Graph-Cut and Belief-Propagation Stereo on Real-World Image Sequences

Joachim Penc[1], Reinhard Klette[2], Tobi Vaudrey[2], and Sandino Morales[2]

[1] Informatics Institute, Goethe University, Frankfurt, Germany
[2] The *.enpeda..* Project, The University of Auckland, New Zealand

Abstract. This paper deals with stereo correspondence search, using graph cuts and belief propagation, for estimating depth maps. The results following different preprocessing steps are evaluated, using the quality of the disparity map. Running times are also investigated. For evaluation purposes, different kinds of images have been used: reference images from the Middlebury Stereo website, synthetic driving scenes, and real-world stereo sequences, either provided by Daimler AG or self-recorded with the research vehicle of the *.enpeda..* project at Auckland University.

1 Introduction

The intention is to reconstruct 3D information out of stereo sequences of 2D images, as available in vision-based driver assistance systems. To perform this goal, one has to match corresponding pixels in the different views to estimate the depth map. Concerning the *.enpeda..* project, this leads us to a proper understanding of the distance of potential obstacles (e.g., other cars, people, or road barriers).

The *depth* of an image pixel is the distance of the corresponding world point from the camera center. A *depth map* is a mapping from pixel values to depth values or depth labels, which are planes parallel to the image plane. Usually these depth labels are represented as grey values (e.g., white for the closest and black for the farthest label).

There exist many approaches which attempt to solve this problem, most of them too slow and/or inaccurate. In this paper we compare graph cut methods – which produce very good results but are quite slow (see [1], [9], [11]) – and belief propagation which has proven to produce good results in reasonable running time (see e.g. [6], [11]). We tested both algorithms with outdoor and indoor images. Furthermore, we pre-processed the images with different filters and compare the different results.

This paper is structured as follows: Section 2 recalls briefly basic definitions for both belief propagation and graph cut stereo. Section 3 specifies the used implementations of both algorithms and the used data sets, and reports then about our results on those data. This is followed by conclusions in Section 4.

2 Depth Map Estimation by Energy Minimization

Our goal is to compute pixel correspondences in a given pair of images, which then determines depth. This refers to the *traditional stereo problem:* for each pixel (location) p in the reference image find the corresponding pixel p' in the matching image. This problem formulization is equal to the so called *pixel labelling problem:* given a set $\mathcal{L}$ of labels assign each pixel p to a label $f_p \in \mathcal{L}$.

2.1 Energy Function

It is commonly assumed that the image pairs have the following properties: *photo-consistency* (a corresponding pixel should have the same intensity value) and *piecewise smoothness* (neighboring pixels are likely to have the same depth value). Regarding these assumptions, an *energy function* E takes into consideration the error when giving the pixels a certain label:

$$E(f) = \sum_{p} D_p(f_p) + \sum_{q \in A(p)} V(f_p, f_q) \tag{1}$$

The first addend or data term is the sum of all penalties $D_p(f_p)$ for assigning (in f) a label $f_p = f(p)$ to a pixel p. The second addend or smoothness term is the sum of all penalties $V(f_p, f_q)$ [often simplified by using $V(f_p - f_q)$] for assigning (in f) labels f_q to pixels $q \in A(p)$ which are adjacent to p.

To minimize the error when labeling the pixels one has to find a labelling f which minimizes the energy function E. Unfortunately finding a global minimum is an NP-complete problem (see [7]), but belief propagation or the graph cut method can still find a good local minimum in a strong sense.

2.2 Belief Propagation Method

A belief propagation algorithm [3] passes messages (the "belief") around in a 4-adjacency image grid. Message updates are in iterations; messages are passed on in parallel, from a pixel to all of its 4-adjacent neighbours. At one iteration step, each pixel of the adjacency graph computes its message based on the information it had at the end of the previous iteration step, and sends its (new) message to all the adjacent pixels in parallel.

Assume that $m^t_{q \to p}$ is the message, send from pixel q to adjacent pixel p at iteration t. For $l \in L$, let

$$m^t_{q \to p}(l) = \min_{h \in L} \left(V(h - l) + D_q(h) + \sum_{s \in A(q) \setminus p} m^{t-1}_{s \to q}(h) \right) \tag{2}$$

l denotes a possible label at p, and h runs through L and is again just a possible label at q. We accumulate at p a one-dimensional array of all messages received from all $q \in A(p)$, and this contains at its position $l \in L$ the following:

$$D_p(l) + \sum_{q \in A(p)} m^t_{q \to p}(l) \tag{3}$$

Besides $D_p(l)$, we also have the sum of all the received message values for $l \in L$. Instead of passing on 1D arrays of length card($\mathcal{L}$), a belief propagation algorithm uses card($\mathcal{L}$) *message boards* of the size of the images, one board for each label l (disparity). At the end of an iteration t, the disparity with minimum costs is selected as being the result for a pixel p.

2.3 Graph Cut Method

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V} = \{v_1, \ldots, v_n, s, t\}$ and edges $\mathcal{E} = \{e_1, \ldots, e_m\}$ with capacities $\{c_1, \ldots, c_m\} \subset \mathbb{N}_0^+$; an (s,t)-*cut* is a partition of $\mathcal{V}$ into subsets S and T with $s \in S$ and $t \in T$. The cost C_{cut} of an (s,t)-cut is the sum of all capacities crossing S and T:

$$C_{\text{cut}} = \sum_{u \in S, v \in T, (u,v) \in \mathcal{E}} C(u,v) \tag{4}$$

A *minimum* (s,t)-*cut* is therefore an (s,t)-cut with minimum costs.

A *flow* φ in the graph $\mathcal{G}$ is defined as follows:

(1) for each edge $e_j \in \mathcal{E}$: $0 \leq \varphi(e_j) \leq c_j$
(2) for each vertex $\tilde{v} \in \mathcal{V} \setminus \{s,t\}$: $\sum_i \varphi(\tilde{v}, v_i) = \sum_j \varphi(v_j, \tilde{v})$
(3) for the source-node s: $\sum_i \varphi(s, v_i) \geq \sum_j \varphi(v_j, s)$
(4) for the target-node t: $\sum_i \varphi(t, v_i) \leq \sum_j \varphi(v_j, t)$

The value Φ of the flow φ is calculated as follows: $\Phi = \sum_i \varphi(s, v_i) - \sum_j \varphi(v_j, s)$
A *maximum flow* $\Phi_{\max}$ is a flow whose value cannot be increased any more.

Ford and Fulkerson (see [4]) proved that the calculation of a maximum flow is equivalent to the calculation of a minimum cut. As there are more efficient algorithms for solving the maximum flow problem than for the minimum cut we refer to the calculation of a maximum flow instead. The algorithms have about $\mathcal{O}(n^4)$ running time but have in practice only about $\mathcal{O}(n^3)$.

Expansion-Move Algorithm. Here we briefly introduce the *expansion-move algorithm*, which is a greedy algorithm with low-order polynomial complexity. However, in practice, it runs in near-linear time to perform the energy minimization via graph cuts.

Given labelings f, f' an α-expansion move from f means

$$f'(p) \neq f(p) \Longrightarrow f'(p) = \alpha \tag{5}$$

for every pixel p in the image.

The following pseudo-code shows the expansion-move algorithm. For a label α compute the lowest energy expansion move from the current labelling to a new labelling if its energy is lower:

(1) start with arbitrary labelling f
(2) success := false

(3) FOR EACH label $\alpha \in \mathcal{L}$
  (3.1) *find* $\hat{f} = \arg\min E(f')$ *among* f' *with one* α*-expansion of* f
  (3.2) IF $E(\hat{f}) < E(f)$ THEN $f := \hat{f}$ AND success := true
(4) IF success == 1 THEN GOTO 2)
(5) RETURN f

The crucial step of this algorithm is 3.1 where the minimum search is performed via graph cuts (in practice – via maximum flow calculation). The pixels of the image represent the vertices of a graph which are 4-connected (n-links). Two additional terminal-nodes s and t are connected to every node (pixel) by an edge; these are referred to as t-links. In Step 3.1, each pixel p is assigned its old label f_p or the new label α. The nodes of the graph (i.e., the pixels) are assigned to partition S (old label) or T (new label), and the capacities of the cutting edges represent the energy for this partitioning (labelling) by assigning data and smoothness terms to the t- and n-links, respectively. A detailed discussion of this method can be found in [1] and [8].

3 Experiments and Results

The evaluation experiments have been performed using a very recent graph cut implementation from V. Kolmogorov and R. Zabih[3] which can detect occlusions quite well; and a modified coarse-to-fine belief propagation algorithm of Felzenszwalb and Huttenlocher[4] implemented by S. Guan for [6]. His implementation focused on more reliable (and time-efficient) matching; therefore using the max-product, 4-adjacency, truncated quadratic cost function, the red-black speed-up method, and coarse-to-fine processing. The parameters for the belief propagation algorithm have been set appropriate to S. Guan's experiments (LEVELS = 5, $\text{DISC}_K = 11$, $\text{DATA}_K = 30$, $\lambda = 0.5$), and the parameters for the graph cut algorithm were chosen automatically by some heuristic procedure of the algorithm which produced the best results. The algorithms were tested on a MacBook Pro with an Intel Core2 Duo CPU (2.50 GHz) and 2 GB memory.

3.1 Data Sets

For the first experiment we used the *Tsukuba* stereo pair from the Middlebury stereo website[5] (see Figure 1). The benefits of using images from the Middlebury website are that they were recorded under ideal light conditions with minimal shadowing, noise or artifacts and – most important – there is a ground truth provided, so that one can easily compare the results of a certain stereo technique with the real depths.

[3] See http://www.adastral.ucl.ac.uk/vladkolm/software/match-v3.3.src.tar.gz
[4] See http://people.cs.uchicago.edu/ pff/bp for original sources.
[5] http://vision.middlebury.edu/stereo

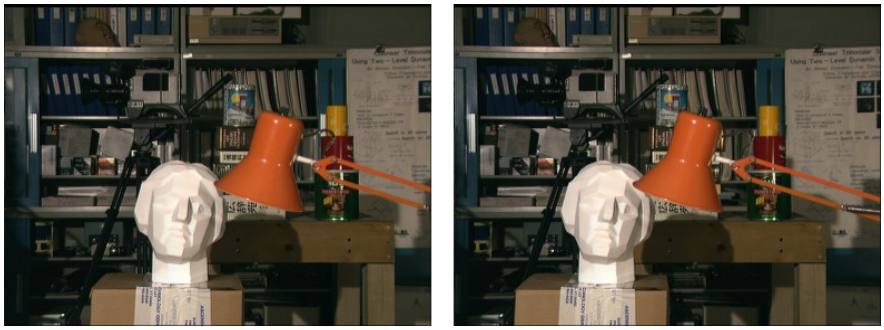

Fig. 1. The *Tsukuba* image pair from the Middlebury Stereo website.

For the second experiment we used one stereo pair of the synthetic driving sequence form [2] (see Figure 2). The sequence consists of 640×480 gray-scaled stereo pairs, and any two views taken out of that sequence have been constructed using exactly the same virtual stereo camera. There are perfect lighting

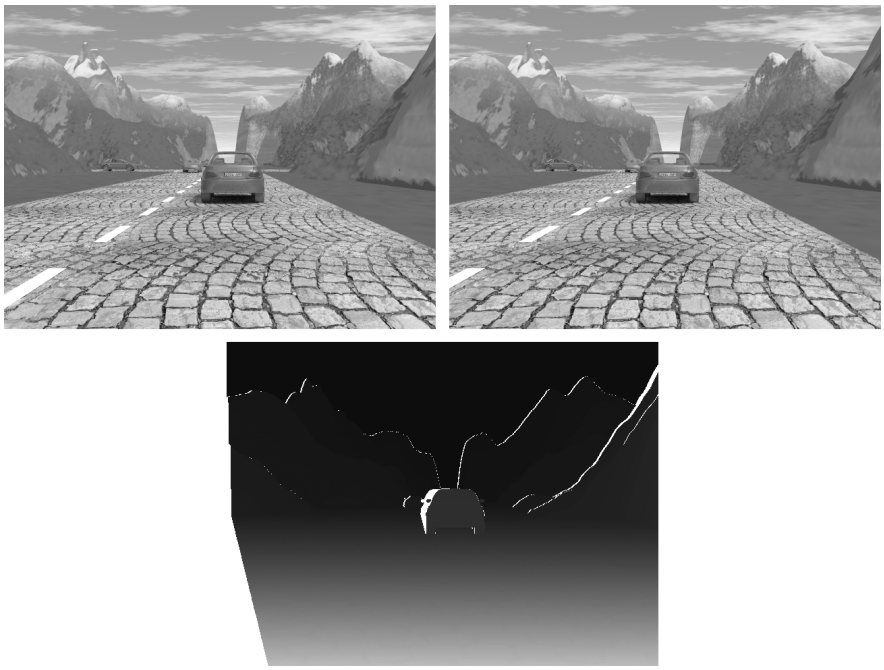

Fig. 2. Top: one pair of images out of 100 pairs of a synthetic driving sequence. Bottom: ground truth for this pair.

Fig. 3. Real-world driving scene images provided by Daimler AG.

Fig. 4. Self-recorded real-world driving scene images gathered by using *HAKA1*.

conditions, but there are shadows, specular reflections, and window transparency. As with the Middlebury stereo images, there is also ground truth provided.

In the next experiments we used a pair of images out of real-world night vision driving scenes provided by Daimler AG, Germany.[6] The night vision (grey-scale) images (see Figure 3) have a resolution of 640×480. Each pair of images (as usual when dealing with real-world data) comes with the expected common problems: artifacts, different light conditions, shadowing, and so forth.

Further experiments have been done using self-recorded images; see examples of those in Figure 4. The images were captured at day with the research vehicle *HAKA1*, from the *.enpeda..* group, using two Point Grey FireFly cameras. The camera system provides 640×480 pixel resolution (hardware-) synchronized grey-scale image pairs, at 30 frames per second.

[6] Downloadable at http://www.mi.auckland.ac.nz

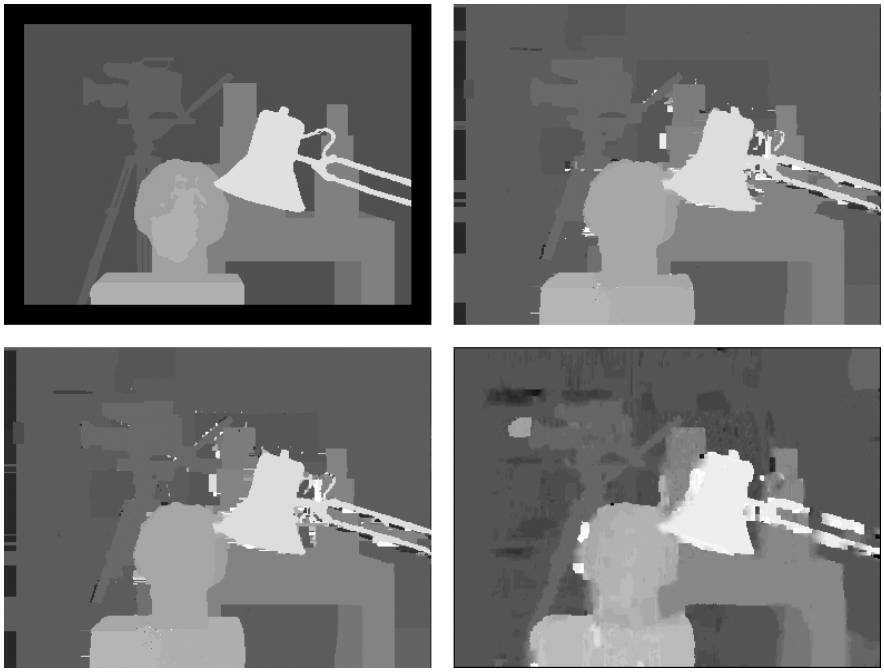

Fig. 5. Upper left: ground truth. Upper right: 1 iteration of the GC algorithm (running time t=30 s). Lower left: 3 iterations of GC (t=58 s). Lower right: BP results (t=9 s).

3.2 Tsukuba

According to [1], the graph cut algorithm only needs a few iterations, and already shows good results after only 1 iteration. The first tests with the *Tsukuba* images were therefore testing this statement and comparing it with the ground truth image. Figure 5 shows the results running 1 and 3 iterations of the graph cut (GC) algorithm, compared to the results when using belief propagation (BP). As one can see here, it is sufficient to run only 1 iteration of the GC algorithm, and the results are quite close to the provided ground truth. The BP algorithm was faster, but the results look very noisy.

We tested the effect of various pre-filtering methods on a nearly perfectly recorded image, the *Tsukuba* scene. The chosen pre-filtering steps were: contrast-enhancing, highpass-filtering, Gaussian-filtering and Sobel-filtering. All filtering resulted in worse quality depth maps and a slight increase in running time (obviously there is no need to filter a perfect image).

3.3 Synthetic Driving Scene

The following experiments were done using an image pair out of the synthetic driving scene. The same tests, as with the *Tsukuba* images were performed, producing similar results. Figure 6 shows some selected results (including the root mean square error (RMS), compared to the ground truth, as defined in

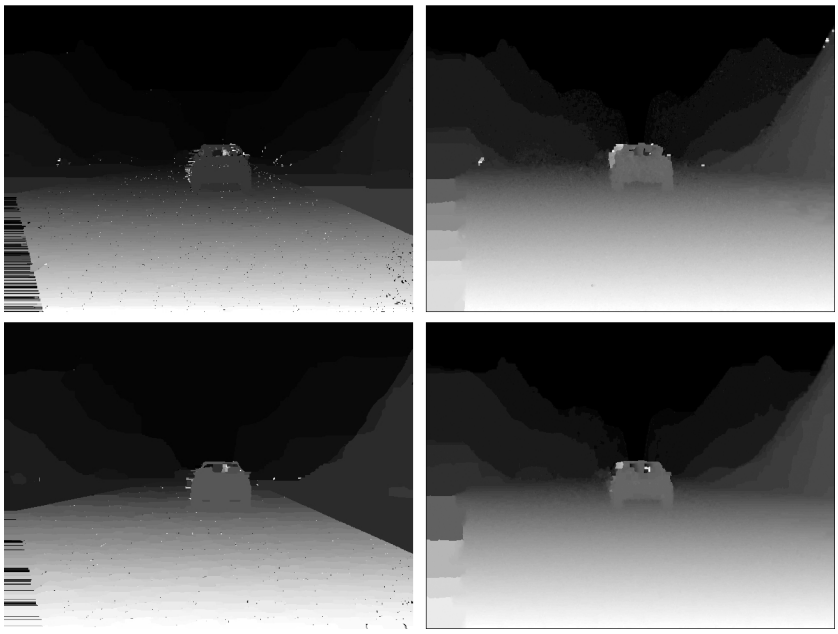

Fig. 6. Upper row: results with Sobel images for GC (left, t=175 s, RMS=4.2) and BP (right, t=47 s, RMS=7.2). Lower row: results with median filtered images for GC (left, t=175 s, RMS=4.3) and BP (right, t=45 s, RMS=7.4)

[11]). Sobel filtering and smoothing of the image (by using a 3×3 median filter) still produced no better results.

3.4 Real Scene (Daimler AG)

The previous results showed us that when dealing with nearly perfectly recorded images or good synthetic ones – good contrast and grey value distribution – any pre-filtering method will result in worse results than dealing with the original images. We will now have a look at real-world driving scenes, and if pre-filtering will improve the depth map results or not. This time we have to measure the quality subjectively, because there is no exact ground truth available. Therefore, in Figure 7 one can see the original left image to get an opinion on what the ground truth could look like and compare this to the results of the stereo algorithms. The first evaluation was testing the influence of the number of iterations for GC (comparing 1 and 3). Also, when initializing GC with the depth map from the previous image in the sequence the results got better.

As one can see in Figure 7, BP produces very large errors and GC looks quite good – only in the lower section where large reflections occurred is where GC fails as well. As S. Guan did, we decided to crop the images and cut out the lower 131 pixel rows to get acceptable results for BP (see Figure 8).

This produces smaller images, allowing faster run-times for the algorithms. Furthermore, the disturbance of the reflections is greatly reduced We applied Sobel pre-filtering (see also Figure 8) to see how the real-world images behave.

Fig. 7. Upper row: original left image (left) and results of 1 iteration GC (t=184 s). Lower row: results of BP (left, t=34 s) and using the previously calculated depth map as input for GC (right, t=182 s).

By using Sobel images, the algorithms produce more depth values but contain more noise. This noise was reduced when the images were smoothed with a 3×3 median filter before applying the Sobel filter. However the results of BP still had large errors. Using contrast enhancement additionally showed the best results for GC and it also improved the BP results: they were much better and had less error. The very best results for both GC and BP could be achieved when dealing with contrast enhancement, followed by smoothing, then Sobel filtering the images (see Figure 9).

Fig. 8. Left: cropped left input image of the Daimler scene. Right: cropped Sobel image.

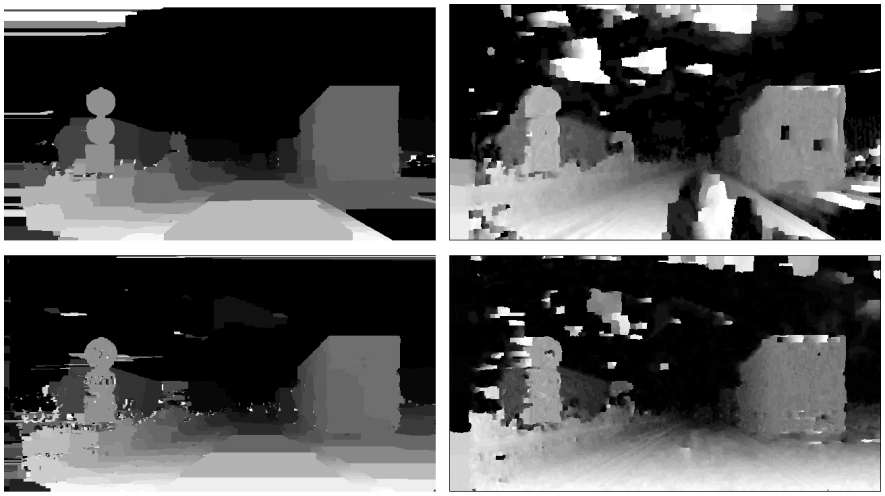

Fig. 9. Upper row: results using enhanced contrast images as input for GC (left, t=69 s) and BP (right, t=16 s). Lower row: results using contrast enhancement before smoothing and Sobel filtering for GC (left, t=63 s) and BP (right, t=17 s). - See Figure 2 for ground truth image.

3.5 Real Scene (self-recorded)

Further experiments have been performed using a self-recorded daytime scene, with our research vehicle *HAKA1*. The chosen image pair is a bit overexposed, which happens quite often when driving during daylight. When applying GC or BP to these images, they both fail because of the missing texture in the upper part of the image. The result using dynamic programming [10], which is robust to various noise and illumination issues and (but produces a so called *scanline effect*), indicates how a depth map should look. When applying Sobel images, contrast enhancement or pre-smoothing before using BP the results don't get better, but with GC they do. So once again GC showed much better behaviour, even with high-occlusion real-world images. Some results are shown in Figure 10.

4 Conclusions

The tests with the synthetic and real-world images, as well as with a reference image pair from Middlebury stereo website, showed clearly that BP is the faster algorithm. Nevertheless the lower 150 pixel problem area in the real-world driving scene images, where heavy differences in the two views occur, produces very bad results with BP, which could be reduced using Sobel preprocessing. The slower GC algorithm produces much better and accurate depth maps – and even produces the only reasonable results when applied to (bad quality) real-world data – but it takes about 4 to 5 times the running time in comparison to BP.

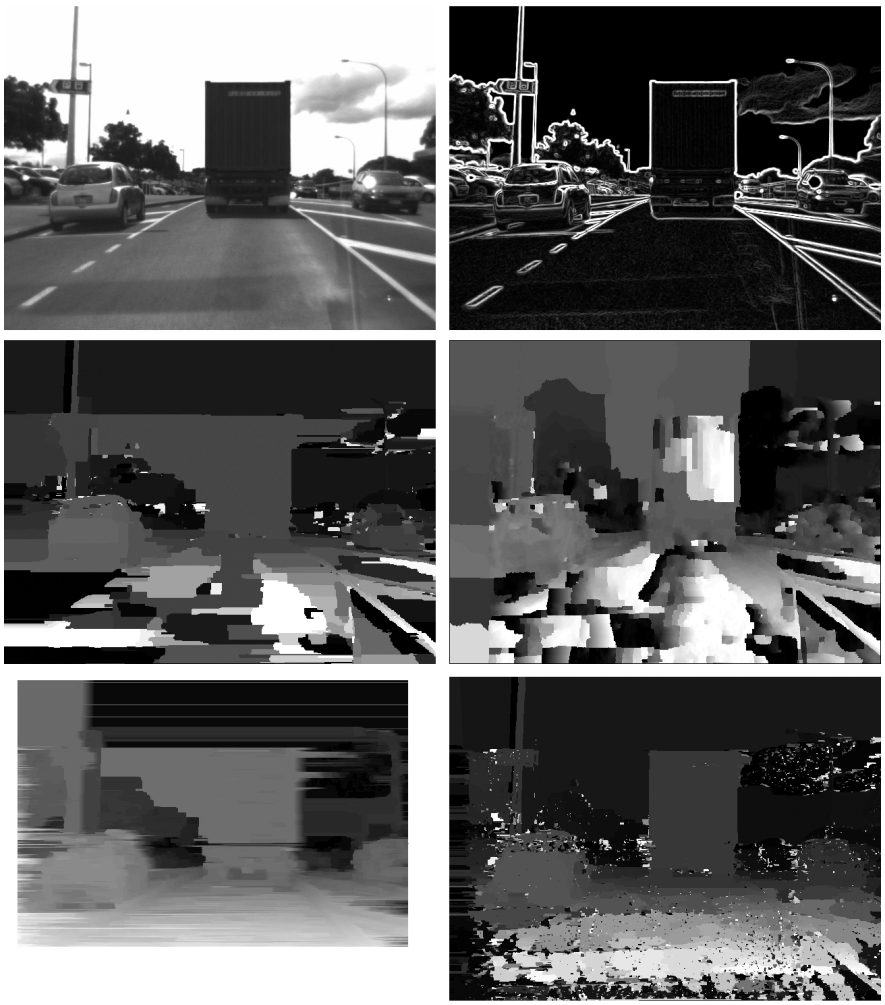

Fig. 10. Upper row: original left image recorded with *HAKA1* (left) and Sobel image (right). Middle row: results for GC (left) and BP (right). Lower row: results for dynamic programming (left) and for GC using Sobel images (right).

References

1. Boykov, Y., Veksler, O., and Zabih, R.: Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis Machine Intelligence*, **23**:1222–1239 (2001)
2. *.enpeda..* image sequence analysis test site (EISATS). [Online]. Available: http://www.mi.auckland.ac.nz/EISATS/

3. Felzenszwalb, P. F., Huttenlocher, D.P.: Efficient belief propagation for early vision. *Int. J. Computer Vision*, **70**:41–54 (2006)
4. Ford, L. R., and Fulkerson, D. R.: Flows in networks. Technical report R-375-PR, US Air Force Project RAND (1962)
5. Guan, S. and Klette, R.: Belief-propagation on edge images for stereo analysis of image sequences. In Proc. *Robot Vision*, LNCS 4931, pages 291 – 302 (2008)
6. Guan, S., Klette, R., and Woo, Y. W.: Belief propagation for stereo analysis of night-vision sequences. In Proc. *PSIVT*, LNCS 5414, pages 932 – 943 (2009)
7. Kolmogorov, V., and Zabih, R.: Multi-camera scene reconstruction via graph cuts. In Proc. *Europ. Conf. Computer Vision*, pages 82 – 96 (2002)
8. Kolmogorov, V., and Zabih, R.: What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Analysis Machine Intelligence*, **26**:65–81 (2004)
9. Kolmogorov, V. and Zabih, R.: Graph cut algorithms for binocular stereo with occlusions. *Handbook of Mathematical Models in Computer Vision* (N. Paragios, Y. Chen (Editor), and O. Faugeras, editors), pages 423–438 (2006)
10. Ohta, Y., and Kanade, T.: Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Trans. Pattern Analysis Machine Intelligence*, **7**:139–154 (1985)
11. Scharstein, D. and Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Computer Vision*, **47**:7 – 42 (2002)
12. Vineet, V., and Narayanan, P. J.: CUDA cuts: Fast graph cuts on the GPU. In Proc. *CVPRW*, DOI: 10.1109/CVPRW.2008.4563095 (2008)