2011 IEEE/RSJ International Conference on
Intelligent Robots and Systems
September 25-30, 2011. San Francisco, CA, USA

Stereo obstacle detection in challenging environments: the VIAC experience

Alberto Broggi, Michele Buzzoni, Mirko Felisa and Paolo Zani
VisLab – Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma, ITALY
http://www.vislab.it
{broggi,buzzoni,felisa,zani}@vislab.it

***Abstract*— Obstacle detection by means of stereo-vision is a fundamental task in computer vision, which has spurred a lot of research over the years, especially in the field of vehicular robotics. The information provided by this class of algorithms is used both in driving assistance systems and in autonomous vehicles, so the quality of the results and the processing times become critical, as detection failures or delays can have serious consequences. The obstacle detection system presented in this paper has been extensively tested during VIAC, the VisLab Intercontinental Autonomous Challenge [1], [2], which has offered a unique chance to face a number of different scenarios along the roads of two continents, in a variety of conditions; data collected during the expedition has also become a reference benchmark for further algorithm improvements.**

I. INTRODUCTION

Fast methods to obtain accurate and dense depth maps are becoming increasingly common [3], [4], [5], and the problem of extracting useful information from such a big amount of data is of great interest. Even without taking into account algorithmic complexity, giving a definition of an obstacle is not a trivial task: an option is to determine a dominant ground surface [6] and consider as an obstacle anything sticking out of it; anyway, there are situations where this approach fails (e.g. very cluttered environments with no clearly visible road area, or lateral slopes). If no assumption can be done, as it was the case during the VIAC expedition, it is safer to consider the the ego-vehicle mechanics (e.g. height, width, maximum traversable slope) to identify areas that cannot be crossed. The downside of this approach is its computational weight, which limits the number of points that can be handled: nevertheless, this paper presents a parallel processing scheme which allows to run at 10 Hz on a commercial hardware platform.

A. Hardware configuration

Images used for stereo reconstruction are acquired at a resolution of 752×480 pixels by a pair of IEEE1394-A cameras equipped with 4 mm, 1/3" lenses; synchronization is guaranteed by a hardware trigger signal. The sensors are mounted right below the solar panel, as can be seen in Fig. 1-a. Processing is performed on a Mini-ITX board with an Intel® Core™2 Quad Q9100 @ 2.26 GHz processor and 4 GB RAM located on the back of the van (Fig. 1-b).

(a)

(b)

Fig. 1. Highlighted in red, hardware components of the stereo system: (a) the forward-looking stereo cameras, and (b) the processing unit in the back.

B. The expedition

During VIAC the vehicles crossed a number diverse environments, ranging from country motorways in Hungary to busy downtowns in Russia, from the 2900 m Lanquan mountain pass to highway construction areas in China (Fig. 2); the weather also changed dramatically, from the hot summer of Ukraine (with an average temperature of 45 °C) to the cold September in Russia, the pouring rain of China, and snow on the mountain passes. Not all of what the vehicles had to face could be anticipated, but that was also the purpose of the test: to design an algorithm as robust as possible, evaluate its performance, and improve it afterwards using the data collected in the most critical scenarios.

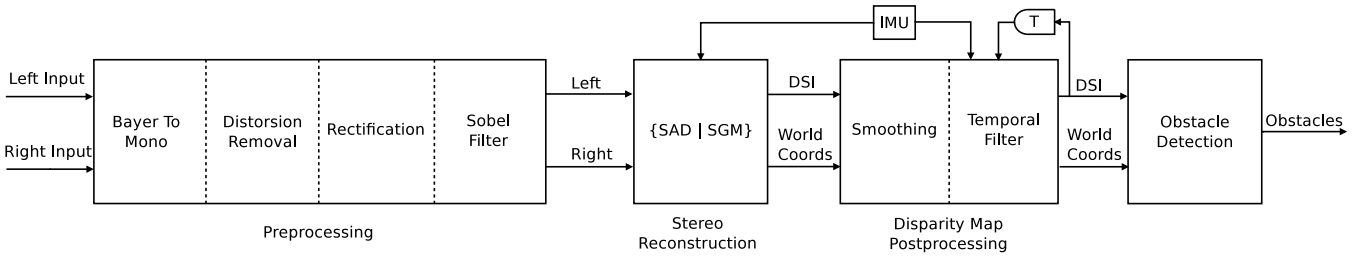


Fig. 3. System architecture.

Fig. 2. The path followed by the vehicles during VIAC

II. ALGORITHMS

Processing happens following the steps presented in Fig. 3:

- *low-level processing* — Bayer-patterned images acquired by the cameras are converted to gray-scale, then lens distortion is corrected [7], and the stereo pair is rectified. A vertical Sobel filtering is used to improve the subsequent matching phase [8];
- *disparity map generation* — stereo reconstruction is carried out; both correlation-based and Semi-Global Matching (SGM) [3] approaches have been tested, as detailed in Sec. II-A;
- *disparity map post-processing* — two filters are applied to further enhance the map quality;
- *obstacle detection* — actual obstacle detection takes place.

A. Stereo reconstruction

Given the tight development schedule, Disparity Space Image (DSI) generation was implemented exploiting the already available window-based SAD correlation technique described in [9]. As reported in Sec. III, the outcome was satisfactory; nevertheless, in the months following VIAC an efficient implementation of the SGM algorithm has been devised, producing even better results.

In order to generate a Disparity Space Image D the Semi-Global Matching algorithm performs an energy minimization step. The energy function $E(D)$ that has to be globally minimized consists of two terms: the pixel-wise matching cost $E_{data}(D)$ and the smoothness constraint $E_{smooth}(D)$:

$$E(D) = E_{data}(D) + E_{smooth}(D) \tag{1}$$

The $E_{data}(D)$ term is the sum of all pixel matching costs C for the disparities of D:

$$E_{data}(D) = \sum_{\mathbf{p}} C(\mathbf{p}, D_{\mathbf{p}}) \tag{2}$$

Instead of using mutual information as the pixel-wise matching function, as it is done in [3], the Hamming distance of the Census transform of a 5×5 window cropped around $\mathbf{p}$ has been exploited, since it provides similar results [12], [4] while reducing the overall computational burden.

The E_{smooth} term adds a small penalty P_1 to all pixels $\mathbf{q}$ in the neighborhood $N\mathbf{p}$ of $\mathbf{p}$, for which the disparity varies from $\mathbf{p}$ by one, and a higher penalty P_2 if the difference is greater:

$$\begin{aligned} E_{smooth} = &\sum_{\mathbf{q} \in N\mathbf{p}} P_1 \mathbf{T}[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] + \\ &\sum_{\mathbf{q} \in N\mathbf{p}} P_2 \mathbf{T}[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1] \end{aligned} \tag{3}$$

with

$$\mathbf{T}[x] = \begin{cases} 1 \text{ if } x \text{ is true} \\ 0 \text{ otherwise} \end{cases} \tag{4}$$

The global minimization of $E(D)$ is a NP complete problem, that SGM approximates by computing the values of $E(D)$ along 1D paths from 8 directions towards each pixel using dynamic programming. The costs $L'_{\mathbf{r}}$ of each path $\mathbf{r}$ are aggregated as described in Eq. 6 for each pixel $\mathbf{p}$ and disparity d:

$$\begin{aligned} L'_{\mathbf{r}}(\mathbf{p}, d) = C(\mathbf{p}, d) + \min(& L'_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d), \\ & L'_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d-1) + P_1, L'_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, d+1) + P_1, \\ & \min_i L'_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, i) + P_2) \end{aligned} \tag{5}$$

The final disparity value for each pixel is then determined by a winner-takes-all strategy applied to the values of $L'_{\mathbf{r}}$. To further improve the results sub-pixel interpolation is performed as well as a median filter and a left-right consistency check.

In order to fully exploit the parallel processing capabilities of modern multi-core CPUs and reach real-time frame rates,

TABLE I

SGM PERFORMANCE METRICS

Algorithm	Hardware platform	Image size $[px]$	Max disparity $[px]$	Time $[ms]$	Disparity bandwidth $[s^{-1} \times 10^6]$
Gehrig ECVW10 [4]	Intel® Core™ i7 975 EX @ 3.3 GHz	640×320	128	224	117
Hirschmüller ISVC10 [10]	NVIDIA® GeForce™ 8800 Ultra	640×480	128	238	165
		320×240	64	76	64
Gehrig ICVS09 [11]	Xilinx® Virtex-4 FX140	$2 \times 340 \times 200$	64	40	218
Nedevschi IV10 [5]	NVIDIA® GeForce™ GTX 200	512×383	56	19	578
this paper	Intel® Core™ i7 920 @ 3.20 GHz	640×320	128	27	970

a multi-threaded, SIMD processing scheme has been devised. The most time-consuming step of the SGM algorithm is path accumulation, since it must be performed for each pixel, disparity, and path: to speed up the processing, for each path direction the pixels are split into several independent slices that are processed in parallel; moreover only the accumulated value is saved into memory, while temporary data needed for incremental processing is kept into the CPU registers[1]. Finally, when computing the results of Eq. 6 the Intel® SSE instruction set is also used, outputting 16 disparity values at a time.

Table I presents a comparison of some state of the art SGM implementations, as reported in the respective papers. Testing conditions differ significantly, so the last column contains the achieved disparity bandwidth measured in number of values computed per second; the hardware platforms are also heterogeneous, both in terms of architecture and performance (e.g. the graphics card used in [5] is about twice as fast as the one used in [10]). The approach presented in this paper runs about eight times faster than the top-scoring CPU implementation on similar hardware, and twice as fast as the best GPU implementation, which uses just four accumulation paths and a single P coefficient when computing E_{smooth}.

B. DSI post-processing

The generated disparity map is post-processed to fix possible spurious values; this is especially useful when using correlation-based stereo, where local minima are more likely to introduce noise.

A first filter analyzes a 3×3 window around each DSI element, checking that its disparity value is close to a sufficient number of neighbors, and marks it as invalid in case this condition is not satisfied. After this step has been performed, invalid pixels whose neighborhood has a variance lower than a fixed threshold are assigned the average value of the surrounding elements.

The second filter uses IMU information to compute the vehicle trajectory between the previous and current frame acquisition, and checks the corresponding disparity maps for consistency. At time T each point $p_i(u,v,d)$ of the depth map D_T is projected into the corresponding world point $p_w(x_w, y_w, z_w)$ [13], which is in turn projected back into DSI coordinates using virtual cameras corresponding to the position and orientation of the stereo rig at time $T-1$ with respect to the current reference system; this process generates the depth map D^T_{T-1}, which can be directly compared to D_{T-1}, computed at $T-1$. Each pixel of D^T_{T-1} is analyzed, and considered valid only if at least one of the pixels of D_{T-1} within a 3×3 window has a disparity value close enough. As it can be seen in Fig. 4-b,c this kind of filter is effective at canceling random noise, which is unlikely to be consistent across frames, but by its very design it leads to suppression of fast-moving objects; therefore the removal threshold must be tuned taking into account the maximum relative speeds of obstacles that need to be detected.

C. Obstacle detection

Since no assumptions could be made on the road infrastructure quality and the kind of traffic to expect, the obstacle detection algorithm design followed the approach first described in [14]. This technique defines a criterion to cluster points into obstacles based on their layout in space and on the physical characteristics of the ego-vehicle, namely its height H_{max}, the minimum relevant obstacle height H_{min} and the maximum traversable slope θ_{max}. Given two points p_{w1} and p_{w2} these constraints are used to define a truncated cone in space, with the vertex corresponding to p_{w1}, as depicted in Fig. 5: if p_{w2} falls within the cone it is labeled as an obstacle, and p_{w1} and p_{w2} are said to be *compatible*.

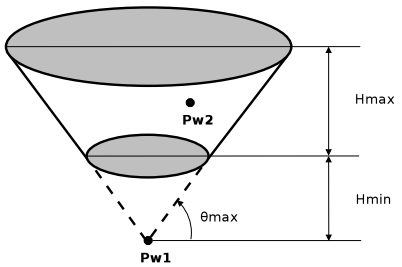
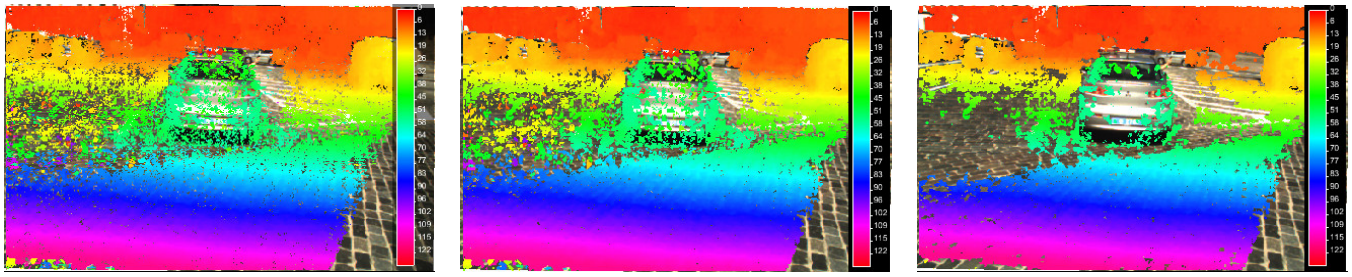


Fig. 5. Compatibility criterion used during the clustering phase. Point p_{w2} is considered an obstacle since its position relative to p_{w1} leads to a path which is not traversable by the ego-vehicle.

To reduce the number of comparisons to perform, compatibility checks are carried out in image coordinates, rather than in the world: this is done by projecting the truncated cone back onto the image using camera calibration information, thus (approximately) obtaining a trapezium, and checking whether the constraint is valid for the disparity points contained within it. Iterating on the disparity space image from

[1]Disparity search ranges of up to 128 are supported; over this threshold not enough XMM CPU registers are available



(a) Input DSI. (b) Smoothing filter. (c) Motion filter.

Fig. 4. Disparity filters; colors encode disparity values. The smoothing stage removes isolated spurious values and fills the holes in uniform areas, while the motion filter removes inconsistent pixels, like the purple ones in the middle of the image. In this case filtering succeeds at suppressing most of the noise due to an instant misalignment of the stereo rig caused by a bump in the road.

bottom to top, and from left to right, it is possible to correctly cluster the whole data set. During the clustering phase each region is assigned a unique label; to ensure that a single obstacle does not get split into multiple adjacent regions the following strategy is adopted: let l_{w1} and l_{w2} be the labels of points p_{w1} and p_{w2} respectively, then

- if $l_{w1} \neq undefined$ and $l_{w2} = undefined$, $l_{w2} \leftarrow l_{w1}$
- else if $l_{w1} = undefined$ and $l_{w2} \neq undefined$, $l_{w1} \leftarrow l_{w2}$
- else if $l_{w1} \neq l_{w2}$ all the points with label l_{w2}, plus p_{w2}, are relabeled as l_{w1}

The final result is saved into two dual representations:

- an image, having the same resolution of the depth map, where each pixel value corresponds to a label ID;
- a vector of regions, each containing the list of its points.

Even using DSI coordinates the computational load associated with the clustering phase is considerable, especially when using the dense depth map produced by the SGM matching algorithm. In order to fully exploit the parallel processing capabilities of the target hardware platform, a multi-resolution, multi-threaded analysis scheme was devised.

The original DSI, along with its associated 3D world points vector, is split into N images, each containing only points corresponding to a predefined range of distances on the X axis, as illustrated in Fig. 6.

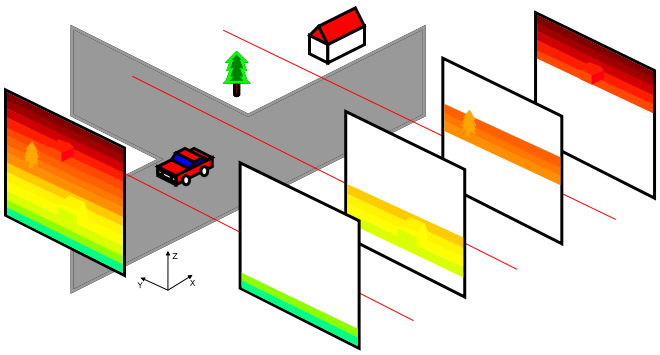


Fig. 6. DSI partitioning. Separate depth maps are created, each corresponding to a slice of world points, then the maps are analyzed in parallel on different CPU cores.

The depth maps are then sub-sampled in order to further reduce the number of comparisons to perform: ideally constant spatial resolution can be achieved [15], but in practice it is more convenient to fix a single resolution for each stripe; using the hardware setup described in Sec. I-A, the partitioning has been experimentally chosen as follows:

- 0 – 5 m $\rightarrow 4\times$ sub-sampling
- 5 – 15 m $\rightarrow 2\times$ sub-sampling
- 15 – 30 m $\rightarrow$ full resolution
- 30 m – $\infty \rightarrow$ full resolution

After this step each slice is processed independently in parallel, leading to N sets of labels; the sets are then copied back on a single, full-resolution map, upscaling the sub-sampled clusters, while checking that the extrapolated points are similar enough to the original ones, in order to avoid introducing blocky artifacts near depth discontinuities.

A final pass is still needed to ensure that all labels are properly merged, since an obstacle spanning across a slice boundary is still split in two distinct labels, as shown in Fig. 7. Each point in each cluster is checked to determine whether the value within the labels image corresponds to the one saved within the regions vector, and in case they are different, the two points sets are merged together and the labels image is updated accordingly.

III. RESULTS

This section presents both algorithm outputs and processing times, using some of the images acquired during VIAC as a reference.

A. Performance

In order to produce a quantitative estimation of the algorithm performance two of the three metrics presented in [16] have been computed on a 1000 frames sequence acquired in the downtown of Kiev (Ukraine) on August, 5, 2010 approximately at 14:00 local time. While this data represents only a tiny fraction of the whole trip it is quite representative of typical driving conditions in city traffic.

The first metric is the false correspondences ratio $m_{fc} = N_{fc}/N$, where N_{fc} are the number of DSI points inside a volume defined by the road plane, the width and height of the ego-vehicle and a safety time gap of 1 s multiplied by the ego-velocity, and N the total number of valid DSI points.

(a) Input DSI. (b) Labels before merging. (c) Merged labels.

Fig. 7. Obstacles labeling on a steep road when driving uphill. First points are clustered according to their relative pose, in parallel, on multiple CPU cores, then the labels are merged and filtered to produce the final obstacles list. Note that despite the slope the vehicle is facing, no false detection is triggered even without performing any explicit modeling of the ground.

The second metric is the leader vehicle lateral position measurement, defined as $m_{lp} = |lp_{measure} - lp_{groundtruth}|$, with the ground truth being generated by direct LIDAR measurements of the preceding vehicle.

Table II contains the computed values, both for the correlation-based and SGM stereo matching algorithms; as a reference, the values reported in [16] have also been inserted in the table, although they refer to a different dataset.

TABLE II
PERFORMANCE METRICS

	This paper		Steingrube ICVS09 [16]	
Algorithm	m_{fc} [%]	m_{lp} [m]	m_{fc} [%]	m_{lp} [m]
Correlation Stereo	3.0	0.15	1.02	0.13
SGM	0.0153	0.11	0.98	0.11

The results are very similar; only the m_{fc} value for the SGM reconstruction case changes significantly, probably because of the different amount of data the test has been run on.

B. Processing time

The algorithms have been benchmarked using the sequence described in Sec. III-A on two different hardware platforms: the first is a desktop PC equipped with an Intel® Core™ i7 920 @ 3.20 GHz processor and 6 GB RAM, while the second is the industrial PC installed on the VIAC vehicles, described in Sec. I-A. Table III contains the processing times breakdown of the whole algorithm pipeline on both systems, with a side-by-side comparison of the SGM and correlation-based stereo approaches. Images have been scaled to a resolution of 500×320 pixels, keeping the total processing time under 100 ms in all cases but one.

Some examples of the algorithm outputs are presented in Fig. 8.

TABLE III
PROCESSING TIMES

	Processing Time [ms]			
Algorithm step	Intel® Core™ i7 920		Intel® Core™ 2 Quad Q9100	
	SGM	SAD	SGM	SAD
Preproc.	2.9	2.9	4.9	5.0
DSI	21.5	5.8	60.2	7.8
DSI filt.	2.8	2.8	6.5	6.7
Obstacle det.	32.1	25.2	59.2	54.1
Total	59.3	36.7	130.8	73.6

IV. CONCLUSIONS AND FUTURE WORK

The obstacle detection system presented in this paper was successfully employed during VIAC, effectively negotiating a wide variety of scenarios; the approach was proven effective even in presence of steep climbs and off-road areas. At the end of the expedition the algorithm has been further enhanced by the introduction of a high performance SGM implementation for the depth mapping stage. Calibration still remains a critical issue, since obstacle detection heavily relies on having correct world coordinates, although the filters described in Sec. II-B greatly reduce the errors introduced by vibrations.

In the future, to obtain a better obstacles segmentation, optical flow could be employed; anyway, an efficient implementation needs to be found in order to keep processing time under control even on conventional hardware.

V. ACKNOWLEDGMENTS

This work has been supported by the European Research Council (ERC) within the Open intelligent systems for Future Autonomous Vehicles (OFAV) Advanced Investigators Grant. Thanks to the cooperation with Piaggio®, the vehicles selected for VIAC are Piaggio® Porter Electric Power.

REFERENCES

[1] A. Broggi, L. Bombini, C. Stefano, P. Cerri, and R. I. Fedriga, "Sensing requirements for a 13,000 km intercontinental autonomous drive," in *Procs. IEEE Intelligent Vehicles Symposium 2010*, La Jolla, CA, USA, June 2010, pp. 500–505.

[2] VIAC http://viac.vislab.it.

[3] H. Hirschmüller, "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information," in *Intl. Conf. on Computer Vision and Pattern Recognition*, vol. 2. San Diego, CA, USA: IEEE Computer Society, June 2005, pp. 807–814.

[4] S. Gehrig and C. Rabe, "Real-time semi-global matching on the cpu," in *ECVW10*, 2010, pp. 85–92.

[5] I. Haller, C. Pantilie, F. Oniga, and S. Nedevschi, "Real-time semi-global dense stereo solution with improved sub-pixel accuracy," in *Procs. IEEE Intelligent Vehicles Symposium 2010*, San Diego, CA, USA, June 2010, pp. 369–376.

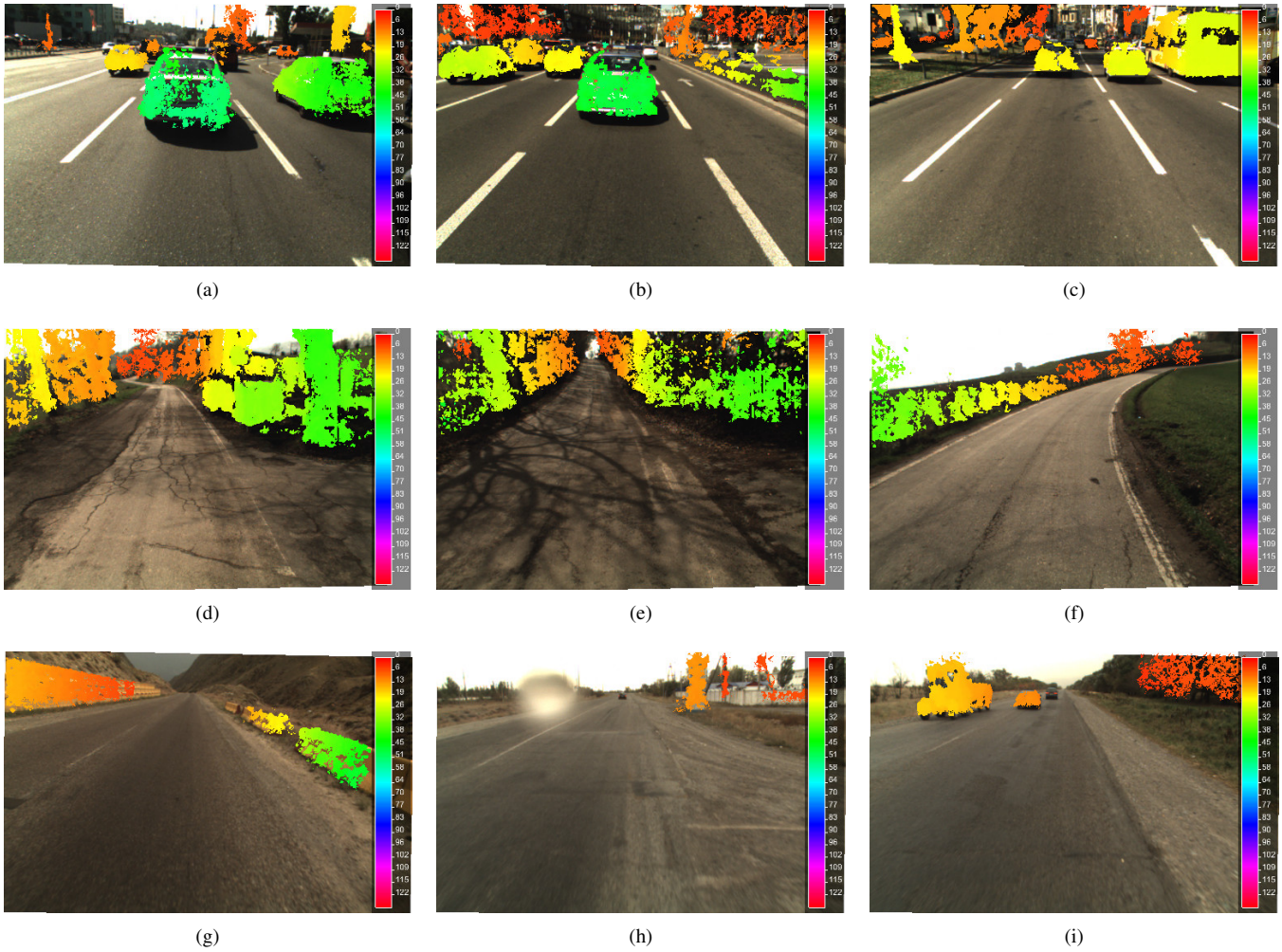


Fig. 8. Some sample outputs in different scenarios. (a) – (c) a busy motorway in Kiev, (d) – (f) country roads with woods and uphill sections, (g) a deserted mountain motorway in Kazakhstan, (h) a raindrop on the right camera and (h) an upcoming tractor.

[6] A. Wedel, H. Badino, C. Rabe, H. Loose, U. Franke, and D. Cremers, "B-spline modeling of road surfaces with an application to free-space estimation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 4, pp. 572 –583, 2009.

[7] F. Devernay and O. Faugeras, "Straight lines have to be straight," *Machine Vision and Applications*, vol. 13, pp. 14–24, 2001.

[8] H. Hirschmüller and S. Gehrig, "Stereo matching in the presence of sub-pixel calibration errors," in *Intl. Conf. on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009, pp. 437–444.

[9] M. Felisa and P. Zani, "Incremental Disparity Space Image computation for automotive applications," in *Procs. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, St.Louis, Missouri, USA, Oct. 2009.

[10] H. Hirschmüller and I. Ernst, "Mutual information based semi-global stereo matching on the gpu." in *ISVC (1) 08*, 2008, pp. 228–239.

[11] S. K. Gehrig, F. Eberli, and T. Meyer, "A real-time low-power stereo vision engine using semi-global matching," in *Proceedings of the 7th International Conference on Computer Vision Systems: Computer Vision Systems*, ser. ICVS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 134–143.

[12] H. Hirschmuller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *PAMI*, vol. 31, no. 9, pp. 1582–1599, September 2009.

[13] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[14] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Auton. Robots*, vol. 18, no. 1, pp. 81–102, 2005.

[15] S. Nedevschi, R. Danescu, R. Schmidt, and T. Graf, "High accuracy stereovision system for far distance obstacle detection," in *Procs. IEEE Intelligent Vehicles Symposium 2004*, Parma, Italy, June 2004.

[16] P. Steingrube, S. K. Gehrig, and U. Franke, "Performance evaluation of stereo algorithms for automotive applications," in *Proceedings of the 7th International Conference on Computer Vision Systems*, ser. ICVS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 285–294.