Real-Time Disparity Map Computation Based On Disparity Space Image

Nadia Baha and Slimane Larabi

Computer Science Department, University of Science and Technology USTHB, Algiers, Algeria
nbahatouzene@usthb.dz, slarabi@usthb.dz

Abstract: The current paper proposes a real-time dense stereo matching algorithm based on a DSI (Disparity Space Image) data structure that can produce accurate disparity map. Our approach divides the disparity map computing into two main steps. The first step deals with the computing of the initial disparity map using DSI method. The second step presents a simple and a fast method to refine the initial disparity map. New strategies and improvements are introduced so that an accurate and fast result can be reached. Experimental results on real data sets were conducted for evaluating the solutions proposed. Also, a comparative evaluation of our method with others methods is presented.

Keywords: DSI (Disparity Space Image), disparity map, Occlusion, Stereovision, Median filter.

1. Introduction
Stereo matching is a problem to find correspondences between two or more input images. It is one of fundamental computer vision problems with a wide range of applications, and hence it has been extensively studied in the computer vision field for decades. Stereo matching consists to find for each point in the left image, its corresponding in the right one. The difference between horizontal distances of these points is the disparity. A disparity map consists of all the possible disparity values in an image. Such a map is basically a representation of the depth of the perceived scene. Therefore, the disparity maps have been used to address efficiently problems such as 3D reconstruction, mobile robot navigation and many other domains.

Despite the simplification brought by the epipolar geometry, the problem of matching remains difficult to solve due to occlusion, luminosity changes between viewpoints and non textured areas. To overcome these difficulties, several methods have been proposed. A state of the art of different existing methods is presented by [1]. In general, stereo algorithms can be categorized into major classes: local methods and global methods. Local algorithms, which are based on correlation, can have very efficient implementations that are suitable for real-time application [2-4]. The central problem of local window-based algorithms is how to determine the size and shape of the aggregation window [5]. That is, a window must be large enough to cover sufficient intensity variation while small enough to avoid crossing depth discontinuities for reliable estimation. This inherent ambiguity causes problems such as noisy disparities in textureless region and blurred object boundaries. Global approaches minimize an overall cost function that involves all the pixels of the image. Field is led to minimize the objective function of energy. Several optimization methods have been proposed such as dynamic programming [6], graph cuts [7], directed anisotropic diffusion [8], belief propagation [9,10] and neural network based approaches [11,12]. A survey for the different approaches can be found in [1,13].

In this paper, we propose a new approach for computing a dense disparity map based on the DSI data structure. As such, stereo images are matched used a DSI composed of gray values representing the cost of all possible matching in stereo images. Our approach divides the matching process into two steps: initial disparity map and refinement of disparity map.

This paper is organized as follows: section 2 presents the stages followed to compute the disparity map. In section 3, experimental results obtained on real images are presented and discussed. Finally, section 4 concludes the paper with some remarks.

2. Initial Disparity Map Estimation
2.1 DSI Computation
Disparity Space Image (DSI) is an explicit representation of the matching space introduced by A. Bobik and S.
Intille [14]. It plays an essential role in the development of the overall matching algorithm which uses the occlusion constraints.

Assuming that the pairs of images are rectified, thus the disparity computation concern two matched points which have the same abscise. For each pixel \( p_1(x_1, y_1) \) in the left image (reference image), the disparity computation will concern all pixels of a window \( W_l \) centered on \( p_1 \). At each pixel \( p_1(x_1, y_1) \) of the \( W_l \), the matched pixel \( p_2(x_2, y_2) \) will appertains to the window \( W_r \) of the right image centred on \( p_1(x_1, y_1) \) (see figure 1). The position of \( W_r \) depends on the disparity \( d \) of the pair \((p_1, p_2)\) which varies from zero to \( d_{\text{max}} \), where \( d_{\text{max}} \) represents the highest disparity value of the stereoscopic images. The relations which bind two matched points \( p_1, p_2 \) of \( W_l \) and \( W_r \) are:

\[
x_i = x + s \cdot d \quad y_i = y, \quad \text{where} \quad s = \{+1, -1\} \quad \text{is a sign chosen so that disparities are always positive.}
\]

To determine the disparity of a given pixel \( p_1(x_i, y_i) \), we calculate for an assumed disparity \( d \) the cost \( DSI^d(p_1) \) of all \( p_i \) of the windows \( W_l \). The size of the window was experimentally chosen to be 7x7.

As the implementation of this method for \( DSI \) computation is time consuming, we propose in the next section the introduction of computation Strategy in order to parallelize the calculation of various costs.

2.2 Computation Strategy

For each pixel \( p_1(x_i, y_i) \) in the left image, the disparity computation will concern all pixels of a window \( W_l \) centered on \( p_1 \). In our case, we used a shiftable window of 7 x7 size (Line x column). In order to increase the computation time of the \( DSI \) method, when we compute the disparity of \( p(i, j+1) \), we perform the operations shown in Figure 2 (a) which avoids recalculating at each time the sum \( AC \) for each pixel in the window \( W_l \). Thus, we compute the disparity for one column at a time rather than the entire window, because the pixels in the neighborhood of \( p(i, j+1) \) are the same as those in the neighborhoods of \( p(i,j) \). The same principle is used to calculate the disparity of the pixel \( p(i+1,j) \). We do the computation for only one line at a time rather than the entire window (see figure 2(b)). By using these operations, the redundancy in computation is completely removed and the computation time has been considerably reduced.

3. Disparity Map Refinement

The resulting disparity map described above is not the optimal one because it still some noise and errors. We propose in the following our disparity map refinement method.
3.1. Refinement Method

We assume that initial disparities of all pixels of the left image are computed. For each pixel $p_i$ of the left image, we first verify if the disparity is dominant in the variable support window $W_i$. If it is the case, this disparity will be considered as the final disparity and does not necessitate any refinement. Otherwise, we do a refinement which consists to select in variable support window $W_i$ the disparity associated to the best score for each pixel. In this step, we apply a vote in order to choose the dominant disparity in the associated $W_i$ using the central pixel $p_i$ and its neighboring pixels. The disparity that will obtain the highest number of confirmations (dominant) will be considered as the new disparity of the central pixel $p_i$. Finally, in order to keep the good trade-off between accuracy and processing time, a simple median filter is applied for smoothing the disparity map.

4. Experimental results

In this section, we describe the experiments conducted to evaluate the performance of the proposed method. The two criteria used for the evaluation are accuracy and computation cost. Many applications not only require accurate disparities, but fast runtime as well. In our case, this work is intended to be used for obstacle detection system of autonomous mobile robot navigation.

4.1 Evaluation in terms of Accuracy

For what concerns accuracy we rely on an evaluation methodology analogous to that adopted on the Middlebury stereo evaluation site [19]. We use four references stereo pairs (Tsukuba, Venus, Teddy and Cones). Figure 3 illustrates the results of the initial disparity maps and final disparity maps obtained for four selected images.

In order to compare the results of different algorithms, we adopt the method similar to that of [1]. Parameters ALL and NOCC are defined according to the Middlebury web site [19]. ALL is the error computed on the whole image and NOCC is the error computed on the whole image excluding the occluded region. Among the quality measures proposed by [1] in their paper we adopted the percentage of bad matching pixels between the computed disparity map $d_C(x,y)$ and the ground truth disparity map $d_T(x,y)$:

$$PBP = \frac{1}{N} \sum (|d_C(x,y) - d_T(x,y)| > \delta_d) \quad (3)$$

Where $\delta_d$ is the error disparity deviating from the ground truth more than 1 pixel.

Table 1 shows the results in term of accuracy obtained by our method on some images available in the Middlebury website[19].

We studied also the influence of window size on the accuracy of the proposed method. Figure 4 illustrates the influence of the window size on the accuracy of our method for each image.

4.2 Evaluation in terms of computation time

Similarly to the evaluation of accuracy, Table 2 shows the processing time obtained by our method when we applied the computation strategy (section 2.2) on the four images pairs with different window size.

The timing tests were performed on a PC Intel Core Duo 2 GHz, 2GB RAM.

4.3 Discussion and comparison

This section presents a comparison between the proposed method and other state-of-art methods. The results obtained by our method are better than some methods reported in the literature [13,16]. Table 3 shows a comparison of stereo vision implementation reported in the literature in terms of computation time. A ranking of each method according to the computation time is shown in the second column.
5. Conclusion

The stereo correspondence problem remains an active area for research. In this paper, we have presented a real-time disparity map estimation algorithm based on the DSI data structure. The disparity map computing process is divided into two main steps. The first one deals with computing the initial disparity map using the DSI structure. The second one presents a simple and fast method to refine the initial disparity map so that an accurate result can be achieved.

Using this method, we approach the results of global methods without sacrificing the simplicity, flexibility and speed of local aggregation methods. As seen in the experiments results, we have performed an experimental investigation in order to assess the impact of the window size and the time optimization techniques. According to the results, we can say that the computation time mainly depends on the image size and the size window.

References

Figure 4. PBP obtained by our method on four image pairs for different window sizes.

Table 1: Accuracy according to the methodology defined by the Middlebury web site.

<table>
<thead>
<tr>
<th>Window</th>
<th>Cones</th>
<th>Teddy</th>
<th>Tsukuba</th>
<th>Venus</th>
<th>Aver. bad Percent.</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL NOCC</td>
<td>ALL NOCC</td>
<td>ALL NOCC</td>
<td>ALL NOCC</td>
<td>ALL NOCC</td>
<td>ALL NOCC</td>
<td>ALL NOCC</td>
</tr>
<tr>
<td>28,9</td>
<td>20,9</td>
<td>32</td>
<td>25,1</td>
<td>15,5</td>
<td>13,7</td>
<td>7,9</td>
</tr>
</tbody>
</table>

Table 2. Processing Time(s): (a) DSI, (b) DSI with Shifting  et  (c) DSI with Shifting + threads

<table>
<thead>
<tr>
<th>Window</th>
<th>Tsukuba</th>
<th>Venus</th>
<th>Teddy</th>
<th>Cones</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) (b) (c)</td>
<td>(a) (b) (c)</td>
<td>(a) (b) (c)</td>
<td>(a) (b) (c)</td>
<td></td>
</tr>
<tr>
<td>3*3</td>
<td>0.531 0.297 0.18</td>
<td>0.75 0.516 0.34</td>
<td>1.922 1.344 0.79</td>
<td>1.922 1.344 0.81</td>
</tr>
<tr>
<td>5*5</td>
<td>0.984 0.377 0.23</td>
<td>1.438 0.734 0.43</td>
<td>4 1.922 1.10</td>
<td>3.968 1.906 1.12</td>
</tr>
<tr>
<td>7*7</td>
<td>1.641 0.484 0.28</td>
<td>2.453 0.875 0.53</td>
<td>7.047 2.438 1.42</td>
<td>7.203 2.438 1.51</td>
</tr>
<tr>
<td>9*9</td>
<td>2.25 0.57 0.34</td>
<td>3.90 1.09 0.62</td>
<td>12.3 2.75 1.71</td>
<td>11.87 2.86 1.7</td>
</tr>
<tr>
<td>11*11</td>
<td>3 0.67 0.4</td>
<td>5.82 1.17 0.71</td>
<td>17.64 3.23 2</td>
<td>17.31 3.45 2.12</td>
</tr>
<tr>
<td>13*13</td>
<td>4.01 0.75 0.43</td>
<td>7.62 1.32 0.81</td>
<td>23.65 3.75 2.4</td>
<td>23.74 3.71 2.4</td>
</tr>
</tbody>
</table>
Table 3. Comparison of stereo vision implementations

<table>
<thead>
<tr>
<th>Author</th>
<th>Time(s)</th>
<th>Algorithm</th>
<th>Image size</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our Method</td>
<td>0.28 s</td>
<td>DSI</td>
<td>Tsukuba</td>
<td>2.0GHz Intel Core Duo</td>
</tr>
<tr>
<td></td>
<td>0.33 s</td>
<td>DSI + Refinement</td>
<td>Tsukuba</td>
<td>2.4 GHz Intel core Duo</td>
</tr>
<tr>
<td>Tombari (2008)[2]</td>
<td>0.2 s</td>
<td>Aggr. Stra. Based on color segm.</td>
<td>Tsukuba</td>
<td>2.4 GHz Intel core Duo</td>
</tr>
<tr>
<td>Gerrits (2006)[4]</td>
<td>2 s</td>
<td>Segment. based</td>
<td>Teddy</td>
<td>2.4 GHz Intel core Duo</td>
</tr>
<tr>
<td>Kim (2005)[6]</td>
<td>4.4 s</td>
<td>Dyn. Prog.</td>
<td>Tsukuba</td>
<td>2.4 GHz Pentium IV</td>
</tr>
<tr>
<td>Veksler (2003)</td>
<td>6 s</td>
<td>Graph-cut</td>
<td>Tsukuba</td>
<td>0.6 GHz, Pentium III</td>
</tr>
<tr>
<td>Tappen (2003)[9]</td>
<td>183 s</td>
<td>Accelerated Belief Prop.</td>
<td>Map</td>
<td>2.4 GHz Pentium IV</td>
</tr>
<tr>
<td>Mattoccia (2009)[3]</td>
<td>13 s</td>
<td>LC Locally Consist.</td>
<td>Tsukuba</td>
<td>2.5 GHz Intel Core Duo</td>
</tr>
<tr>
<td>Yoon (2006)[17]</td>
<td>60 s</td>
<td>Window-based</td>
<td>Tsukuba</td>
<td>AMD 2700</td>
</tr>
<tr>
<td>Vanetti (2009)[11]</td>
<td>100 s</td>
<td>Self org. Map</td>
<td>All images</td>
<td>1.8 GHz AMD Processor</td>
</tr>
<tr>
<td>Venkatesh (2007)[12]</td>
<td>120 s</td>
<td>Self org. Map</td>
<td>256 x 256</td>
<td>1.4 GHz Pentium IV</td>
</tr>
<tr>
<td>Tombari (2007)[15]</td>
<td>33 mn 34 s</td>
<td>Segment Support</td>
<td>Teddy</td>
<td>2.4 GHz Intel core Duo</td>
</tr>
</tbody>
</table>