EFFICIENT STEREO MATCHING USING 4-WAY SCANLINE OPTIMIZATIONS

Searidang Pa

ABSTRACT

Local stereo matching algorithm based on window block matching has low computational cost but have unsatisfactory result while 2-D global optimization algorithm such as graph cut can achieve impressive result but cannot have fast runtime. This paper is inspired the semi-global matching (SGM) algorithm introduced by Hirschmüller, which offers a great balance between accuracy. SGM combines scanline optimizations from different directions. Each scanline optimization encourage the pixels to have similar disparity along the direction. This paper also compares two robust matching costs: Birchfield Tomasi metric and the Census Transform, and shows that matching cost plays a significant role in the accuracy of the algorithm. The algorithm achieves impressive result within 6-8 seconds using only one CPU. The implication of the result is that with a better hardware and software to exploit the parallelizability of the algorithm, one could achieve real-time good stereo matching result.

Index Terms— Stereo matching, Semi-global-matching, Birchfield Tomasi metric, Census Transform

1. INTRODUCTION

Binocular stereo vision has many important application such as robot navigation, 3D reconstruction, and so on. With just two cameras, we can infer depth as the higher the disparity, the closer the point is to the camera. Stereo matching is the most essential and also the most challenging part of stereo vision. In this paper, the problem we want to solve is how to find the disparity between the left and the right images assuming they are rectified, meaning the corresponding point for each point in the left image is on the same line of the right image.

Usually, there are two strategies to stereo matching: local or global [1]. Local algorithms, such as window-based matching using zero-normalized cross correlation or sum of square difference, have very cheap computational cost. However, they either perform badly near objects boundary because the size of the window is too big, or they have a lot of noise because the window is too small. Moreover, they perform especially poorly on textureless and occlusion region as they do not take into consideration the information from the pixels nearby. On the other hand, global algorithms, which use graph cut, loopy belief propagation, and so on achieve great result, but they have high computational costs [3]. Apart from hand crafted algorithms, there are also novel deep learning methods, which can achieve spectacular result efficiently. However, it would require a huge amount of training data, carefully hand-crafted architectures, and a lot of effort and weeks to train the parameters and the hyperparameters [4].

Most stereo matching algorithm can be broken down into 4 steps: matching cost computation, cost aggregation, disparity computation, and disparity refinement [2]. The approach of paper is a 4-way scanline optimization inspired by the Semi-Global Matching algorithm (SGM) introduced by Hirschmüller [3]. The algorithm offers an impressive balance between speed and performance. Instead of trying optimize the whole image like graph cut, the algorithm propagates the information from each point to its neighbor from multiple directions. Moreover, this paper will also look at the two robust matching costs: Census Transform and the Birchfield Tomasi metric on the gradient of images. The information propagation of each direction can be achieved efficiently using dynamic programming [3]. The algorithm is also designed to take advantage of vectorization as best as possible. It also has a fast runtime for images of big resolution by using image pyramid.

2. METHODOLOGY

2.1. Preprocessing

Before the matching cost is computed, the two images are read and converted into grayscale if the images are colored. Despite losing some information, converting grayscale cut down a huge amount computational cost. Moreover, the grayscale images are also normalized by dividing each pixel value by 255. Normalization of the input into a range between 0 and 1 not only makes it easier to tune parameters in the next step of the pipeline, but it also helps remove some noise due to poor contrast or glare.

2.1.1. Image Pyramid Scheme

The algorithm can also achieve fast runtime for images of big resolution by using image pyramid. We first use Gaussian pyramid reduction, and after the completion stereo matching algorithm, we use Gaussian pyramid expansion to get the high resolution disparity map.

2.2. Matching Cost Computation

Our goal in the matching cost computation step is to calculate the cost volume. Each element in the cost volume

matrix is the cost corresponding the pixel of coordinate (x,y) matched at disparity d [5]. Not only is having cost volume a neat way of keeping all of the matching cost in one data structure, but it also helps create a pipeline where we can vectorize the algorithm in the cost aggregation and disparity computation step.

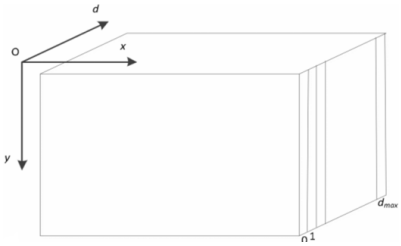

Figure 1: The cost volume has dimension h x w x d_{max} where h, w are the height and width of the image, and d_{max} is the maximum search space

2.2.1. Census Transform

The Census Transform is a local binary encoding where the pixels in a (2n+1) x (2n+1) window are compared with the central pixel. Census Transform is robust as it limits the influence of outliers. Each pixel would have a corresponding "string" of binary value of length $(2n+1)^2$ with a value of 1 when the neighbor pixel has value greater than the central pixel [5].

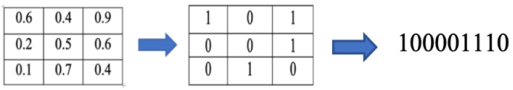

Figure 2: An example of a census transform of 3x3 window

The difference between two CT strings is given by the Hamming distance, how many bits has to be flipped to make the two CT bit strings the same.

To compute the cost volume in a vectorized way, for each row, I, first, perform the census transform for each pixel for the left and the right image and I store the CT bit strings in two matrices. Then, I can take the difference between the left matrix and the right matrix shifted to the left by d to compute the matching cost of the row disparity d. I tried to further vectorize the algorithm by first converting the left and the right image into matrices of dimension h x w x d_{max} x $(2n+1)^2$; however, it turned out to be slower because of the huge memory space cost. Lastly, the output is normalized by dividing each element in the cost volume by $(2n+1)^2$.

2.2.2. Birchfield Tomasi Metric

The intensity of the pixels, in real life, are continuous, but image sensor registers them as discrete integers. This lead to a problem called the sampling problem, where corresponding pixels can be mismatched because to the influence of image sensor. Usually, the influence depends on how much intensity of the surrounding pixels change. The Birchfield Tomasi metric addresses this problem specifically, by not only measuring the difference between the compared two pixels, but also doing a half pixel linear interpolating surrounding the two pixels [6].

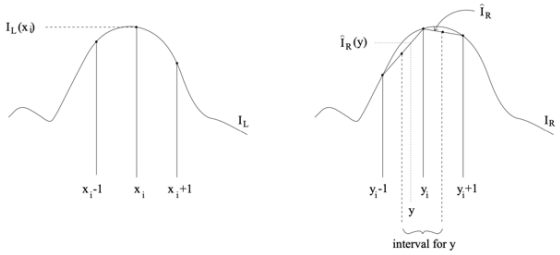

Figure 3: The left and the right curve represent continuous intensity graph of three pixels on the left and the right image respectively

Let $I_L(x_i)$, $I_R(y_i)$ be the intensity of pixel x_i, y_i on the left and the right image respectively. Let $I_R^-(y_i)$ be the linear interpolation between pixel and its left neighbor, and $I_R^+(y_i)$ be the linear interpolation between pixel and its right neighbor for the right image, and vice versa for the left image. We define the minimum difference between the left image and the sampled right image as d_L, and vice versa, d_R for the right image and the sampled left image. Therefore, we calculate $d(x_i, y_i)$, the BT cost of matching two pixel x_i and y_i, as the minimum between d_L and d_R [6].

$$d_L = min\{|I_L(x_i) - I_R(y_i)|, |I_L(x_i) - I_R^-(y_i)|, |I_L(x) - I_R^+(y_i)|\} \quad (1)$$

$$d_R = min\{|I_R(y_i) - I_L(x_i)|, |I_R(y_i) - I_L^-(x_i)|, |I_R(y_i) - I_L^+(x_i)|\} \quad (2)$$

$$d(x_i, y_i) = min(d_L, d_R) \quad (3)$$

To limit the influence of outliers in the cost aggregation step, I cap the BT difference at some constant. Our algorithm computes a sum from three different BT cost on the two images: one on the horizontal gradient (C_1), one on the vertical gradient (C_2), and another one on the intensity (C_3). Since the experiment found the gradients to be a more reliable additional information than the intensity, I combine the three costs in the following way:

$$Cost\_Volume = ((C_1 + C_2)*0.85 + C_3) *0.5 \quad (4)$$

2.2. Cost Aggregation

Even if the matching costs are robust, we would still have problems at textureless surface and occlusions because there is not much features or information those regions can give us. Moreover, there are also other obstacles such as repetitions of features, noises, specular surfaces and so on. From observation, neighboring pixels tend to have similar disparity. Therefore, having a smoothness constraint would help the matching much more robust.

The aggregated cost along a direction can be thought as information passing of the pixel at each disparity level along the direction. More precisely, if we let AgC_1 be the aggregated cost for the direction from left to right, CV be the cost volume from the matching cost computation step, and P be the smooth penalty cost, then for each pixel with coordinate (x,y) and for each disparity level d, we have

$$AgC_1(x, y, d) = CV(x, y, d) + min_{i=1:dmax}(AgC_1(x-1, y, i) + P) \quad (5)$$

The first term on the right hand side is the matching cost for the pixel at disparity level d. The second term is a minimization of the information from the previous pixel at all disparity level with added smoothness cost [3]. Hence, the aggregated cost for each pixel at disparity d equals to matching cost at disparity d plus the information of aggregated cost of the previous pixel of the from all disparity level.

The computation of the aggregated cost for other directions is similar. If AgC_2 is the aggregated cost from right to left, AgC_3 is the aggregated cost from top to bottom, AgC_4 is the aggregated cost from bottom to top, and the final aggregated cost is AgC, then we have

$$AgC_2(x, y, d) = CV(x, y, d) + \min_{i=1:dmax}(AgC_2(x+1, y, i) + P) \quad (6)$$
$$AgC_3(x, y, d) = CV(x, y, d) + \min_{i=1:dmax}(AgC_3(x, y+1, i) + P) \quad (7)$$
$$AgC_4(x, y, d) = CV(x, y, d) + \min_{i=1:dmax}(AgC_4(x, y-1, i) + P) \quad (8)$$
$$AgC = AgC_1+AgC_2+AgC_3+AgC_4 \quad (9)$$

The variable P depends on the type of smooth cost. I consider two different types of smooth cost: capped linear and piecewise constant, both of which have two parameters P_1 and P_2 where $P_1 < P_2$. The piecewise constant cost penalize P_2 if for disparity difference greater than 1 and P_1 if disparity difference is 1. The capped linear cost penalizes more if the disparity difference is greater, but capped at P_2. P_2 exists so that we do not want to penalize neighboring pixels with big disparity difference too high to allow for disparity discontinuity, which happens around the boundary of objects.

2.3. Disparity Computation

After we have obtained the cost aggregation of all 4 directions, to compute the disparity, we compute the disparity map by taking the disparity that correspond to the lowest aggregated cost for each pixel. Since the aggregated cost volume has dimension h x w x d_{max}, for each pixel, we can find the indices of minimum value in the third dimension efficiently due to vectorization.

2.4. Disparity Refinement

I use the median filter to refine the disparity map as it helps remove salt and pepper noise. The median filter works by looking at the n x n region around the point and replace the value of the point with the median in the region.

2.5. Parameter Tuning

To tune the parameters and to evaluate the performance of the stereo algorithm, I use the root-mean-squared (RMS) error between the computed disparity $d_c(x,y)$ and the ground truth disparity $d_T(x,y)$ [2]. I use the Cone and Teddy image pairs and their ground truth from the Middlebury benchmark website. I do not evaluate on the first 50 columns from the left corner as there is no corresponding match on the right image. Let N be the total number of pixels compared, then we have:

$$RMS = \sqrt{\left(\frac{1}{N}\sum_{(x,y)}|d_c(x,y) - d_T(x,y)|^2\right)} \quad (10)$$

Experiments with the sizes of the median filter, 3x3 median filter produces the best result. Moreover, I also find linear smooth cost seem to outperform piecewise constant smooth cost for all values of P_1 and P_2. To tune P_1 and P_2, I use linear smooth cost and a 3x3 median filter.

Table 1: The RMS for different values of P_1 and P_2 for Cone and Teddy image test using BT matching cost

P1	P2	Teddy	Cone	P1	P2	Teddy	Cone
0.02	0.1	5.9749	5.4472	0.04	0.5	5.6514	5.2045
0.02	0.15	5.7545	5.2948	0.04	0.6	5.6082	5.1933
0.02	0.2	5.7016	5.2217	0.04	0.7	5.5979	5.165
0.02	0.25	5.6621	5.1982	0.04	0.8	5.5987	5.1641
0.02	0.3	5.637	5.1654	0.04	0.9	5.6024	5.164
0.02	0.35	5.6306	5.141	0.04	1	5.6187	5.1637
0.02	0.4	5.6275	5.1323	0.05	0.25	5.8187	5.3359
0.02	0.45	5.6317	5.1251	0.05	0.375	5.7401	5.2919
0.02	0.5	5.6463	5.1214	0.05	0.5	5.6997	5.257
0.03	0.15	5.8464	5.3603	0.05	0.625	5.6508	5.2237
0.03	0.225	5.7468	5.2361	0.05	0.75	5.6122	5.2064
0.03	0.3	5.6953	5.2208	0.05	0.875	5.6066	5.1994
0.03	0.375	5.6538	5.1943	0.05	1	5.6055	5.1971
0.03	0.45	5.6065	5.1734	0.05	1.125	5.6089	5.1966
0.03	0.525	5.5875	5.1649	0.05	1.25	5.612	5.1965
0.03	0.6	5.5887	5.1477	0.06	0.3	5.827	5.3557
0.03	0.675	5.6159	5.1432	0.06	0.45	5.7563	5.3102
0.03	0.75	5.6242	5.1413	0.06	0.6	5.7223	5.2669
0.04	0.2	5.8278	5.349	0.06	0.75	5.6681	5.2488
0.04	0.3	5.7407	5.2608	0.06	0.9	5.6392	5.2417
0.04	0.4	5.6991	5.2338	0.06	1.05	5.6343	5.2352

From the table, $P_1 = 0.03$, $P_2 = 0.6$. corresponds to the lowest average RMS out of the two pairs of image test. I conducted a similar experiment and find $P_1 = 0.15$, $P_2 = 1.5$ to be the optimal for the Census Transform matching cost. Of course, we cannot try out every possible value of P_1 and P_2, and it might not be the best for other image pairs, but it seems to be working well with other images.

3. EVALUATION

After tuning the parameters P_1 and P_2 for both matching cost, we can see the role of matching cost on the root mean square error and see which one does better. One big advantage the BT matching cost has over the CT matching cost is that the CT cost lose information around the edge as it is window based. For using image pyramid scheme for images of big resolution, losing information of 2 rows could expand to 8 rows if the reduced factor is 3. Even if we do not evaluate around the edge, the BT cost still outperforms the CT cost.

Table 2: RMS on Teddy and Cone for the Birchfield matching cost (BT) and Census Transform matching cost (CT) with and without evaluating around the edge

Cost Type	Evaluate around edge		Not Evaluating around edge	
	Cone	Teddy	Cone	Teddy
CT	6.5832	6.9674	5.2338	5.6918
BT	5.5887	5.1477	5.1165	5.6058

Therefore, out of all the things that I have tried out, the BT matching cost with linear smooth cost and a 3x3 median filter give the best possible result.

Figure 4: disparity map ground truth, Cone image test, proposed method's disparity map

Figure 5: disparity map ground truth, Teddy image test, proposed method's disparity map

The algorithm takes about 7-10 seconds on a CPU. We could see that the algorithm does well around most of the object's boundary and most uniform regions. There is no streak artifact, which is the disadvantage of 1-D optimization algorithm such as Dynamic Programming.

However, algorithm also has disadvantages that are hard to overcome. It does not perform reliably at occlusion region and at textureless region, as shown in figure 9, as it only encourages all neighboring pixel to have similar disparity and does not care about how similar the pixels are to each other. We can see a smudge around the teddy bear's right arm and around the edge of the house roof. The algorithm can also sometimes mistake small objects as a part of a uniform surface as we can see in the Cone test image where the top half of one of the sticks in the cup disappear. Another source of error is the fact that the output disparity is integer while the ground truth disparity is not as shown in the figure below. The change in disparity level on a uniform surface is not as smooth as it should be. This source of error can be overcome only by sup-pixel interpolation.

The algorithm can also be made more robust by having 8 directions instead of just 4 as in the original paper by Hirschmüller. However, diagonal directions does not allows for vectorization in the cost aggregation step, unlike the vertical and horizontal directions. I have also implemented 8-way optimization, which did improve the output near the object's boundary and textureless region, but the runtime is unreasonable (30s).

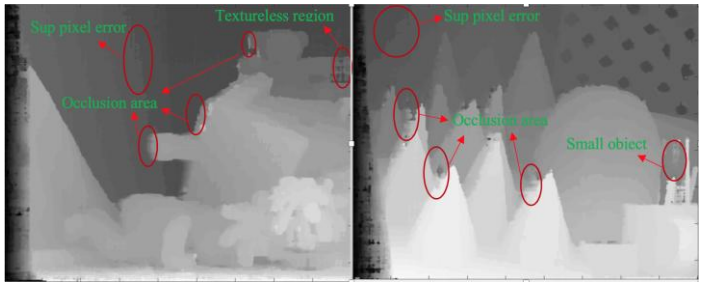

Figure 9: Error analysis of algorithm on Teddy and Cone test image

4. CONCLUSION

All in all, semi-global matching has proved to achieve impressive accuracy with a reasonable runtime. We also see that the importance of robust matching cost. Due to the shortness of time, there are a lot of potential to SGM that I have not exploit to the fullness. SGM is very parallelable as each scanline optimization can be done independently. MATLAB is not a suitable platform to do parallel programing as it has a lot of unnecessary overheads. The use of parallel programming such as GPU programming, multi-thread and so on could even make the algorithm run in real-time. Furthermore, we could have done much more in the disparity refinement step such as sup-pixel interpolation, or left-right check consistency to find inconsistency and fill those pixels using image segmentation techniques.

5. REFERENCE

[1] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[2] D. Scharstein, R. Szeliski, and R. Zabih, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," in *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, Kauai, HI, USA, 2001, pp. 131–140.

[3] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient Deep Learning for Stereo Matching," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 5695–5703.

[4] J. Lim, Y. Kim, and S. Lee, "A census transform-based robust stereo matching under radiometric changes," in *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Jeju, South Korea, 2016, pp. 1–4.

[5] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Bombay, India, 1998, pp. 1073–1080.