Enforcing Temporal Consistency in Real-Time Stereo Estimation

Minglun Gong

Department of Math and Computer Science,
Laurentian University, Sudbury, ON, Canada

Abstract. Real-time stereo matching has many important applications in areas such as robotic navigation and immersive teleconferencing. When processing stereo sequences most existing real-time stereo algorithms calculate disparity maps for different frames independently without considering temporal consistency between adjacent frames. While it is known that temporal consistency information can help to produce better results, there is no efficient way to enforce temporal consistency in real-time applications.

In this paper the temporal correspondences between disparity maps of adjacent frames are modeled using a new concept called disparity flow. A disparity flow map for a given view depicts the 3D motion in the scene that is observed from this view. An algorithm is developed to compute both disparity maps and disparity flow maps in an integrated process. The disparity flow map generated for the current frame is used to predict the disparity map for the next frame and hence, the temporal consistency between the two frames is enforced. All computations are performed in the image space of the given view, leading to an efficient implementation. In addition, most calculations are executed on programmable graphics hardware which further accelerates the processing speed. The current implementation can achieve 89 million disparity estimations per second on an ATI Radeon X800 graphic card. Experimental results on two stereo sequences demonstrate the effectiveness of the algorithm.

1 Introduction

Stereo vision studies how to estimate disparity maps based on spatial correspondences among the input images captured at different views [2, 11]. It has been one of the most actively researched topics in computer vision with a variety of algorithms proposed in the past few years. Some of these have obtained excellent results by casting stereo vision as a global optimization problem and solving it using techniques such as graph cuts [7] and belief propagation [12].

Many applications, including robot navigation and immersive teleconferencing, require disparity maps to be generated in real-time. While global optimization techniques help to produce accurate disparity maps, they generally require long computation time. Most real-time stereo applications today either optimize each pixel locally using a simple winner-take-all (WTA) approach [4, 6, 8, 15-17] or optimize different scanlines separately using dynamic programming [3, 5].

When handling a stereo sequence, the above real-time algorithms process different frames in the sequence independently, without considering the temporal consistency

A. Leonardis, H. Bischof, and A. Pinz (Eds.): ECCV 2006, Part III, LNCS 3953, pp. 564 – 577, 2006.

between adjacent frames. Previous research has shown that that utilizing temporal consistency information helps to produce better results [1, 13, 18]. However, there is no efficient way to enforce temporal consistency in real-time stereo matching.

In this paper, the temporal consistency between disparity maps of adjacent frames is modeled using a new concept called disparity flow. Disparity flow is defined in the disparity space of a given view and can be considered as view-dependent scene flow [14]. Just as a disparity map is a 2D array of scalars describing the observation of the 3D geometry in the scene from a given view, a disparity flow map is a 2D array of 3D vectors depicting the observation of the 3D motion in the scene from a given view. Since both disparity maps and disparity flow maps are defined in the disparity space, using disparity flow maps to enforce temporal consistency is very efficient.

An algorithm is presented in this paper to compute both disparity maps and disparity flow maps in an integrated process. The disparity flow map obtained for the current frame is used to predict the disparity map for the next frame and hence the temporal consistency between the two frames is enforced. The disparity maps found also provide the spatial correspondence information which is used to cross-validate the disparity flow maps estimated for different views.

All computations involved in the algorithm can be performed in the image space of a given view. This allows for efficient implementation using programmable graphics hardware which further accelerates the processing speed. When handling binocular stereo sequences, the current implementation can produce disparity maps for both views at 17.8 frames per second (fps) on an ATI Radeon X800 graphic card, i.e., about 89 million disparity estimations per second (Mde/s).

1.1 Related Works

Several techniques have been proposed to obtain more accurate disparity maps from stereo sequences by utilizing consistency in the temporal domain [1, 13, 18]. Some of them assume either that the scenes are static/quasi-static or that the motion is negligible compared to the sampling frequency [1, 18]. These approaches can produce accurate disparity maps for static scenes based on stereo sequences captured under varying lighting conditions, but they have difficulty handling dynamic scenes or scenes with constant lighting.

How to enforce temporal consistency for dynamic scenes has been investigated in [9, 13]. In Tao et al.'s approach [13], the input images are segmented into homogeneous color regions and each segment is modeled using a 3D planar surface patch. The projections of a given planar patch on two adjacent frames are related by a temporal homography. The temporal homography, together with the spatial homography, is then used to estimate the parameters of the planar patch. Since their approach is segmentation-based both the accuracy of the results and the processing speed are limited by the image segmentation algorithm used.

Leung et al.'s approach [9] does not require image segmentation. The temporal consistency is enforced by minimizing the difference between the disparity maps of adjacent frames. However, since disparity changes are always penalized, this approach may have difficulties in handling scenes that contain large motions. This approach is also designed for offline processing only — it takes pre-captured stereo sequences as input and calculates the disparity maps for all frames at the same time.

Different from the above approaches, the proposed algorithm models temporal consistency in disparity space using the concept of disparity flow. This makes it possible to enforce temporal consistency in real-time online stereo calculation.

This paper is also related to existing graphics hardware based stereo matching techniques. Modern programmable graphics hardware allows developers to write their own computational kernels that can be executed in parallel on the Graphics Processing Units (GPUs). Several approaches have been proposed to accelerate the stereo matching computation using the processing power of GPUs [5, 15-17]. They typically use the graphics hardware's texture capability to compute the matching costs and then select optimal disparity values for different pixels. In [17], a very high processing speed of 289Mde/s has been achieved on an ATI 9800 card.

The proposed algorithm differs from existing GPU-based stereo approaches in that it estimates both disparity maps and disparity flow maps for the input stereo sequence. The disparity flow maps obtained depict the 3D motion of the scene and are used to enforce temporal consistency between disparity maps of adjacent frames. As yet, there seem to be no published reports on implementing a 3D motion estimation algorithm on the GPU.

2 Definition of Disparity Flow

A disparity flow map is a 2D array of 3D vectors defined on a particular view. Assume that at frame t and under a given view k, a pixel (u,v) has a disparity value d (the disparity can be defined either based on a stereo pair or more generally based on the inverse distance between the corresponding 3D point and the image plane of view k [10]). The triple $<u,v,d>$ is called the disparity space coordinate of the corresponding 3D point at frame t and under view k. Due to the motion of this 3D point, the disparity space coordinate of this point may change to $<u+\Delta u,v+\Delta v,d+\Delta d>$ at frame t+1. The difference between the two coordinates, $<\Delta u,\Delta v,\Delta d>$, is defined as the disparity flow of pixel (u,v) at frame t.

The relationships among disparity, disparity flow, and optical flow is illustrated in Fig. 1 using a binocular stereo scenario. Assume that, at time t, a physical point at

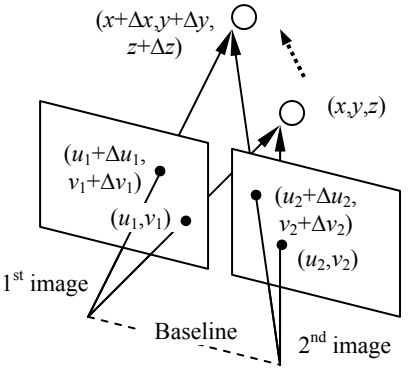


Fig. 1. Disparity flow under a binocular stereo scenario

location (x,y,z) in the scene is observed at pixel (u_1,v_1) in the first image and at pixel (u_2,v_2) in the second image. This point has a 3D motion $(\Delta x,\Delta y,\Delta z)$, which causes optical flows of $(\Delta u_1,\Delta v_1)$ and $(\Delta u_2,\Delta v_2)$ being observed in the two images. Now assume that, when observed from the first view, the disparity of the physical point is d_1 at time t and $d_1+\Delta d_1$ at time $t+1$. When observed from the second view, the disparity of the same point is d_2 and $d_2+\Delta d_2$ for time t and time $t+1$. According to the definition above, the disparity flow of pixel (u_1,v_1) is $<\Delta u_1,\Delta v_1,\Delta d_1>$, and that of pixel (u_2,v_2) is $<\Delta u_2,\Delta v_2,\Delta d_2>$. As a result, the first two coordinates in the disparity flow of a given pixel is simply the optical flow observed at that pixel, while the third coordinate is equal to the change in the corresponding 3D point's disparity value.

2.1 Constraints Between Disparity and Disparity Flow

While the disparity flow map provides temporal correspondences between disparity maps of adjacent frames, the disparity map also provides spatial correspondences between the disparity flow maps of different views. Two additional constraints can be derived based on these relations, which help to produce better disparity and disparity flow maps. In this section, these two constraints are formulated under a rectified left-and-right stereo scenario. It is noteworthy that similar constraints exist for arbitrary stereo pairs, though not in as concise a form.

First, a disparity flow map obtained at a given view can be used to enforce a temporal consistency constraint between the disparity maps of adjacent frames at the same view. Assume that the disparity and the disparity flow found for a given pixel (u,v) at view k are d and $<\Delta u,\Delta v,\Delta d>$, respectively. According to the definition of the disparity flow, this suggests that the corresponding 3D point moves from coordinates $<u,v,d>$ in the disparity space of view k to coordinates $<u+\Delta u,v+\Delta v,d+\Delta d>$. Therefore, the disparity of pixel $(u+\Delta u,v+\Delta v)$ in the next frame should be $d+\Delta d$, i.e., the following temporal consistency constraint holds:

$$\left.\begin{aligned} D_t(u,v) &= d \\ \mathbf{F}_t(u,v) &= \langle \Delta u, \Delta v, \Delta d \rangle \end{aligned}\right\} \Rightarrow D_{t+1}(u+\Delta u, v+\Delta v) = d + \Delta d \qquad (1)$$

where $D_t(u,v)$ and $\mathbf{F}_t(u,v)$ are the disparity and disparity flow of pixel (u,v) at frame t.

Secondly, a disparity map obtained for a given frame also provides a spatial consistency constraint on disparity flow maps generated at different views for the same frame. As shown in Fig. 1, assume that pixel (u_2,v_2) in the right view is the corresponding pixel of (u_1,v_1) in the left view and that the disparity flows of these two pixels are $<\Delta u_1,\Delta v_1,\Delta d_1>$ and $<\Delta u_2,\Delta v_2,\Delta d_2>$, respectively. Then:

- the epipolar constraint gives:

$$\left.\begin{aligned} v_1 &= v_2 \\ v_1 + \Delta v_1 &= v_2 + \Delta v_2 \end{aligned}\right\} \Rightarrow \Delta v_1 = \Delta v_2$$

- since the two image planes are coplanar, the distances from the 3D point to both image planes are the same, i.e.:

$$d_1 = d_2 = d$$
$$\Delta d_1 = \Delta d_2 = \Delta d$$

- finally, based on the definition of the disparity, the following can be derived:

$$\left.\begin{array}{l} d = u_1 - u_2 \\ d + \Delta d = (u_1 + \Delta u_1) - (u_2 + \Delta u_2) \end{array}\right\} \Rightarrow \Delta u_1 = \Delta u_2 + \Delta d$$

Hence, the following spatial consistency constraint can be derived, which specifies the relations between the disparity flow maps for the left and the right views:

$$\left.\begin{array}{l} D^{right}(u,v) = d \\ \mathbf{F}^{right}(u,v) = \langle \Delta u, \Delta v, \Delta d \rangle \end{array}\right\} \Leftrightarrow \left\{\begin{array}{l} D^{left}(u+d,v) = d \\ \mathbf{F}^{left}(u+d,v) = \langle \Delta u + \Delta d, \Delta v, \Delta d \rangle \end{array}\right. \quad (2)$$

where $D^{left/right}(u,v)$ and $\mathbf{F}^{left/right}(u,v)$ are the disparity and disparity flow of pixel (u,v) in the left/right views, respectively.

3 The Proposed Real-Time Stereo Algorithm

In order to derive a simple and efficient algorithm for real-time applications, it is assumed that the input stereo sequences are pre-rectified. In addition, scenes are assumed to be Lambertian so that the constant brightness assumption holds. For simplicity, here the algorithm is discussed under a binocular (left-and-right) stereo setting. As shown in the experiments, when trinocular (left-center-top) stereo sequences are available the algorithm can also make use of the additional view to better solve the visibility problem.

The outline of the presented algorithm is shown in Fig. 2 In the following sections, different stages shown in the figure are discussed in detail.

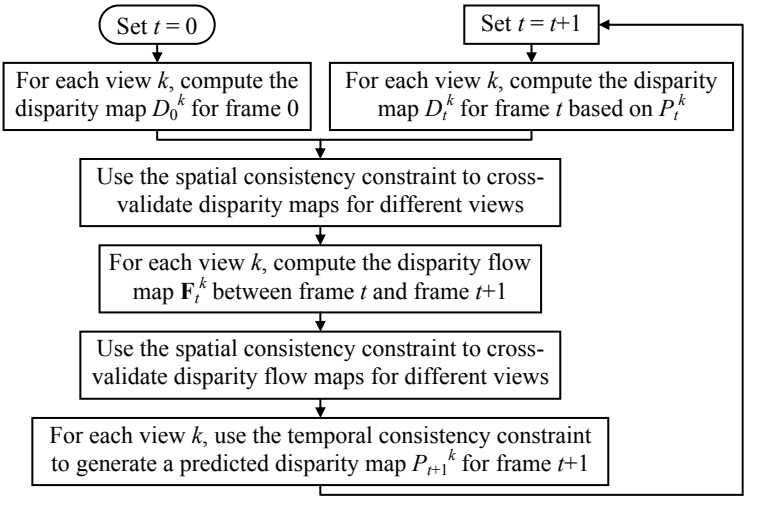


Fig. 2. The outline of the proposed real-time stereo algorithm

3.1 Compute a Disparity Map for the First Frame of Each View

For a given binocular stereo sequence, the algorithm starts by computing the disparity maps for the first frames of both views. Since there is no temporal consistency information available yet, the process used at this step is similar to existing GPU-based stereo algorithms [15-17]. Without losing generality, only the process for the right view in the left-and-right stereo pair is described in detail. This process involves three steps: matching costs calculation, cost aggregation, and disparity optimization.

The first step calculates the costs of assigning different disparity hypotheses to different pixels at the first frame (frame 0) of the right view. The obtained costs form a 3D matrix $\mathbf{C}_0^{right}$, which is often referred as the disparity space. Based on the constant brightness assumption, the cost for each disparity assignment is calculated using the color differences between the corresponding pixels in the two views, i.e.:

$$\mathbf{C}_0^{right}[u,v,d] = \frac{\min\left(\left|I_0^{right}(u,v) - I_0^{left}(u+d,v)\right|, c_{\max}\right)}{c_{\max}}$$

where $I_t^k(u,v)$ is the color of pixel (u,v) at frame t of view k. $c_{\max}$ is a predefined value for maximum matching cost. For color images the average absolute difference among the three color channels is used. The final costs are normalized to the interval [0,1].

In order to achieve real-time performance the above cost calculation is conducted on the GPU. The input stereo images are treated as textures and a pixel shader is used to calculate the cost for different pixels under different disparity hypotheses. To fully utilize the vector processing capacity of the GPU, the pixel shader calculates the matching costs for four different disparity hypotheses in one rendering pass and packs the costs into the four color channels of the rendering target. For efficiency, the results obtained for the four different disparity hypotheses groups are tiled together and kept as a single 2D texture (see Fig. 3 as an example).

In the second step, the matching costs calculated based on a single pixel are propagated to its neighbors. Similar to existing stereo approaches, the shiftable square window is used [11]. The entire aggregation step is implemented on the GPU with four rendering passes involved. The first two rendering passes replace a current cost with the average cost of its local neighbors along horizontal and then vertical directions, which give the effect of mean filtering over a local square window. The next two passes replace a cost with the minimum cost of its local neighbors along horizontal and then vertical directions, which give the effect of a shift filter. In the experiments shown in this paper, 9×9 mean filter and 5×5 shift filter are used.

The third step searches for an optimal disparity map D_0^{right} based on the cost matrix $\mathbf{C}_0^{right}$. To achieve real-time performance the simplest local WTA optimization is used to find the disparity value that gives the smallest matching cost at each pixel in the image. This process requires $D/4$ rendering passes, where D is the total number of disparity hypotheses. The first rendering pass takes the first tile in the 2D texture shown in Fig. 3 as input, computes for different pixels the smallest costs among those for the first four disparity hypotheses, and stores the costs and the corresponding disparity values in the green and red channels of the rendering target respectively. The remaining rendering passes step through other tiles in the texture

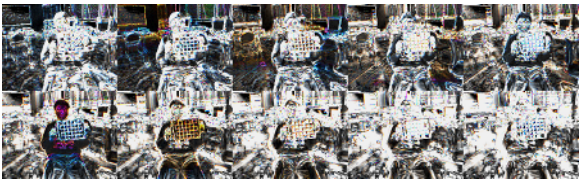
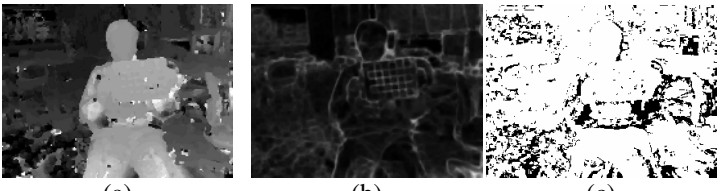


Fig. 3. The texture used for representing the 3D disparity space. The matching costs under 40 different disparity hypotheses are encoded using 10 tiles.



(a) (b) (c)

Fig. 4. The texture that encodes the result of disparity computation. The red channel (a) keeps the best disparity hypotheses, the green channel (b) stores the corresponding matching costs, and the blue channel (c) indicates whether the disparity values pass the validation. Image intensity in (a) is adjusted for better visibility.

and update the minimum costs and the best disparity values at different pixels. At the end of the process, the red channel of the output texture holds the disparity maps generated (shown in Fig. 4).

3.2 Cross-Validate Disparity Maps for Different Views

The WTA optimization is efficient but may produce noisy disparity maps. In order to distinguish correct disparity values from potentially incorrect ones, a cross-validation process is implemented to verify the disparity maps generated for the current frame *t*. According to the spatial consistency constraint (Eq. 2), the disparity value at pixel (*u*,*v*) in the right view is considered potentially incorrect if it fails the test below:

$$\left| D_t^{right}(u,v) - D_t^{left}\left(u + D_t^{right}(u,v), v\right) \right| \le 1$$

Please note that the above test condition does not require the corresponding disparity values in the two views to be exactly the same. This is reasonable as the true value normally lies in between two quantized disparity values. Similarly, the disparity map for the left view is validated using the following criteria:

$$\left| D_t^{left}(u,v) - D_t^{right}\left(u - D_t^{left}(u,v), v\right) \right| \le 1$$

The validation process is implemented on the GPU using one rendering pass for each view. When processing view *k*, the pixel shader takes both disparity maps as input, tests the disparity value of each pixel in view *k*, and sets the blue channel of each pixel to either '1' or '0' according to whether the corresponding disparity value passes the validation (see Fig. 4 as an example).

3.3 Compute a Disparity Flow Map for Each View

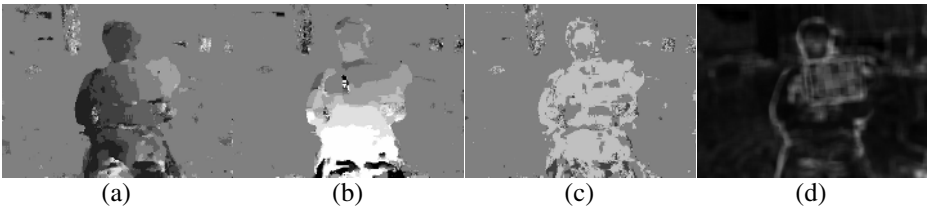
In the next stage, a disparity flow map between frame t and frame t+1 is computed for each of the two views. Again, only the process for the right view is described in detail. The one for the left view is similar.

Under the binocular stereo setting, if a pixel (u,v) in the right view has a disparity value of d and a disparity flow of $<\Delta u,\Delta v,\Delta d>$, in the next frame the corresponding 3D point should move to a location that projects to pixel $(u+\Delta u,v+\Delta v)$ in the right view and to pixel $(u+\Delta u+d+\Delta d,v+\Delta v)$ in the left view. Based on the constant brightness assumption, the correct disparity flow for pixel (u,v) should minimize the color difference between $I_t^{right}(u,v)$ and $I_{t+1}^{right}(u+\Delta u,v+\Delta v)$, as well as between $I_t^{right}(u,v)$ and $I_{t+1}^{left}(u+\Delta u+d+\Delta d,v+\Delta v)$. Similar to the disparity map computation process, the costs for assigning different disparity flow hypotheses to different pixels in the frame t of the right view are kept in a 5D matrix $\mathbf{B}_t^{right}$. This 5D matrix is referred as the disparity flow space in this paper and is calculated using:

$$\mathbf{B}_t^{right}[u,v,\Delta u,\Delta v,\Delta d]=\frac{\min\left(\begin{array}{c}\left|I_t^{right}(u,v)-I_{t+1}^{right}(u+\Delta u,v+\Delta v)\right|+\\ \left|I_t^{right}(u,v)-I_{t+1}^{left}(u+\Delta u+D_t^{right}(u,v)+\Delta d,v+\Delta v)\right|,2c_{\max}\end{array}\right)}{2c_{\max}}$$

The above equation is calculated on the GPU using a process similar to the one for computing costs in the disparity space. For efficiency, the costs in the 5D disparity flow space are also packed into a 2D color texture. The color texture contains multiple tiles with each tile keeping the matching costs for all pixels under four different disparity flow hypotheses. Under this packing scheme, costs in a given tile can be calculated using a single rendering pass: the pixel shader takes I_t^{right}, D_t^{right}, I_{t+1}^{right}, and I_{t+1}^{left} as input textures, calculates the matching costs for the current pixel under the four disparity flow hypotheses, and stores the costs into different channels of the rendering target.

The next step is cost aggregation, in which the matching costs are convoluted on the GPU with a 9×9 mean filter, followed by a 5×5 shift filter. The aggregated matching costs are then used for searching optimal disparity flows using a GPU-based local WTA procedure. Similar to the one described in section 0, this procedure takes multiple rendering passes, with each rendering pass handling four disparity flow hypotheses. The output of the procedure is a 2D texture with the minimum matching



(a) (b) (c) (d)

Fig. 5. The texture that encodes the result of disparity flow computation. The red (a), green (b), and blue (c) channels keep the Δu, Δv, and Δd components of the best disparity flow hypotheses. The alpha channel (d) stores the corresponding matching costs. Image intensities in (a), (b), and (c) are adjusted for better visibility.

cost for each pixel stored in its alpha channel and the three components of the corresponding disparity flow vector encoded in its red, green, and blue channels (see Fig. 5 as an example).

3.4 Cross-Validate Disparity Flow Maps for Different Views

Similar to the case of disparity map computation, the disparity flow maps generated using local WTA approach can be noisy. An additional validation process is used to distinguish correct disparity flows from potentially incorrect ones so that only the former ones are used to enforce the temporal consistency constraint.

Based on the spatial consistency constraint (Eq. 2), the disparity flow found for pixel (u,v) in the right view is considered potentially incorrect if it fails the test below:

$$\mathbf{F}_t^{right}(u,v) = \mathbf{F}_t^{left}\left(u + D_t^{right}(u,v), v\right) - \left\langle \mathbf{F}_t^{right}(u,v)\big|_d, 0, 0 \right\rangle$$

where $\mathbf{F}|_d$ is the Δd component of the disparity flow vector.

Similarly, the disparity flow for the left view is validated using the following criteria:

$$\mathbf{F}_t^{left}(u,v) = \mathbf{F}_t^{right}\left(u - D_t^{left}(u,v), v\right) + \left\langle \mathbf{F}_t^{left}(u,v)\big|_d, 0, 0 \right\rangle$$

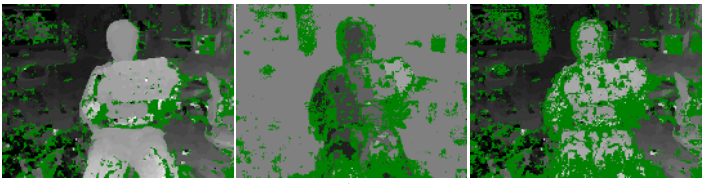
The validation process is implemented on the GPU and takes one rendering pass for each view to be validated. The pixel shader takes both disparity flow maps as input textures and tests disparity flows for different pixels of the current view. The output is a copy of the original disparity flow map with information about whether a given disparity value passes the validation stored in the alpha channel of the corresponding pixel.

3.5 Predict a Disparity Map for the Next Frame of Each View

With both disparity maps and disparity flow maps calculated for frame t, it is now possible to predict the disparity maps for frame t+1 based on the temporal consistency constraint. For a given view k, the predicted disparity map $P_{t+1}{}^k$ is calculated from D_t^k and $\mathbf{F}_t^k$ based on Eq. 1. To prevent error propagation, a pixel is used if and only if both the disparity value and the disparity flow found for this pixel are validated, i.e., they pass the corresponding cross-validation processes.

The predicting process goes through all pixels that have validated disparity values and disparity flows, warps these pixels to the next frame based on their disparity flows, and sets the disparity values for the corresponding pixels. It is possible that two or more pixels in the current frame are warped to the same pixel in the next frame. In such a case, the highest disparity value will be used as it represents the 3D point that is the closest to the camera and should be the visible one.

Implementing the above forward mapping process on current graphics hardware is difficult as writing to an arbitrary position of the rendering target is not supported. Hence, this process is implemented using a CPU-based procedure. Since the inputs of the procedure are the disparity map and disparity flow map and the output is a single predicted disparity map, there is very little overhead for transferring data between the system memory and the video memory.



(a) (b) (c)

Fig. 6. Disparity prediction process: (a) validated disparity map for frame t; (b) validated disparity flow map (Δu channel only) for frame t; (c) predicted disparity map for frame t+1. Green color indicates pixels that do not pass the validation.

3.6 Compute Disparity Maps Based on Previous Predictions

When predicted disparity maps are generated by previous calculations, the process used for computing the disparity maps differs slightly from the one described in section 0. When computing D_t^k for each give view k, the new process uses the predicted disparity map P_t^k as a guide so that validated matches found for the previous frame can help to solve ambiguities.

As shown in Fig. 6(c), even though the P_t^k is generated using only validated disparity values and disparity flows found for the previous frame, it may still contain mismatches. Therefore, if all disparity values predicted by P_t^k were selected into D_t^k directly, these mismatches would be propagated over different frames. To prevent error propagation, in the proposed algorithm the predicted disparity values are used to adjust matching costs only. The final disparity value calculated for a given view may differ from the original prediction.

The new disparity map computation process involves four steps: matching cost calculation, cost adjustment, cost aggregation, and disparity optimization. In the first step, a 3D matching cost matrix $\mathbf{C}_t^k$ is initialized based on the t^{th} frames captured at different views. The same GPU-based procedure as the one described in section 0 is used here. The output of the procedure is a 2D texture, shown in Fig. 7(a), encoding the costs calculated.

The second step takes both $\mathbf{C}_t^k$ and P_t^k as input textures and updates the matching cost matrix using the following equation:

$$\mathbf{C}_t^k[u,v,d] = \begin{cases} \mathbf{C}_t^k[u,v,d] \times 3 & \text{If } P_t^k(u,v) \text{ is validated} \wedge P_t^k(u,v) \neq d \\ \mathbf{C}_t^k[u,v,d] & \text{Otherwise} \end{cases}$$

As suggested by the equation, if a disparity value $P_t^k(u,v)$ is predicted for pixel (u,v), the matching costs for all other disparity hypotheses d, $d \neq P_t^k(u,v)$, are tripled. This encourages disparity value $P_t^k(u,v)$ to be selected into the final disparity map D_t^k so that D_t^k is temporally consistent with D_{t-1}^k.

The cost adjustment step is implemented on the GPU using one rendering pass. The output of the shader is a 2D texture that encodes the adjusted matching costs. As shown in Fig. 7(c), the texture that encodes the adjusted matching costs has higher contrast than the one encoding the unadjusted costs. This suggests that there should be fewer ambiguous matches when using the adjusted costs to compute disparity map.

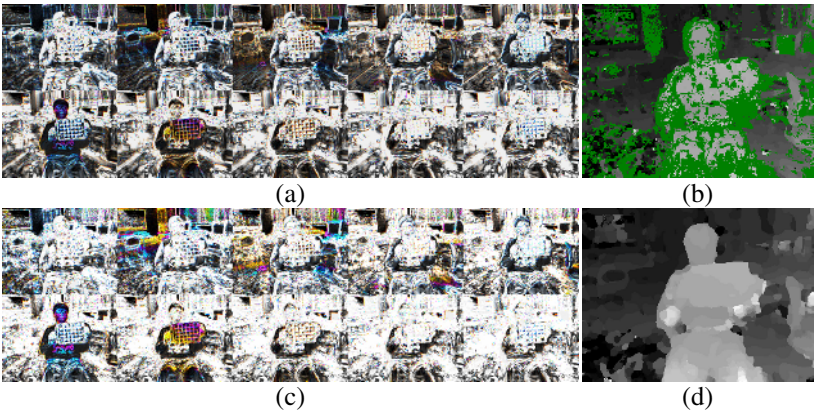


Fig. 7. Stereo matching based on previous prediction: (a) matching costs calculated using input images; (b) previous disparity map prediction P_t^k (the same image as Fig. 6(c)); (c) adjusted matching cost based on P_t^k; (d) the final disparity map D_t^k, in which many incorrect predictions shown in P_t^k are corrected

In the next two steps, the adjusted matching costs are aggregated before they are used for computing the disparity map. The same GPU-based procedures as the ones described in section 0 are used in these two steps. As shown in Fig. 7(d), since the disparity optimization step is based on the aggregated matching costs, isolated noise in the predicted disparity maps is corrected.

4 Experimental Results

The algorithm presented in this paper is tested using a variety of stereo sequences. Due to the space limits, only the results for two sequences are shown here. The first one is a color sequence captured using PointGrey's Bumblebee stereo camera; while the second one is a grayscale sequence captured using PointGrey's Digiclops camera. Both sequences are captured at 512×384 resolution, but are cropped and downsampled to 288×216 to focus on the moving persons as well as to remove the black border and oversampling caused by the rectification process. As shown in Fig. 8(a), both sequences are challenging due to the existence of textureless surfaces (whiteboard in both scenes), non-Lambertian reflection (highlighted background wall in the first scene and floor in the second scene), and areas with periodic textures (checkerboard pattern in the second scene).

The algorithm is configured to use the same set of parameters for both sequences. The disparity computation process considers 40 different disparity hypotheses. The search range for the disparity flow computation is set to [-4,4] for both horizontal and vertical directions, but is set to [-1,1] for the disparity direction since the motion along the disparity direction is much smaller. As a result, the total number of disparity flow hypotheses is 243.

As shown in Fig. 8(b), for both sequences the disparity maps generated using only the 15th frames captured at different views are quite noisy. Many mismatches are

Fig. 8. Results comparison on the 15th frames of both sequences: (a) source images; (b) disparity maps generated without enforcing temporal consistency constraint; (c) disparity maps generated with temporal consistency constraint enforced; (d) cross-validated disparity maps using spatial consistency constraint

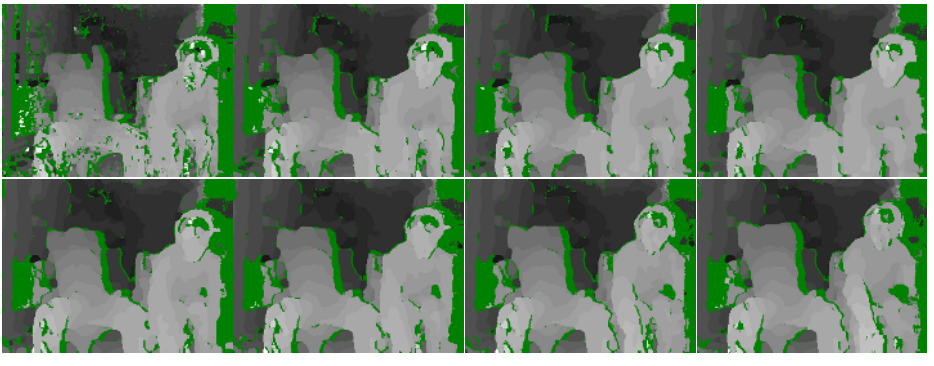


Fig. 9. Semi-dense disparity maps generated for the first eight frames in the binocular sequence. As more frames became available, there are fewer mismatches caused by ambiguities and fewer pixels with unvalidated disparities as well.

caused by ambiguities such as in the area of the checkerboard pattern in the second scene. Enforcing temporal consistency constraint helps to remove these mismatches. The disparity flows obtained, shown in Fig. 8(c), are much smoother and appear mostly accurate. Most of the remaining errors are caused by occlusions, which are filtered out in the semi-dense disparity maps obtained after the cross-validation process (shown in Fig. 8(d)). It is noteworthy that there are considerably fewer mismatches caused by occlusions in the results generated for the trinocular sequence as the algorithm can utilize the additional view to solve occlusions.

A screen captured animation is submitted with the paper. The animation compares the disparity sequences generated using both the single frame approach and the proposed approach. The latter shows noticeable improvements in temporal consistency. At the beginning of the animation one can also observe how mismatches are gradually removed from the estimation results as more frames become available (see Fig. 9 as an example).

In terms of the processing speed, testing shows that, for the binocular sequence above, the current implementation can generate disparity maps for both the left and the right views at 17.8 fps on a 3GHz P4 computer equipped with an ATI Radeon X800 card. This means that the algorithm can perform 89M disparity evaluations per second. It is worth noting that the disparity flow maps for the two views are also generated at the same time.

5 Conclusions

A real-time stereo matching algorithm that enforces temporal consistency constraint is presented in this paper. The temporal correspondences between the disparity maps of adjacent frames are modeled using disparity flow, which can be considered as view-dependent scene flow. The concept of disparity flow provides simple and efficient ways to enforce temporal consistency between disparity maps generated for the adjacent frames at the same view, as well as to cross-validate the spatial consistency between disparity flow maps generated for the same frame at different views.

The proposed algorithm integrates the disparity map and disparity flow map computations in an integrated process. As a result, both computations benefit from each other. In particular, when generating disparity map for the next frame, the disparity flow map obtained for the current frame is used to enforce the temporal consistency constraint through a disparity predicting process. To prevent mismatches being propagated over time, only validated disparity values and disparity flows are used in the disparity predicting process. Furthermore, the predicted disparity values are used to guide the disparity computation through the cost adjustment process instead of being used directly in the disparity map for the next frame.

In order to achieve real-time performance and to utilize the processing power of GPUs the proposed algorithm uses the simplest WTA optimization in both disparity and disparity flow computation. However, the idea of using disparity flow map to enforce temporal consistency constraint can be integrated with global optimization techniques as well. The resulting algorithm will be able to produce temporally consistent disparity maps for stereo sequences though not at real time speed.

Acknowledgements

The author would like to thank Mr. Cheng Lei and Mr. Liang Wang for capturing the stereo sequences used in this paper. This research is supported by NSERC and Laurentian University.

References

1. Davis, J., Nehab, D., Ramamoothi, R., and Rusinkiewicz, S.: Spacetime stereo: a unifying framework for depth from triangulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **27** (2005)
2. Dhond, U. R. and Aggarwal, J. K.: Structure from stereo - A review. *IEEE Transactions on Systems, Man and Cybernetics*. **19** (1989) 1489-1510.

3. Forstmann, S., Ohya, J., Kanou, Y., Schmitt, A., and Thuering, S.: Real-time stereo by using dynamic programming. *Proc. CVPR Workshop on Real-time 3D Sensors and Their Use*. Washington, DC, USA. (2004) 29-36.
4. Gong, M. and Yang, R.: Image-gradient-guided real-time stereo on graphics hardware. *Proc. International Conference on 3-D Digital Imaging and Modeling*. Ottawa, ON, Canada. (2005) 548-555.
5. Gong, M. and Yang, Y.-H.: Near real-time reliable stereo matching using programmable graphics hardware. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. San Diego, CA, USA. (2005) 924-931.
6. Hirschmuller, H., Innocent, P. R., and Garibaldi, J.: Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*. **47** (2002)
7. Hong, L. and Chen, G.: Segment-based stereo matching using graph cuts. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. Washington, DC, USA. (2004) 74-81.
8. Kanade, T., Yoshida, A., Oda, K., Kano, H., and Tanaka, M.: A stereo engine for video-rate dense depth mapping and its new applications. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. (1996) 196-202.
9. Leung, C., Appleton, B., Lovell, B. C., and Sun, C.: An energy minimisation approach to stereo-temporal dense reconstruction. *Proc. International Conference on Pattern Recognition*. Cambridge, UK. (2004) 72-75.
10. Okutomi, M. and Kanade, T.: A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **15** (1993) 353-363.
11. Scharstein, D. and Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*. **47** (2002) 7-42.
12. Sun, J., Li, Y., Kang, S. B., and Shum, H.-Y.: Symmetric stereo matching for occlusion handling. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. San Diego, CA, USA. (2005) 399-406.
13. Tao, H., Sawhney, H. S., and Kumar, R.: Dynamic depth recovery from multiple synchronized video streams. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. Kauai, Hawaii, USA. (2001)
14. Vedula, S., Baker, S., Rander, P., Collins, R., and Kanade, T.: Three-dimensional scene flow. *Proc. International Conference on Computer Vision*. (1999)
15. Woetzel, J. and Koch, R.: Real-time multi-stereo depth estimation on GPU with approximative discontinuity handling. *Proc. European Conference on Visual Media Production*. London, United Kingdom. (2004)
16. Yang, R. and Pollefeys, M.: Multi-resolution real-time stereo on commodity graphics hardware. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. Madison, WI, USA. (2003) 211-220.
17. Yang, R., Pollefeys, M., and Li, S.: Improved real-time stereo on commodity graphics hardware. *Proc. CVPR Workshop on Real-time 3D Sensors and Their Use*. Washington, DC, USA. (2004)
18. Zhang, L., Curless, B., and Seitz, S. M.: Spacetime stereo: shape recovery for dynamic scenes. *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. Madison, WI, USA. (2003) 367-374.