

MSC:
41A05
41A10
65D05
65D17

Keywords:
stereo matching
cost aggregation
deep learning
guided-filter

A B S T R A C T

In this paper, we present a deep self-guided cost aggregation method used to obtain an accurate disparity map from a pair of stereo images. Conventional cost aggregation methods typically perform joint image filtering on each cost volume slice. Thus, a guidance image is necessary for the conventional methods to work effectively. However, a guidance image might be unreliable due to several distortions, such as noise, blur, radiometric variation. Based on our observations, each cost volume slice can guide itself based on the internal features. However, finding a direct mapping function from the initial and filtered cost volume slice without any guidance image is difficult. To solve this problem, we use an advanced deep learning technique to perform self-guided cost aggregation. Because of the absence of ground truth cost volume, we offer the solution for the dataset generation. Our proposed deep learning network consists of two sub-networks: dynamic weight network and descending filtering network. We integrate the feature reconstruction loss and the pixelwise mean square loss function to preserve the edge property. Experimental results show that the proposed method achieves better results even though it does not employ a guidance image.

© 2018 Elsevier B.V. All rights reserved.

1. Introduction

During the last few decades, stereo matching has been a popular means of finding a correspondence map between two rectified images. Extensive studies have been conducted to obtain either fast or accurate stereo matching results [5,8,10,29,32]. In general, a stereo matching method consists of four steps: cost computation, cost aggregation, disparity computation, and disparity refinement [25]. In local or global stereo matching, cost aggregation plays a major role. Since Yoon and Kweon [33] introduced an adaptive method that achieves comparable results to those of global stereo matching, several cost aggregation methods have been developed [11,27]. Most of these methods have a similar concept to that of joint image filtering that utilizes a guidance image. Thus, the existence of a guidance image is inevitable. These methods assume that the weights of corresponding pixels on a color image are approximately constant to those on a disparity image. However, this assumption often fails to hold true when a patch with a high texture variance has a similar disparity value. In addition, the conventional methods fail when the guidance image is unreliable. Various stereo matching methods have been introduced to deal with unreliable images due to noise [7], radiometric variation [13,21], blur [2,30], and severe weather condition [19].

In this paper, we propose a novel cost aggregation method that removes the dependency on the guidance image. We observe that the initial cost volume possesses specific features that act as a guidance for itself. However, finding a correlation between non-filtered and filtered cost volume without any guidance image is complicated. Therefore, we employ a deep convolutional neural network (CNN) in our approach to enable a cost volume slice to perform a self-guided cost aggregation. Fig. 1 illustrates the conceptual differences between the conventional and proposed cost aggregation methods. We introduce two sub-networks, dynamic weight network and descending filtering network, to estimate the importance weight for each pixel and simultaneously to perform edge-aware filtering operator. To the best of our knowledge, the proposed work is both the first self-guided and first local cost aggregation method based on the deep learning approach. In summary, this paper makes the following contributions.

- We introduce a self-guided cost aggregation method for stereo matching that does not require any guidance color image.
- We propose a novel deep convolutional network consisting of two sub-networks: dynamic weight network and descending filtering network.

This paper is organized as follows. We introduce the related works in Section 2. Section 3 describes the detail of the proposed
method. Experimental results are presented in Section 4. Finally, we conclude in Section 5.

2. Related works

Stereo matching has been an active research topic during the last few decades. Scharrstein and Szeliski [25] surveyed various stereo matching techniques, a study that later became the main guideline for stereo matching research. Although four main steps are used in stereo matching, the focus of this study is solely on the cost aggregation step. Tombari et al. [27] were the first to evaluate the cost aggregation step. They classified various cost aggregation strategies and performed numerous experiments on known techniques. Hosni et al. [11] extended the work in [27] by adding new adaptive support weight (ASW) techniques, evaluating larger datasets, and providing general insights about ASW techniques. They showed that cost aggregation using a guided filter [12] performed better than other methods in terms of accuracy and computational time.

In early studies on cost aggregation, Yoon and Kweon [33] utilized a symmetric bilateral filtering method to filter 3D cost volume during the cost aggregation step. The weight for each pixel was computed based on the color similarity and spatial distance from the center pixel inside the support window. Their method performed better than previous local algorithms and comparably to global matching algorithms. Zhang et al. [36] introduced a method to generate a shape-adaptive support region based on the upright cross local support skeleton. The cross skeleton was reconstructed based on color similarity and connectivity constraints. In addition, the researchers introduced an efficient means to perform cost aggregation using orthogonal integral images. Hosni et al. [12] proposed an efficient cost volume filtering method that outperformed [33] in terms of both quality and computational speed. They employed the most popular edge-aware filter known at the time, called guided filter [6]. Pham and Jeon [20] employed the domain transform [4] to perform edge-aware cost aggregation.

Instead of reducing the complexity with respect to matching window size, Min et al. [18] reduced the computational redundancy among the search spaces. They proposed an efficient cost aggregation method based on a joint histogram. Two approximations of a search space and window size were used simultaneously for fast computation. Yang [31] introduced a non-local cost aggregation method based on a minimum spanning tree (MST). They inserted a full image into a tree structure that each pixel in the image could efficiently contribute to other similar pixels. However, their method depends on the local weight which is set as an edge in the tree structure. To improve performance, Mei et al. [15] proposed a novel tree structure called a segment tree (ST). They generated a sub-tree for each segment in an image and then connected those sub-trees to form a fully segmented tree. They considered not only local edge weights but also non-local segment properties for each pixel contribution.

Although most methods have been developed in a single scale, [35] and [3] proposed an integrated solution of various scale cost aggregation methods. Zhang et al. [35] introduced an optimization function that employs a regularizer to aggregate cost across multiple scales. Choi and Chang [3] determined pixelwise mixing coefficients of each scale filter adaptively. These coefficients were obtained by performing supervised learning. Both of the aforementioned methods outperformed the single scale cost aggregation methods.

Note that most conventional methods require a guidance image in order to perform cost aggregation. In this paper, our focus is to develop a novel cost aggregation method that is free of a guidance image. Note that the absence of the guidance image is a big handicap because much less information exists to perform edge-aware filtering. Therefore, we adopt the deep learning approach during the cost aggregation step. To the best of our knowledge, the proposed method is the first deep learning based and self-guided cost aggregation method.

3. Proposed method

3.1. Stereo matching

In our study, we focus on local stereo matching to evaluate the performance of the proposed method. We employ Census filter [34] to compute the matching cost, which is proven to be robust when illumination varies [9]. The matching cost is described as follows.

\[
F_{\mathbf{j}} \mid \mathbf{K} = \begin{cases} 1 & \text{if } I_{\mathbf{i}} < I_{\mathbf{j}} \\ 0 & \text{otherwise} \end{cases}
\]

where \( i \) and \( N_i \) are the coordinate of a center pixel and its pixel set in the support region, respectively. \( I_{\mathbf{i}} \) and \( F_{\mathbf{j}} \) denote the intensity and bit value of pixel \( j \), consecutively. A bit string is generated from a 9 × 7 window and stored in 64-bit integer. Using the bit string, we measure the matching cost \( C_{\mathbf{j}}^{\mathbf{i}} \) for each pixel \( i \) and disparity label candidate \( j \). Hirshmüller and Scharstein [9] utilize the
same bit strings length to compute the Census cost. Note that we utilize Matlab SDK\(^1\) in Middlebury Stereo Vision Evaluation Page to compute the data cost. The matching cost is the Hamming distance computed as follows.

\[
C_{ij}^0 = \sum_{j} \begin{cases} 
1 & \text{if } F_{ij}^{M} \neq F_{j-I-I}^{E} \\
0 & \text{otherwise}
\end{cases}
\]

\[(2)\]

After matching cost computation, we perform a weighted cost aggregation, which is described in general terms as follows.

\[
C_{i} = \sum_{j \in N} W_{ij} \cdot C_{ij}^0
\]

\[(3)\]

where \(C_{ij}^0\) is the filtered cost of pixel \(i\) and disparity label candidate \(l\). Each cost from each pixel \(C_{ij}\) is grouped into cost volume slice \(C_l\) based on the label candidate \(l\). Then, each cost volume slice \(C_l\) is grouped into a cost volume \(C\). Most conventional methods \([12,15,31,33]\) require a guidance color image to measure the weight value \(W_{ij}\) for each pixel \(j\) in the support region. They assume that the weight values in the color and disparity patches are similar. However, these conventional methods often fail when the support region having similar disparity shows high color variance or when the guidance image is unreliable because of noise, blur, etc.

Instead of using a guidance color image, we use the input cost volume slice \(C_{ij}^0\) to guided itself, thus yielding a self-guided cost aggregation. We observe that measuring the weight directly from the cost volume slice is difficult because it is very noisy. Fig. 2(b) and (c) show the disparity maps computed from the initial and self-guided filtered cost volume. It demonstrates that a self-guided filtered cost volume generates a similar disparity map to that of the initial noisy cost volume, which is far from accurate. To solve this problem, we employ a deep learning technique to learn the weight and perform weighted cost aggregation simultaneously. The proposed method is called deep self-guided cost aggregation. Fig. 2(d) shows that the proposed method obtains a fine disparity map despite not utilizing a guidance color image. The deep learning networks are described in detail in the following subsections.

To obtain the disparity map, we employ winner-takes-all (WTA) method \((l = \arg \min_{i \in \mathbb{L}} C_{ij})\). Because we focus on local stereo matching, we do not employ any global optimization method. In addition, in order to compare the discrimination power of the proposed with that of the conventional methods, the post processing step normally used to measure the disparity map error is skipped.

3.2. Deep neural network architecture

The proposed deep cost aggregation network consists of two sub-networks: dynamic weight and descending filtering networks. The dynamic weight network is inspired by Brabandere’s dynamic filter network \([1]\). Whereas the original dynamic filter network generates a dynamic filter to be applied to an input, the proposed network generates a set of dynamic weights for each input pixel. These weights help to guide the filtering network. Each pixel from the input cost volume slice is then multiplied by \(S (=32)\) dynamic weight values.

The results are used as input for the subsequent filtering network. The descending filtering network performs an edge-aware filtering operation for the input cost volume slice. Thus, we utilize an edge-aware filtered cost volume slice as the learning target.
An overview of the network architecture is shown in Fig. 3, where \( W \) and \( H \) are the width and height of the input, respectively. In addition, \( d \) and \( r \) are the diameter and radius of the support region, respectively. Based on the proposed network, Eq. (3) is modified to the following:

\[
C_{i,j} = \Phi_2(\Phi_1(C_{i,j}^0) \odot \psi(C_{i,j}^0))
\]

where \( \Phi_1 \) and \( \Phi_2 \) denote the dynamic weight and descending filtering networks, respectively. \( \psi \) is the replication operator for the input cost volume slice. Then, the pixelwise multiplication operator is represented by \( \odot \).

For the dynamic weight network, we set up the first layer with \( S \) filters of \( 1 \times 1 \times 1 \). This means that the input cost volume slice is multiplied by various weights before it is convolved with the spatial filters. If the dynamic weight only has 1 channel, the important weight for a pixel is only one which is very limited. By utilizing various weights, it can give more combination of important pixels as the input for the filtering network. We then use \( D \) convolutional layers in which rectified linear unit (ReLU) layers follow the \( D - 1 \) layers. The convolutional layers consist of \( S \) filters of \( 3 \times 3 \times 5 \). For the \( 19 \times 19 \) support region, we set \( D = 9 \) so that the receptive field size is the same as the support region size. The output of this network are the \( S \) channels of \( W \times H \) weights. To perform pixelwise multiplication of the weights and input cost volume slice, a replication process for the input must be conducted beforehand so that both have the same number of channels.

For the descending filtering network, the initial layer is a convolutional layer that consists of \( S \) filters of size \( 1 \times 1 \times S \). Furthermore, we employ \( F \) convolutional blocks that each consists of 2 convolutional layers. Both convolutional layers have \( S \) number of 2D filters with different directions (horizontal and vertical), as shown in Fig. 3. The ReLU layers follow all convolutional layers in the convolutional blocks. The size of the filter in each convolutional block decreases with consecutive convolutional blocks. For the \( 19 \times 19 \) support region, the first convolutional block consists of \( S \) filters of \( 1 \times 19 \times S \) and \( 19 \times 1 \times S \) size. The size of the filters in the second and subsequent convolutional blocks are \( 1 \times 15 \times S, 15 \times 1 \times S, 11 \times 1 \times S, 11 \times 1 \times S \), and so on. The filter size is reduced until the last convolutional layer becomes a filter of \( 1 \times 1 \times S \) size.
To preserve the edges more effectively, the network uses a cost volume slice as the input and predicts an edge-aware filtered cost volume slice. To preserve the edges to an even greater extent, we adapt the feature reconstruction loss function by Johnson et al. [14] to our approach. We utilize the feature maps from the first 4 of the 16 layers VGG network [26] pretrained on ImageNet [22]. The early layers represent low-level features that can preserve the texture and shape of the output cost volume. We also integrate the pixelwise mean square loss function with the feature reconstruction loss function.

Given $M$ training samples $[C^T_i, c^T_i, \lambda_i^T]$, the network parameters are trained by minimizing the following loss function.

$$\text{Loss} = \|C^T_i - C_i\|^2 + \sum_{\nu} \lambda_{\nu}\|\phi_{\nu}(C^T_i) - \phi_{\nu}(C_i)\|^2$$

where $C^T_i$ and $\phi$ denote the target cost volume slice and feature reconstruction operator, respectively. $\lambda_\nu$ is the importance weight for feature loss computed on the $\nu$ layer. The loss function is normalized by the number of pixels. While pixelwise mean square loss takes care of the overall value of output cost volume slice, the feature reconstruction loss is useful to preserve the edge information in the cost volume slice. With the deep features-based loss, we put more weight on loss at the edge region. Thus, we can force the network to perform edge-aware cost aggregation.

### 3.3. Dataset generation

Because of the absence of a ground truth cost volume slice, applying a deep learning approach during the cost aggregation step is difficult. Thus, we utilize a guided filtered cost volume [12] as the target. However, instead of using the input color image for guidance, we employ the ground truth disparity map. Fig. 4 shows a comparison of the disparity maps from the filtered cost volume using both guidance images. The latter one performs cost aggregation more effectively because it generates weights based on the ground truth disparity map. Note that this is consistent with the initial intention of the proposed self-guided cost aggregation, which is to learn the weight from a cost volume slice. The comparison confirms that the cost volume slice is more suitable for using weights in the ground truth disparity map than in the color image.

Note that the proposed deep network is an end-to-end learning system. In this study, we randomly extract 1000 patches from each training image. To train the edge-preserving characteristic, 80% of the patches are those around the depth discontinuities. We select only the patches of the cost volume slice on the ground truth disparity label that have high reliability. Note that patch selection is important and strongly related to deep learning performance.

Fig. 5 provides examples of the training dataset. We observed that the input cost volume slice is noisy, whereas the target cost volume slice is edge-aware filtered. When a simple mean filter is applied to the input patch, the filtering operator smooth the edge information, as less information is available for filtering. By contrast, the deep self-guided filter achieves better results even when a lack of information exists. We confirmed that the proposed deep learning approach can perform self-guided edge-aware filtering.

To increase the data variety, we employ two data augmentation methods. First, we add rotation variance by rotating each input and target cost volume slice through three angles $\theta$ (90°, 180°, 270°). Then, a contrast variance is considered by applying a power transformation with two $\gamma$ values ($= 0.67, 1.5$). While the purpose of rotation augmentation is clear, the intention of contrast variance is ambiguous. Note that the input data is not the stereo image pair but the cost volume slice. Thus, the contrast variance is addressed to deal with the matching cost difference between low and high costs. Fig. 6 shows the illustration of the data augmentation.

### 4. Experimental results

The proposed algorithm is implemented on an Intel i7 4770 @ 3.4GHz with 16GB RAM. We use MatConvNet deep learning toolbox [28] to perform the deep cost aggregation method. We use the Middlebury dataset [23,24] and KITTI dataset [16,17] in order to validate the performance of the proposed cost aggregation method. We collect 40 images from 2005, 2006, and 2014 Middlebury datasets to train the network parameters. We then test the learned parameters on 15 training and 15 test images from the Middlebury evaluation (version 3) dataset and 200 training images and 200 test images from the 2015 KITTI dataset.

In our experiment, the patch sizes used for input and target cost volume slices are 35 × 35 and 17 × 17, respectively. The parameters for the guided filtering in the dataset generation are fixed at $r = 9$ and $\epsilon = 0.0001$. We train the deep network using stochastic gradient descent to minimize the loss function. We train for 30 epochs and the learning rate is 0.00001. The batch size is 64. From the 40 training images, we extract 480,000 examples in which 20% of them comprise the validation set. With the Adiornack data, the training process requires approximately 4 hours and the proposed cost aggregation method consumes 2.86 seconds on a PC with NVIDIA GeForce GTX 1080.

To evaluate the performance of the deep self-guided cost aggregation method (DEEP), we compare it to the state-of-the-art methods such as GF [12], NL [31], ST [15], and DF [20]. We also compare the results with the simple 7 × 7 mean filter (BOX). Note that BOX is similar to the proposed method in that it does not use a guidance image. Where possible, we utilize the code implemented by the authors in [35] for comparison. We implement the code of domain transform based cost aggregation (DF) which is not available. For the parameter settings, we use the identical value as in the original papers. For GF [12], $r$ and $\epsilon$ are set to 9 and 0.0001, respectively. $\sigma$ is set to 0.1 for both NL [31] and ST [15]. The constant parameter $k$ in ST [15] is set to 1200. For DF [20], we apply $\sigma_t = 25 \text{ and } \sigma_t = 0.1$. We do not perform a post processing method
Table 1

Bad pixel percentage comparison (Middlebury training data).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>BOX</th>
<th>GF</th>
<th>NL</th>
<th>ST</th>
<th>DF</th>
<th>DEEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adirondack</td>
<td>16.32</td>
<td>11.05</td>
<td>18.05</td>
<td>17.58</td>
<td>10.53</td>
<td><strong>8.88</strong></td>
</tr>
<tr>
<td>Art</td>
<td>18.10</td>
<td>14.18</td>
<td>16.51</td>
<td>16.31</td>
<td>16.62</td>
<td><strong>13.79</strong></td>
</tr>
<tr>
<td>Motorcycle</td>
<td>11.51</td>
<td>10.07</td>
<td>13.34</td>
<td>14.73</td>
<td>13.38</td>
<td><strong>8.66</strong></td>
</tr>
<tr>
<td>MotorcycleE</td>
<td>10.50</td>
<td>9.35</td>
<td>12.90</td>
<td>13.85</td>
<td>11.89</td>
<td><strong>7.78</strong></td>
</tr>
<tr>
<td>Piano</td>
<td>23.89</td>
<td>18.80</td>
<td>24.71</td>
<td>21.85</td>
<td>18.23</td>
<td><strong>17.55</strong></td>
</tr>
<tr>
<td>Pianol</td>
<td>41.36</td>
<td>32.20</td>
<td>35.63</td>
<td>34.17</td>
<td>38.40</td>
<td><strong>31.41</strong></td>
</tr>
<tr>
<td>Pipes</td>
<td>15.54</td>
<td>13.71</td>
<td>14.74</td>
<td>13.95</td>
<td>16.35</td>
<td><strong>12.38</strong></td>
</tr>
<tr>
<td>Playroom</td>
<td>29.47</td>
<td><strong>21.93</strong></td>
<td>25.00</td>
<td>22.17</td>
<td>25.57</td>
<td>23.98</td>
</tr>
<tr>
<td>Playtable</td>
<td>38.88</td>
<td>40.79</td>
<td>47.87</td>
<td>45.83</td>
<td>39.75</td>
<td><strong>36.67</strong></td>
</tr>
<tr>
<td>PlaytableP</td>
<td>23.88</td>
<td>27.68</td>
<td>31.75</td>
<td>30.42</td>
<td>25.88</td>
<td><strong>19.91</strong></td>
</tr>
<tr>
<td>Recycle</td>
<td>19.42</td>
<td>13.62</td>
<td>16.17</td>
<td>17.07</td>
<td>13.00</td>
<td><strong>11.44</strong></td>
</tr>
<tr>
<td>Shelves</td>
<td>45.63</td>
<td>40.89</td>
<td>39.76</td>
<td><strong>39.42</strong></td>
<td>42.55</td>
<td>41.13</td>
</tr>
<tr>
<td>Teddy</td>
<td>10.69</td>
<td>8.15</td>
<td><strong>7.43</strong></td>
<td>7.50</td>
<td>10.44</td>
<td>8.24</td>
</tr>
<tr>
<td>Vintage</td>
<td>39.49</td>
<td>36.54</td>
<td>55.28</td>
<td>48.27</td>
<td>37.30</td>
<td><strong>33.53</strong></td>
</tr>
<tr>
<td>Average</td>
<td>21.90</td>
<td>18.86</td>
<td>22.87</td>
<td>21.79</td>
<td>20.13</td>
<td><strong>17.06</strong></td>
</tr>
</tbody>
</table>

Fig. 11. 2015 KITTI test dataset. (a) Input left image; (b) Disparity maps of the proposed method (DEEP); (c) Stereo errors of (b); (d) Disparity maps of CF [12]; (e) Stereo errors of (d); (f) Disparity maps of NL [31]; (g) Stereo errors of (f). Blue and red colors denote the correct and wrong estimates, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1 lists the bad pixel percentages for each item of Middlebury training data and the weighted average of bad pixel percentages for all datasets, as conducted by the Middlebury evaluation SDK. The bad pixel percentage is calculated for only the non-occluded region with an error threshold of 1. The table shows that the proposed method achieves the smallest mean error of all the compared cost aggregation methods. Note that our method does not employ a guidance color image, which represents a major advantage over the other methods. Fig. 7 provides a qualitative comparison of the selected data (Adirondack, Motorcycle, Piano, Pipes).

We also compare the performances of the methods with and without data augmentation for the same test set. The weighted average of bad pixel percentages of the proposed method with the data augmentation is 17.06%; without data augmentation, it is 17.69%. We confirm that data augmentation assists the deep network in learning the parameters. Then, analysis to the number of the dynamic value $S$ is performed. We measure the average bad pixel percentages of Middlebury training data for various $S$ values as shown in Fig. 8. It is validated that $S = 32$ obtains the lowest bad pixel percentage among others.

We also test the trained parameters on the Middlebury test dataset by submitting on the benchmarking website. Note that we could not compare with other cost aggregation algorithms due to its unavailability on the evaluation site. Table 2 shows the bad pixel percentage for each data in Middlebury test dataset with the error threshold 4. The weighted average bad pixel percentage for the proposed method is 18%. Fig. 9 shows the disparity maps of selected Middlebury test dataset.

Table 3 shows the comparison of average bad pixel percentage of KITTI training dataset for non-occluded and whole regions, consecutively. Same evaluation metric with KITTI benchmark is uti-
lized with an error threshold is 3. It shows that the proposed method also achieves the smallest average error in KITTI training dataset. The disparity maps generated by our approach are depicted in Fig. 10.

We also perform the evaluation on KITTI test dataset by submitting the results to KITTI benchmark (DSGCA in the website). Note that we focus on local cost aggregation only so that we do not employ any global optimization nor post-processing steps. Despite that, it still overcomes the guided filter based cost aggregation (GF) [12] and non-local cost aggregation (NL) [31]. Note that we only compare with the algorithms with their results are available on the KITTI benchmark. Table 4 shows that the proposed method obtains the lowest average error among those methods. The disparity comparison is shown in Fig. 11. We believe that the performance can be improved by applying state-of-the-art global optimization or post-processing methods.

5. Conclusion

We proposed a novel deep convolutional network for use in conducting self-guided cost aggregation for stereo matching. Unlike conventional approaches, the proposed method did not employ a guidance color image. Instead, we used a deep learning approach to run a self-guided cost aggregation. Our deep network consisted of two sub-networks: dynamic weight and descending filtering networks. The first was employed to learn the importance weight for each pixel in the input, whereas the second was used to perform edge-aware filtering. In our study, a set of training patches were randomly generated from a set of training images. Although there was no ground truth cost volume slice, we employed a filtered cost volume slice with ground truth disparity map for guidance. Our experiments revealed that the proposed method achieved the best results when compared to state-of-the-art techniques.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2016R1A2B4014731).

References


