Using Temporal Correlation to Optimize Stereo Matching in Video Sequences

Ming LI†, Li SHI†, Xudong CHEN†, Nonmembers, Sidan DU[α], Member, and Yang LI[β], Nonmember

SUMMARY The large computational complexity makes stereo matching a big challenge in real-time application scenario. The problem of stereo matching in a video sequence is slightly different with that in a still image because there exists temporal correlation among video frames. However, no existing method considered temporal consistency of disparity for algorithm acceleration. In this work, we proposed a scheme called the dynamic disparity range (DDR) to optimize matching cost calculation and cost aggregation steps by narrowing disparity searching range, and a scheme called temporal cost aggregation path to optimize the cost aggregation step. Based on the schemes, we proposed the DDR-SGM and the DDR-MCCNN algorithms for the stereo matching in video sequences. Evaluation results showed that the proposed algorithms significantly reduced the computational complexity with only very slight loss of accuracy. We proved that the proposed optimizations for the stereo matching are effective and the temporal consistency in stereo video is highly useful for either improving accuracy or reducing computational complexity.

key words: stereo matching, disparity, computer vision, temporal correlation, convolutional neural network

1. Introduction

Stereo matching is a fundamental problem of computer vision and plays an important role in many applications, such as the street view reconstruction, the body position detection and the advanced driver assistance systems (ADAS) [1]. There is a huge demand to efficiently generate disparity on various hardware architectures for real-time applications such as ADAS, where the computational complexity of algorithm is crucial.

State-of-the-art stereo matching algorithms involves energy minimization methods such as the belief propagation [2], the graph cuts [3] and the semi-global matching (SGM) [4], which significantly improved the accuracy of disparity estimation. More recently, the convolutional neural networks (CNN) [5]–[8] based methods made further progress. However, these algorithms generally considered more about the accuracy of disparity instead of the computational complexity, getting more and more time consuming. MCCNN [9] combines conventional stereo matching method and the convolutional neural network. MCCNN follows the general steps of traditional methods such as SGM, but it uses CNN to calculate the matching cost. MCCNN algorithm uses several convolutional layers to extract features of image patches, and then uses a series of full connected layers to calculate the similar score from the feature vectors. According to the Middlebury benchmark [10] and the KITTI benchmark [11], most matching algorithms proposed in recent three years are still far from real-time implementation even running on the most powerful hardware such as NVidia GTX Titan X GPU [12] or multi-core Intel Xeon CPU.

Among algorithms that focused on real-time implementation, most of them were developed based on the SGM, because the SGM and its variants make a good balance between the accuracy and the computational complexity [13]–[14]. The SGM algorithm uses dynamic programming to integrates the matching cost along multiple 1-D energy paths to approximate a global energy regularized cost function. The energy function consists of the cost value and two penalty terms for slanted surfaces and discontinuities respectively. It has been implemented on various hardware architectures such as CPUs, GPUs and FPGAs, with different strategies and modification for speeding-up [3], [15]–[19]. However, these methods are still facing great challenge in real-time implementation, especially when higher resolution and larger disparity range applied.

In the recent years, temporal consistency between consecutive frames has been getting attention in the stereo matching problem. Many techniques have developed to achieve better accurate performance of stereo matching [20]–[23] using either spatial-temporal window or motion flow. For example, [24] presented a temporal dual-cross-bilateral grid algorithm to reduce flickering and noise. Researches in [25]–[27] extended the guided image filter to the spatio-temporal domain to improve accuracy. [28] defined scene flow and recovered by coupling the optical flow estimation. [29] proposed a weighted mode filtering method and extended it to temporal domain. However, all these algorithms aim to generate more accurate disparity utilizing spatial-temporal consistency and algorithm computational complexity was significantly increased. To our best knowledge, in the only work [1] that utilizes temporal correlation for algorithm speeding-up, Jiaafari et al. proposed a stereo matching method for the ADAS that uses temporal consistency to reduce the disparity range. However, this method only calculates the disparities at the edge pixels instead of generates an entire disparity map. The method is unable to identify those pixels that which disparity range was incorrectly predicted, causing significant accuracy loss.
In this work, we aim to optimize the real-time performance of stereo matching algorithm, using temporal consistency between consecutive frames. We propose the dynamic disparity range (DDR) scheme and the temporal path aggregation scheme for stereo matching. Based on the schemes, we proposed the DDR-SGM and the DDR-MCCNN algorithms for the stereo matching in video sequences. The remainder of this work is organized as follows. In Sect. 2, the detail of dynamic disparity range (DDR) and the temporal path aggregation are presented. The DDR-SGM and DDR-MCCNN are proposed in Sects. 3 and 4 respectively. Evaluation results of the proposed algorithms along with comparisons will be given in Sect. 5. In Sect. 6, Parallel implementation is discussed and a conclusion is presented.

2. Optimizing the Matching Cost Computation and the Cost Aggregation Using Temporal Correlation

Most stereo matching algorithms consist of three steps: (1) Matching cost calculation. (2) Cost aggregation. (3) Disparity optimization and disparity refinement. For each pixel from the input image pairs, the conventional stereo matching algorithms calculate the matching cost and aggregation energy value for every possible disparity value. The total operations of the algorithm can be simply expressed as $W \times H \times D$, where $W$, $H$ are the width and height of the input image respectively and $D$ is the range of the disparity, which must cover all the possible disparities of ground truth.

In a video sequence that content does not change suddenly, either RGB images or disparity maps from consecutive frames are highly temporally correlated. This basic idea is the fundament of all the video compression algorithms. Similarly, the disparity map can be well predicted from previous frames. Therefore, the temporal consistency of disparity maps could be highly useful for either improving accuracy or reducing computational complexity in a stereo matching problem. However, all known efforts of algorithm speeding-up ignored the temporal correlation.

2.1 The Dynamic Disparity Range

Because the disparity at a certain pixel is temporally correlated with disparities at this pixel in other frames, we can simply predict the disparity from the previous values:

$$\begin{cases} 
\text{Disp}_{k-1}(u,v) = d \\
\text{Disp}_k(u,v) = d + \Delta d 
\end{cases}$$

where $(u,v)$ represents the coordinates of a pixel and $\text{Disp}_{k-1}$ represents the disparity map of the $(k-1)$th frame. Therefore, we can predict the disparity $d$ of pixel $(u,v)$ in the $k$th frame by adding a small value $\Delta d$ to the disparity value of the same pixel in the $(k-1)$th frame.

The framework of proposed optimization for the stereo matching was shown as Fig. 1. Optimizations were made to both the cost calculation step and the cost aggregation step. We assumed that true disparity value in current frame was totally covered by a relatively narrow disparity range with width of $R = 2\Delta d$. The center line of the range was determined by the disparity value in the previous frame. To reduce the computational complexity. Matching cost and the energy aggregation could be only calculated within this disparity range, instead of a fixed full range. Ideally, the total operations could be greatly reduced to $W \times H \times R$, if the width of the disparity range $R$ is much smaller than full range $D$.

To accelerate the stereo matching algorithm, the cost calculation and aggregation should be performed only in a small neighbor range which covers the true disparities as many as possible. We thereafter defined a searching neighbor range that is centered by disparity map $\text{Disp}_{k-1}$ in previous frame with a width of $R$ and we named it the dynamic disparity range (DDR).

We first study a simple version of the problem in a 1-D situation. Considering one row in the image, the disparities from the row in the previous and the current frame were shown in Fig. 2. Due to the correlation between consecutive frames, disparities in current frame (solid line in Fig. 2) can be well predicted from the previous disparities (dot line in Fig. 2). The DDR was shown as the area between the dash lines that centered by previous disparities. Ideally, the
proportion of the area between the DDR and the full range (the area between maxD and minD) can be roughly treated as the improvement of acceleration. However, the true disparities at some points may fall out of the DDR, as shown in the light grey area in Fig. 2, especially at edges of moving objects in the image that disparity changes rapidly. To avoid calculating disparity incorrectly in the area where true value falls out of the DDR, the searching range must be extended to the full range. Therefore, identifying those pixels which disparities rapidly changes and falls out of the DDR is an essential problem.

We set a threshold (TH) to determine the confidence of the disparity and estimate whether the true disparity falls out of DDR. As shown in algorithm 1, if min(Lr(p,d)) > TH, the disparity is not included in the DDR, which means the true disparity is not included in the DDR, the DDR should be extended to the full range of disparity and re-calculate the matching cost and aggregation cost. Finally, a winner-takes-all strategy was used to optimize the disparity. Because the disparity refinement and the L-R check is independent to the matching algorithm, they will not be discussed in this work.

**Algorithm 1 DDR algorithm procedure**

```plaintext
//min_D: minimal value of full disparity range
//max_D: maximum value of full disparity range
//d_pre: the disparity in previous frame
//min_r: minimal value of dynamic range
//max_r: maximum value of dynamic range
//generate dynamic range
min_r = max(min_D, d_pre - range/2);
max_r = min(max_D, d_pre + range/2);
//avoid stepping across boundary
for d = min_r, d < max_r, d++
calculate matching cost
for r of all paths
   calculate aggregation cost Lr
end for
//calculate in dynamic searching range
if min(Lr) > TH
if larger than threshold, extend to the full range
for d = min_D, d < max_D, d++
   if d >= min_r & d <= max_r
      continue;
   else
      calculate matching cost in full range
      for r of all paths
         calculate aggregation cost Lr
      end for
   end if
end for
end if
```

Cost Aggregation Combined with Temporal Path

In addition, to optimize the cost aggregation, we proposed a new temporal path to the conventional lateral energy aggregation, which iterates along time axis. The cost aggregation was calculated by summing the cost value along multiple paths including the temporal path. The temporal path aggregation was calculated according to the path value of the frame k − 1, within the disparity range R. Original SGM algorithm calculates the cost aggregation along several paths in different directions. These paths traverses in the same frame, the lateral plane. Thereafter they were called the lateral paths. Generally, the number of paths are configurable according to where the balance point was set between accuracy and real-time performance. Typical settings used 3, 4 and 8 paths.

\[
L^k_i(p,d) = C^k_i(p,d) + \min_i L^{k-1}_i(p,d) + Q_1, L^{k-1}_i(p,d+1) + Q_2, \min_j L^{k-1}_j(p,i) + Q_2) - \min_j L^{k-1}_j(p,j)
\]

(2)

Because of the correlation among frames in a continuous video sequence, some occluded pixels may reveal from previous frames. In addition, we may also improve the accuracy of disparity in this manner. Therefore, we proposed a new method that combine the temporal aggregation path into the cost aggregation procedure of the original SGM. As shown in Fig. 3, the temporal aggregation path is the iterative process from the first frame which traverses in the direction of time. The algorithm of temporal path aggregation is similar to those lateral paths as shown in Eq. (2), while p refers to one pixel in the image. \(L^1_t\) refers to the aggregation cost of temporal path in frame k and \(L^{k-1}_t\) refers to the aggregation cost in frame k − 1. \(Q_1\) and \(Q_2\) are the penalty terms similar with the \(P_1\) and \(P_2\) in the original aggregation step, but penalty terms may have different values for lateral and temporal paths.

The sum cost with temporal path can be represented as Eq. (3).

\[
S(p,d) = \sum_i L^k_i(p,d) + w \cdot L^t_k(p,d)
\]

(3)

While \(L_t\) represents the lateral path and \(L^t\) represents the temporal path. Considering different frame rate of the video sequences, a coefficient w is set to adjust the weight of temporal path.
3. The DDR-SGM

Because the SGM [2] and its various variants gain a good balance between the accuracy and computational complexity, they are widely used in real-time application scenarios.

In procedure of SGM algorithm, cost aggregation is expressed as Eq. (4). where \( L_r(p,d) \) is the aggregation cost value along the path \( r \) for pixel \( p \) and disparity \( d \). It equals to the cost value \( C(p,d) \) plus the minimum path cost of the previous pixel \( p - r \) with \( P_1 \) and \( P_2 \), the penalty terms of the smoothness term. In general, \( P_1 \) has to be smaller than \( P_2 \). \( P_1 \) penalizes inclined surfaces and \( P_2 \) penalizes discontinuous points.

\[
L_r(p,d) = C(p,d) + \min\left(L_r(p - r,d), L_r(p - r,d - 1) + P_1, \min_i L_r(p - r,i) + P_2\right) - \min_j L_r(p - r,j)]
\]

To calculate the disparity at the pixel \( p \), the aggregation costs along every path at every disparity within the searching range will be calculated. The true disparity of \( p \) is the disparity that minimize the sum of all aggregation costs along every path. Consequently, the larger the aggregation cost value at a certain pixel is, the less confidence level of the disparity has. Therefore, we could identify those pixels where disparities have rapid changes and fall out of DDR by simply checking the aggregation costs without bringing any extra computation.

The proposed DDR-SGM method follows the pseudocode in algorithm 1. Matching cost and aggregation cost will only be calculated within the DDR. For one aggregation path \( r \), every disparity \( (d) \) in the range has an aggregation cost \( L_r(p,d) \). A threshold \( (TH) \) was set to determine the confidence of the disparity and estimate whether the true disparity falls out of DDR. If \( \min_i (L_r(p,i)) > TH, i \in DDR \), the DDR should be extended to the full range of disparity and re-calculate the matching cost and aggregation cost.

As Fig. 4 shows, the disparity-row map was generated by calculating disparities of the same row in two contiguous frames from the Tanks dataset built by Christian Richardt et al. [24]. The x-axis of the map is the x coordinate of pixels in the row, and the y-axis is the disparity. The full disparity range is from 0 to 63. In this map, the green dot line respects the disparities of pixels in the row in previous frame, while the red line shows the disparities in current frame. The summary aggregation cost of each pixel is shown as mapped color in Fig. 4, and the lower grey lever refers to the smaller cost value. The blank area between maxD and minD is the uncalculated pixel-disparity combinations. By this mean, the proposed DDR avoids the calculation for this proportion to reduce the computational complexity.

As shown in Fig. 4, red line is very close to the green one, which means the disparities at most pixels are similar with previous frame. The disparity searching range that was generated by the previous frame well predict the disparity in the current frame. In the areas that disparity changes rapidly, the searching range is automatically extended to the full range (the rectangle area that extended to the full range). As shown in figure, only few pixels need to extend the searching range.

4. The DDR-MCCNN

MCCNN is a stereo matching method that combines convolutional neural network and conventional cost aggregation. The procedure of MCCNN is similar to SGM and consists of three steps: matching cost calculation, cost aggregation, disparity optimization and refinement. MCCNN uses a convolutional neural network to extract feature vectors and a multiple layer of full connect (FC) network to calculate the matching cost (the FC-cost). To smooth the disparity map, the original MCCNN method uses semi-global or CBCA aggregation methods in the cost aggregation step. As the results that Zbontar et al. [9] shows, the MCCNN achieved very good performance of the matching accuracy. However, this method has high computational complexity and is time-consuming.

Here we propose an optimized method called DDR-MCCNN which uses dynamic disparity range to accelerate the matching cost calculation and aggregation. DDR-MCCNN calculates and aggregates the matching cost only within the dynamic disparity range instead of the full range, which greatly reduce computational complexity. Similarly, we use Algorithm 1 to determine whether the pixel needs to extend to the full range and re-calculate the cost.

In most implementations of MCCNN, CNN based feature extraction and FC based matching cost calculation were coded with CUDA in a highly parallel manner. Usually the FC network consumes much more computational time than CNN. Because FC-cost is implemented in parallel, calculating the FC-cost in a single pixel is much more time consuming than in a batch of pixels. Therefore, it’s more reasonable to calculate the matching cost for those pixels extended to full disparity range with a time-efficient cost operator. In the proposed DDR-MCCNN, we calculate the dot prod-
uct of two feature vectors (DP-cost) instead of FC-cost for range extended pixels. As a compromise, DP-cost is much more time-efficient than FC-cost suffering slight loss of accuracy. In order to integrate the two different types of cost in aggregation step, FC-cost and dot-product cost should be normalized.

We plot the joint distribution of FC-cost and DP-cost in Fig. 5. We use cumulative distribution function (CDF) to normalize the two types of matching cost. As shown in Eq. (3), $N_k$ is the normalized distribution, and $P_r(x)$ is the CDF of input cost.

$$N_k = \int P_r(x)dx$$ (5)

Therefore, the optimized DDR-MCCNN, as shown in Fig. 6, consists of the following steps: (1) calculate the feature vectors of the patches through the CNN; (2) calculate the FC-cost in the dynamic range, and calculate the DP-cost in the full disparity range; (3) cost aggregation, use normalized DP-cost if need to extend the range; (4) generate the disparity map and refinement.

5. Evaluation

5.1 Experimental Condition

Even though there are many variants of the SGM, most of them differs from each other by different strategies of matching cost calculation. Our methods of the DDR and temporal cost aggregation should be effective for all SGM based algorithms. Therefore, we chose the census transform in the matching cost calculation step in evaluation of DDR-SGM, which is proved to have better performance than SSD and BT, to calculate the matching cost. At the cost aggregation stage, our evaluation followed the standard procedure. And in the evaluation of DDR-MCCNN, we calculate the FC-cost in the dynamic disparity range and use DP-cost for range extended pixels.

To speed up the cost computation, we used a two-step strategy which is accepted in many implementations of SGM, to calculate the census transform: The L and R codes calculation step, which compares the center point with its neighbor to generate a logic vector, and the exclusive-or operation step, which performs the exclusive-or operation between the two L and R codes. This two-step strategy significantly reduced the computational complexity of the census transform and is easy to be parallelized. Similarly, other matching cost calculation operations could be optimized by this mean.

Most research papers in this field use two popular datasets, the Middlebury and the KITTI datasets. The Middlebury dataset does not contain continuous image pairs. The KITTI dataset contains image pairs from video sequences, which have large frame interval up to 5 seconds and is not suitable to evaluate our algorithms. The dataset built by Christian Richardt et al. [24] (400 x 300 pixels) and the scene-flow dataset created by Nikolaus Mayer et al. [30] (960 x 540 pixels) are only two published stereo video datasets with ground truth labeled. We evaluated the proposed DDR-SGM and DDR-MCCNN on both two datasets. The evaluation programs for DDR-SGM were coded with C++ and ran on an Intel i7 6700K CPU with single thread. While the evaluation programs for DDR-MCCNN were coded with Lua, C++ and CUDA, and ran on an AMD Ryzen Threadripper 1950X CPU and a NVidia 1080Ti GPU. And the convolutional and full-connect networks are implemented by Torch. Due to the lack of research on stereo matching in video sequence and limited datasets that marked with ground truth, we used another dataset in the temporal cost validation experiment, the Middlebury Stereo Dataset. A window slides over the image to simulate the camera panning in a video sequence, as shown in Fig. 11 (a).
5.2 Evaluation of DDR-SGM

We compared the error rates and the computational time for each video sequence in the datasets. The frames from the videos, the disparity maps and the DDR masks for DDR-SGM on Richardt’s dataset [24] were shown in Fig. 7. And the results on the scene-flow dataset [30] were shown in Fig. 8. The white pixels in the DDR masks are those pixels whose disparity range were extended to the full range. The black pixels are those which only calculate matching costs and aggregation in much narrower range. As shown in Fig. 7, only a small proportion of pixels only few of pixels need to be extended the disparity range and to re-calculate the cost and aggregation value. Results show that disparity in current frame can be well predicted by previous frames. In general, the most of incorrectly predicted pixels are located edge of objects, where the disparity changes sharply.
Table 1  Average error rate of DDR-SGM and SGM

| Datasets          | Bad-4 average error rate(%) | Bad-2 average error rate(%) |
|-------------------|------------------------------|----------------------------|
|                   | DDR-SGM | SGM | DDR-SGM | SGM |
| Book              | 3.5     | 3.5 | 4.0     | 4.0 |
| Street            | 3.8     | 3.7 | 4.6     | 4.4 |
| Tanks             | 4.2     | 4.2 | 4.9     | 5.0 |
| Temple            | 5.9     | 5.6 | 7.2     | 7.0 |
| Tunnel            | 2.8     | 2.7 | 2.8     | 2.8 |
| All               | 4.0     | 4.0 | 4.7     | 4.6 |
| A rain of stones  | 3.5     | 2.7 | 15.2    | 14.4 |
| Flower storm      | 8.8     | 7.3 | 14.7    | 13.4 |
| Lone tree         | 8.4     | 8.3 | 13.0    | 13.0 |
| All               | 6.9     | 6.1 | 14.3    | 13.6 |

To quantify the accuracy of the algorithms, the bad-pixel-2 (b-2) and bad-pixel-4 (b-4) error rates were calculated. For the real-time application scenario, the b-4 and b-2 index are more commonly accepted. As shown in Figs. 7 and 8, the error rate curves of DDR-SGM and SGM are very close, while the computational time curves of DDR-SGM are much lower than those of SGM. On the Book data, DDR-SGM and SGM achieved the same b-2 of 3.5% and b-4 rate of 4.0% on average respectively (see Table 1). On the Tunnel and Lone tree data, DDR-SGM suffered very slight loss of accuracy under b-2, but achieved the same b-4 error rate. And on the other data the error rates of DDR-SGM are a little bit higher than those of SGM. However, compared with the total error rate, the loss is very slight. The reason is that the threshold $TH$ did not identify all the incorrectly predicted pixels, causing error of disparity. But a proper $TH$ value can make the loss considerably slight. Interestingly, on the Tanks data, DDR-SGM achieved higher accuracy than SGM under b-4 (4.9% and 5.0% respectively). This is probably because the smaller DDR range that generated by the previous frame filtered some noise of matching cost out. Comparing with the error rates of DDR-SGM and SGM on different datasets, the average loss of the accuracy on dataset [30] is higher than that on Richardt's dataset [24]. This is probably caused by the lack of texture in the scene-flow dataset [30]. However, the loss of the accuracy on scene-flow dataset is also acceptable according to the results in Fig. 8 and Table 1.

The computational time of DDR-SGM and SGM for each frame were also shown in Figs. 7 and 8. In our evaluation on dataset [24], the width of DDR was set to 10, while the full disparity range was 64 for conventional SGM. The full range was set to 160 and the DDR width was 32 on scene-flow dataset [30]. As Figs. 7 and 8 showed, the computational time of DDR-SGM is at the same level with that of SGM at the first frame. The computational time of DDR-SGM at the first frame is even higher than that of SGM on some sequences. This is because there is no prior knowledge for the first frame to generate the DDR. And a large number of pixels need to extend the disparity range and re-calculate, which caused the increase of computational time.

However, from the second frame, the computational time of DDR-SGM dropped rapidly. The DDR mask maps in Fig. 7 show only few of pixels need to be extended the
disparity range and to re-calculate the cost. As shown in Table 2, the computational time significantly benefits from DDR-SGM method. The improvement in Table 2 represents the proportion of time saved by DDR. On Richardt’s dataset [24], DDR-SGM reduced the average computational time from 0.377 second per frame to 0.191 second per frame. The computational time on scene-flow dataset [30] is larger than that on [24] because of the higher resolution of images and larger disparity range. On average SGM takes 5.282 seconds to calculate a disparity map on dataset [30], while DDR-SGM only takes 2.624 seconds. Results show that DDR-SGM significantly reduced the computational time by narrowing the disparity searching range, for all test data.

Taken together, our evaluation results indicate that DDR-SGM can significantly reduce the computational time for all the SGM based algorithms, while suffering only very slight loss of accuracy. On average, DDR-SGM consumes only 50.7% computational time of SGM to generate a disparity map in a video sequence on Richardt’s dataset [24], suffering only less than 0.1% loss of the accuracy. And on scene-flow dataset [30], DDR-SGM costs 49.7% time of SGM with 0.8% loss of the accuracy.

5.3 Evaluation of DDR-MCCNN

We use several video sequences in the dataset [24] and [30] to evaluate our optimization of MCCNN and show the results in Tables 3, 4, Figs. 9 and 10. We also calculate the bad-pixel-2 (b-2) and bad-pixel-4 (b-4) error rates to evaluate the accuracy of two methods. As shown in Figs. 9 and 10, the error rates curves of DDR-MCCNN are very close to those of MCCNN. The error rate of DDR-MCCNN is only about 1.2% higher than MCCNN on the scene-flow dataset [30], and 0.3% on Richardt’s dataset [24]. Since the DP-cost used in the extended area can’t achieve the performance as the FC-cost, this loss of accuracy is acceptable. The computational times of DDR-MCCNN and MCCNN are shown in the Table 4 and Figs. 9, 10. On average, the DDR-MCCNN takes 2.67 seconds per frame while the MCCNN costs 3.82 seconds for each frame on Richardt’s dataset [24], with 30.1% computational time saved. And the DDR-MCCNN takes less than 75% time cost of MCCNN on the scene-flow dataset [30]. The computational time of DDR-MCCNN on high resolution images is 22.7 seconds per frame, while the MCCNN costs 30.4 seconds. According to the computational time curves in Figs. 9 and 10, the time to calculate each frame is basically same.

Because our implementation of the MCCNN was coded with CUDA and well optimized for the parallel implementation, computational time reduction is not as much
as that in DDR-SGM. Due to the slightly differences of the implementations, the DDR-MCCNN has a larger error rate increasing than the DDR-SGM. In the implementation of the MCCNN, FC-cost (Full Connected cost) were calculated through an FCN (full connected network) in a batch manner. The proposed DDR needs to re-calculate the cost value for those extended pixels. However, calculating the FC-cost for a single pair of patches is extremely inefficient. As a compromise, DP-cost (Dot product cost) was proposed and it was much more time-efficient than FC-cost. But this scheme suffered slight loss of accuracy.

There is no prior knowledge for the first frame to generate the DDR. So a large number of pixels will be re-calculated within the full disparity range. The re-calculating step in DDR-MCCNN used the DP-cost instead of the FC-cost. Because calculating of the DP-cost is much faster than the FC-cost, the increasing of computational time in DDR-MCCNN isn’t as much as in DDR-SGM at the first frame. The proportion of extended pixels has a greater influence to the computational time in DDR-SGM than in DDR-MCCNN.

According to Figs. 7, 8, 9 and 10, the disparity maps
calculated by DDR-SGM are smaller than those calculated by DDR-MCCNN with several columns missing on the left. This is because the difference between SGM and MCCNN. The basis of stereo matching is to find every pixel in one image a matching pixel in the other in the disparity range. We chose the left image as the reference image and found matching pixels in the right image. Hence, the first d columns on the left in the left image may not have the corresponding matching pixels in the right image, where d is the disparity range. In the implementation of SGM and DDR-SGM, we skipped the first d columns on the left and started calculating from the d + 1 column. While the MCCNN can predict the disparity of those edge pixels. Therefore, the MCCNN and DDR-MCCNN output the full resolution disparity maps in our results.

The original SGM is a popular algorithm because of its excellent real-time performance. The results of DDR-SGM shows that the dynamic disparity range can significantly accelerate the algorithm with slight loss of accuracy. The DDR-SGM has better performance on low-resolution images in dataset [24]. The computational time increase when the images become larger. Therefore, the DDR-SGM is suitable for the low-resolution and real-time applications.

The original MCCNN is one of few state-of-the-art algorithms which achieves very high accuracy in recent three years. According to the evaluation results, the error rates of the DDR-MCCNN are lower than those of the DDR-SGM. The tunnel data is an exception that error rate of DDR-MCCNN is higher and it is probably caused by the repetitively texture. On average, the proposed DDR-MCCNN achieved good performance on accuracy. However, the computational complexity of such algorithms are very high and are very time-consuming even on a powerful GPU array. Therefore, the DDR-MCCNN can be used in the high-accuracy applications.

In summary, our DDR optimization reduces computational time significantly while suffer a little loss of accuracy. The dynamic disparity range method can optimize any stereo matching method that follows similar procedure and has pixel-wise matching cost calculation and aggregation steps. By using temporal correlation to narrow the searching range of disparity, we can accelerate the algorithm with only a little loss of accuracy.

5.4 Evaluation of the Temporal Cost Aggregation

As shown in Fig. 11 (a), to evaluate the performance of the temporal cost aggregation, we selected several combinations of lateral paths with and without the temporal path to calculate the b-4 error rate for all pixels and non-occluded pixels respectively. In the Fig. 11 (b) and 11 (c), 2L+1T represents the combination of the L1 and L2 lateral paths with the temporal path. 4L represents the configuration of the L1, L2, L3 and L4 lateral paths. 4L+1T represents the combination of 4L with the temporal path T, and so on.

Out results showed that the error rates of path configuration that combined with temporal path are higher than those without temporal path in first several frames. It is a reasonable because the iterative process needs several frames to improve the accuracy. After about 10 frames of iteration, configurations with the temporal path achieved higher accuracy than those without the temporal path. The computational complexity ratio of 2L+1T and 4L roughly equals 3 : 4. However, 2L+1T achieved higher performance than conventional 4L configuration. The computational complexity ratio of 4L+1T and 5L roughly equals, but the accuracy of 4L+1T was improved significantly. However, for the combination of large number of lateral ag-
aggregation paths, improvement is not as significant as for few lateral aggregation paths combination. 8L+1T performances slightly better than 8L. Taken together, by introducing temporal cost aggregation to conventional SGM, we significantly improved the accuracy with fewer computational demand, especially for those situation that real-time performance is critical and only few paths are allowed to reduce computational complexity. If we simply use the temporal path as an additional aggregation path, it indeed improves the accuracy instead of the speed. However, in some circumstance, few lateral + temporal paths can achieve better performance than more lateral path. For example, 2L+1T paths achieved better accuracy than 4L paths (Fig. 11), which need less computing. It’s more like an optional method that we can choose to use it for higher accuracy or shorter computational time.

We tested the temporal path aggregation on more images in the Middlebury Stereo Dataset and showed the results in Table 5. According to the evaluation results, 2L+1T achieve similar accuracy with 4L while has less computational complexity. 4L+1T achieved better performance than 5L with the same computational complexity. By adding an extra temporal path T, accuracy of 8L+1T is also improved compared with 8L.

### Table 5: All-pixels and non-occluded error rates of paths combinations

| Dataset  | 2L+1T | 4L | 4L+1T | 5L | 8L+1T | 8L |
|----------|-------|----|-------|----|-------|----|
| Motor    | 17.2  | 17.0| 16.5  | 16.8| 15.8  | 15.8|
| Piano    | 11.6  | 11.3| 10.7  | 11.0| 10.2  | 10.2|
| Pipes    | 18.2  | 17.8| 17.4  | 17.4| 16.7  | 16.8|
| Adirondack | 14.5  | 17.6| 14.4  | 15.2| 12.4  | 12.6|

| Dataset  | 2L+1T | 4L | 4L+1T | 5L | 8L+1T | 8L |
|----------|-------|----|-------|----|-------|----|
| Motor    | 10.2  | 9.9 | 9.3   | 9.7 | 8.6   | 8.7|
| Piano    | 7.4   | 7.2 | 6.4   | 6.9 | 5.9   | 6.0|
| Pipes    | 12.3  | 11.4| 11.2  | 11.2| 10.1  | 10.2|
| Adirondack | 8.4   | 12.0| 8.1   | 9.1 | 6.1   | 6.4|

### 5.5 Selection of Parameters R and TH

To evaluate the effects of the range (R) of the dynamic disparity and the threshold (TH), we calculated the computational time and the error rate under different combinations of the two parameters. The results are shown in the Fig. 12. Here R varies from 2 to 24 with a step value of 2, and TH varies from 10 to 50 with a step value of 10.

Results indicate the computational time increases linearly along with the increasing of R, given a certain TH value (Fig. 12a). On the other hand, the error rate decreases along with the increasing of R (Fig. 12b). The higher the TH value is set, the lower the computational time can be achieved. Meanwhile, higher TH value suffers higher error rate. However, when the TH value is larger than 30, the improvement of speed by higher TH gets smaller. In addition, the error rate converged to a constant rapidly when R is larger than 10 and TH is smaller 30.

Therefore, in order to balance the speed and accuracy, we set the range R to 10 and set the threshold TH to 30 in all experiments.
6. Discussion and Conclusion

According to our experiment results, the DDR-SGM achieved significant improvement of speeding up the SGM, which reduced 50% computational time on average, while suffered only very slight loss of accuracy. The DDR-MCCNN reduce more than 25% computational time and suffer slight loss of accuracy. However, the improvement is less than theoretical prediction that is roughly as much as the DDR mask ratio showed. This problem was caused by two main reasons. First, the threshold for aggregation value and the re-calculation at range extended pixels consume extra computational time. Second, most of matching cost operation can be divided into a disparity-range-independent stage and a disparity-range-dependent stage. As shown in our evaluation, we used a two-step strategy to calculate the census transform: The L and R codes calculation step, which is disparity-range-independent and could not be optimized. Only operations of the exclusive-or was reduced by narrowing the disparity range. Even though there are many variants of SGM, we chose the census transform rather than SSD and BT to calculate the matching cost, because the census transform has been proved to have better performance in such real-time application scenario. The evaluation results proved that proposed scheme of DDR can significantly reduce the computational time for all the SGM based algorithms and MCCNN. By using temporal correlation to narrow the searching range of disparity, the proposed DDR algorithm can speed up any stereo matching method that possesses pixel-wise matching cost calculation and aggregation steps.

In addition, we proposed a method that uses temporal cost aggregation to improve the stereo matching. The results proved that our scheme of temporal path can significantly improve the accuracy with fewer computational demand, especially for the scenario that allows only limited paths to perform cost aggregation for speeding up.

In a real-time application scenario, parallel implementation of a stereo matching algorithm is very important. An Algorithm should be easily divided into many independent sub-tasks and those sub-tasks could be processed simultaneously for speeding up. Among various hardware platform, the GPU device received more attentions in recent years. There are a plenty of researches on the parallel implementation of SGM method and using CUDA to program on NVidia GPU devices[13], [31], [32]. CUDA divides the computational units into devices (such as GPUs) and hosts (CPUs). Multiple threads organized in blocks compute the data using the same kernel-function which is known as single-program multiple-data (SPMD). Furthermore, memory management is crucial to programing with CUDA. The global memory is used for data transfer between CPU and GPU, while shared memory is used to across a block which means the threads within the same block can share data with each other.

The SGM method can be well implemented in paral-
References

[1] I.E. Jaafari, M.E. Ansari, L. Koutti, A. Mazoul, and A. Ellahyani, “Fast spatio-temporal stereo matching for advanced driver assistance systems,” Neurocomputing, vol.194, pp.24–33, 2016.

[2] H. Hirschmuller, “Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005, pp.807–814, 2005.

[3] S.K. Gehrig, F. Eberli, and T. Meyer, “A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching,” International Conference on Computer Vision Systems: Computer Vision Systems, pp.134–143, 2009.

[4] H. Hirschmüller, “Stereo processing by semiglobal matching and mutual information,” IEEE Trans. Pattern Anal. Mach. Intell., vol.30, no.2, pp.328–341, 2008.

[5] S. Zagounyko and N. Komodakis, “Learning to compare image patches via convolutional neural networks,” Computer Vision and Pattern Recognition, pp.4353–4361, 2015.

[6] J. Žbontar and Y. LeCun, “Computing the stereo matching cost with a convolutional neural network,” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, pp.1592–1599, 2015.

[7] W. Luo, A.G. Schwing, and R. Urtasun, “Efficient Deep Learning for Stereo Matching,” Computer Vision and Pattern Recognition, pp.5695–5703, 2016.

[8] H. Park and K.M. Lee, “Look Wider to Match Image Patches with Convolutional Neural Networks,” IEEE Signal Process. Lett., vol.24, no.12, pp.1788–1792, Dec. 2017.

[9] J. Žbontar and Y. LeCun, “Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches,” vol.17, pp.2287–2318, 2015.

[10] Scharstein, Daniel, Szeliski, and Richard, “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms,” International Journal of Computer Vision, vol.47, pp.7–42, 2002.

[11] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” Computer Vision and Pattern Recognition, pp.3061–3070, 2015.

[12] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, et al., “EndtoEnd Learning of Geometry and Context for Deep Stereo Regression,” 2017 IEEE International Conference on Computer Vision (ICCV), Venice, pp.66–75, 2017.

[13] M. Michael, I. Salmen, I. Stallkamp, and M. Schlipsing, “Real-time stereo vision: Optimizing Semi-Global Matching,” 2013 IEEE Intelligent Vehicles Symposium (IV), pp.1197–1202, 2013.

[14] K.J. Lee, K. Bong, C. Kim, and H.-J. Yoo, “An intelligent ADAS processor with real-time semi-global matching and intention prediction for 720p stereo vision,” 2016 IEEE Hot Chips 28 Symposium (HCS), pp.1–1, 2016.

[15] C. Banz, S. Hesselbarth, H. Flatt, H. Blume, and P. Pirsch, “Real-time stereo vision system using semi-global matching disparity estimation: Architecture and FPGA-implementation,” International Conference on Embedded Computer Systems, pp.93–101, 2010.

[16] I. Haller and S. Nedevschi, “GPU optimization of the SGM stereo algorithm,” IEEE International Conference on Intelligent Computer Communication and Processing, pp.197–202, 2010.

[17] S. Sim, K. Min, S. Lee, H. Son, and J. Kim, “Efficient hardware architecture for real-time Semi-Global Matching,” Soc Design Conference, pp.262–263, 2015.

[18] F. Schumacher and T. Greiner, “Matching cost computation algorithm and high speed FPGA architecture for high quality real-time Semi Global Matching stereo vision for road scenes,” IEEE International Conference on Intelligent Transportation Systems, pp.3064–3069, 2014.

[19] R. Spangenberg, T. Langner, S. Adelfeld, and R. Rojas, “Large scale Semi-Global Matching on the CPU,” Intelligent Vehicles Symposium Proceedings, pp.195–201, 2014.

[20] H. Tao, H.S. Sawhney, and R. Kumar, “Dynamic depth recovery from multiple synchronized video streams,” Proc. 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, vol.1, pp.I-118–I-124, 2001.

[21] L. Zhang, B. Curless, and S.M. Seitz, “Spacetime stereo: shape recovery for dynamic scenes,” Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, vol.2, pp.II-367-74, 2003.

[22] J. Davis, D. Nehab, R. Ramamoorthi, and S. Rusinkiewicz, “Space-time stereo: a unifying framework for depth from triangulation,” IEEE Trans. Pattern Anal. Mach. Intell., vol.27, no.2, pp.296–302, 2005.

[23] M. Gong, “Enforcing Temporal Consistency in Real-Time Stereo Estimation,” Lecture notes in computer science, vol.3953, pp.564–577, 2006.

[24] C. Richardt, D. Orr, I. Davies, A. Criminisi, and N.A. Dodgson, “Real-Time Spatiotemporal Stereo Matching Using the Dual-Cross-Bilateral Grid,” Proc. Computer Vision - ECCV 2010, European Conference on Computer Vision, Heraklion, Crete, Greece, Sept. 5-11, 2010, pp.510–523, 2010.

[25] A. Hosni, C. Rhemann, M. Bleyer, and M. Gelautz, “Temporally consistent disparity and optical flow via efficient spatio-temporal filtering,” Pacific Rim Conference on Advances in Image and Video Technology, pp.165–177, 2012.

[26] H. Liu, C. Liu, Y. Tang, H. Sun, and X. Li, “Spatio-Temporal Consistency Enhancement for Disparity Sequence,” International Journal of Signal Processing Image Processing & Pattern Recognition, vol.7, 2014.

[27] Y. Shin and K.-J. Yoon, “Spatiotemporal stereo matching for dynamic scenes with temporal disparity variation,” 2013 IEEE International Conference on Image Processing, pp.2242–2246, 2013.

[28] F. Huguet and F. Devernay, “A Variational Method for Scene Flow Estimation from Stereo Sequences,” 2007 IEEE 11th International Conference on Computer Vision, pp.1–7, 2007.

[29] D. Min, J. Lu, and M.N. Do, “Depth Video Enhancement Based on Weighted Mode Filtering,” IEEE Trans. Image Process., vol.21, no.3, pp.1176–1190, 2012.

[30] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, et al., “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, pp.4040–4048, 2016.

[31] K. Zhu, M. Butenuth, and P. d’Angelo, “Comparison of Dense Stereo Using CUDA,” Trends and Topics in Computer Vision: ECCV 2010 Workshops, Heraklion, Crete, Greece, Sept. 10-11, 2010, Revised Selected Papers, Part II, K.N. Kutulakos, Ed., ed Berlin, Heidelberg: Springer Berlin Heidelberg, pp.398–410, 2012.

[32] C.D. Pantilie and S. Nedevschi, “Optimizing the Census Transform on CUDA enabled GPUs,” 2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing, pp.201–207, 2012.
Ming Li received the B.S. degree in School of Electronic Science and Engineering from Nanjing University, Nanjing, China, in 2017. He is currently working towards M.S. degree in Nanjing University. His research interests include image processing, computer vision and machine learning.

Li Shi received his B.S. degree in electronics engineering from Nanjing University, Nanjing, China, in 2015. He is currently working toward the M.S. degree in Nanjing University, Nanjing, China, since 2015. His research interests include computer vision and image processing.

Xudong Chen received the B.S. degree in School of Electronic Science and Engineering from Nanjing University, Nanjing, China, in 2017. He is a master degree candidate at Nanjing University. His research interests include 3D representation and computer vision.

Sidan Du (M’06) received the B.S. and M.S. degrees in electronic engineering from Xidian University, Xian, China in 1984 and 1987, respectively, and the Ph.D. degree in physics from Nanjing University, Nanjing, China, in 1997. She is currently a Professor in the school of Electronic Science and Engineering, Nanjing University. Her research interests include signal processing and computer vision.

Yang Li (M’09) received the B.S. and M.S. degrees in mechanical engineering from Southeast University, Nanjing, China, in 2000 and 2003, respectively, and the Ph.D. degree in electronics engineering from Nanjing University, Nanjing, China, in 2006. He is an Associated Professor with School of Electronic Science and Engineering, Nanjing University, China, since 2009. His research interests include digital signal processing and computer vision.