2013 IEEE Intelligent Vehicles Symposium (IV)
June 23-26, 2013, Gold Coast, Australia

Terrain Mapping for Off-road Autonomous Ground Vehicles Using Rational B-Spline Surfaces and Stereo Vision

Alberto Broggi[1], Elena Cardarelli[1], Stefano Cattani[1] and Mario Sabbatelli[1]

Abstract— **Autonomous Ground Vehicles designed for extreme environments (e.g mining, constructions, defense, exploration applications) require a reliable estimation of terrain traversability, in terms of both *terrain slope* and *obstacles* presence. In this paper we present a new technique to build, in real time and only from a 3D points cloud, a dense terrain elevation map able to: 1) provide slope estimation; 2) provide a reference for segmenting points into terrain's *inliers* and *outliers*, to be then used for obstacles detection. The points cloud is first smartly sampled into a 2.5 grid map, then samples are fitted into a rational B-Spline surface by means of re-weighted least square fitting and equalization. To meet an extensive range of extreme off-road scenarios, no assumptions on vehicle pose are made and no road infrastructure or a-priori knowledge about terrain appearance and shape is required. The algorithm runs in real time; it has been tested on one of VisLab's AGVs using a modified SGM-based stereo system as 3D data source.**

I. INTRODUCTION

Reliable perception of terrain slope and terrain traversability is a key-feature for any unmanned ground vehicle designed for extreme environments. This is often achieved processing 3D points clouds coming from high-quality dense depth maps. The majority of the approaches project depth information into digital elevation maps [1] or into various types of Cartesian grid maps, containing cells of uniform size. Cells typically store information about the corresponding world portions, depending on the algorithm approach and type of sensor used. The so called 2D grid maps just contains *occupancy* information (traversable, not traversable); these are useful for basic obstacle avoidance on flat terrains, based on simple range sensors. When cells store also height information, we talk about 2.5D grid maps [2]. More complex grid maps embed full 3D information using adjacent stack of cells [3], or octrees connected cubes [4], and are able to represent objects at multiple heights located at the same range and azimut.

Elevation maps, as well as cubes grids, do not provide an immediate classification of their cells as belonging or not the terrain. Some authors [5] enrich cells' geometric characteristics with visual information, like color and texture, to understand if a given cell is part of the terrain, applying machine learning techniques on the basis of a priori knowledge. When it is possible to use assumptions on road model and on vehicle pose, successful solutions of geometry-based obstacle/terrain segmentation have been proposed,

[1] A. Broggi, E. Cardarelli, S. Cattani and M. Sabbatelli are with VisLab-Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Parma {cattani, smario, cardar, broggi}@vislab.it

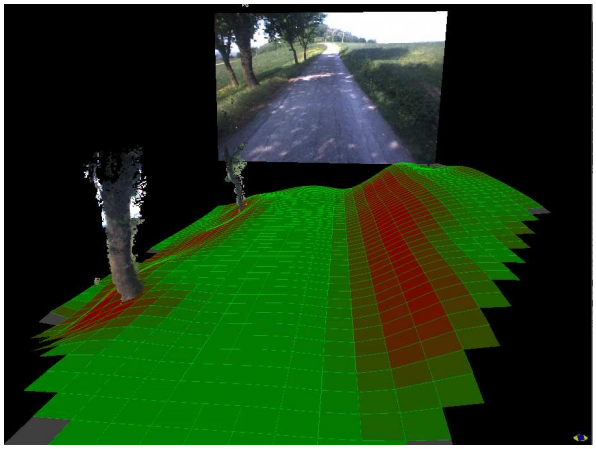


Fig. 1. The image shows a terrain slope with a synthetic grid, obstacles appears with their original color and enriched with 3D info, while a picture of the scene is also shown to compare the results.

in some cases working in image coordinates by using v-disparity space [6]; in other cases applying RANSAC fitting to segment the 3D points in road inliers and outliers [7].

In our previous work [8] we presented a geometry based technique able to segment 3D data points into terrain inliers and outliers, without any constrains on vehicle pose and surface roughness. The segmentation is made on the basis of *traversable terrain* concept: *any surface where the vehicle can drive on*. In other words, *any surface not too rough and steep for the vehicle capabilities*. The algorithm block diagram is shown in Fig. 2. It is pipeline oriented, where each block can be implemented independently, as far as the data flow is respected.

In [8] Ants Colony Optimizations (ACO) was used to fit points into a certain number of longitudinal and lateral 2D models, then fused together to obtain a 3D Cartesian model of the terrain. ACO works very satisfactorily, but it is computationally demanding (as well as RANSAC based techniques), since it requires many iterations for each model to reach optimal results. Moreover, subdividing the world in several 2D models, not constrained with each other but just merged at the final stage, leads to a noisy and, occasionally, unstable results.

In this paper we present a new technique to build, in real time, the map of the *traversable terrain*, fitting 3D points into a rational B-Spline surfaces [9] [10]. Oppositely to the ACO based approach, made of a mesh of several 2D models, B-Splines provide a full 3D elevation map model,

978-1-4673-2754-1/13/$31.00 ©2013 IEEE

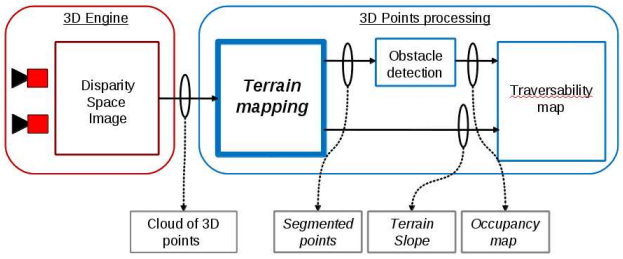


Fig. 2. The block diagram of our traversability mapping algorithm: first a 3D point cloud is generated; then, a fully derivable surface model of the terrain is build, able to segment points into terrain inliers and outliers; finally, slope and occupancy information are fused together to build a traversability cost map. This paper focuses mainly on terrain map creation.

where each sample represents a constrain for the others, improving robustness; at the same time, B-Splines are locally controllable by control points, allowing the same ACO's sensitivity to localized terrain slopes. Finally, the proposed iterative re-weighted fitting method performs terrain's inliers and outliers segmentation with very few iteration, compared to ACO and RANSAC, ensuring computational efficiency in a wider range of scenarios.

Example of the final results is shown in Fig. 1.

II. ALGORITHM OVERVIEW

The approach can be described as a 5 steps procedure:

1) *3D world points cloud*. A data set of 3D world points (x_i, y_i, z_i) is generated. The data set must provide enough points to support the desired resolution over the area of interest. Implementation details in Section III.
2) *2.5D grid sampling*. Each 3D point is projected onto its corresponding cell, that will contain every point that belongs to its volume, regardless they represent terrain, an object, or spurious noise. The number of cells and their size are defined by the region of interest and the required resolution. Resulting map has $m_0 \times m_1$ Cartesian cells, each one characterized by (x, y) coordinates and z height, hereinafter called $p(x_i, y_i, Z_i)$. Details on sampling method are in Section IV.
3) *B-Spline fitting*. The goal is to extract the *main terrain surface* from the sampled points grid $p(x_i, y_i, Z_i)$: a 3D surface representing the terrain where it is possible to drive safely and where objects stand on. Section V shows how repeated least square fitting and equalization of sampled points, with different patterns of B-Spline control points, order, and weights, leads to the desired segmentation in an efficient way.
4) *Slope estimation*. Once the terrain surface is available in B-Spline form, terrain slope can be easily computed for any (x, y) location contained into the interest area just by derivation.
5) *Obstacle detection and Traversability cost*. Points (x_i, y_i, z_i) not belonging to the traversable terrain are clustered together into obstacles candidates. Then, a Kalman filter is applied to prune false positives. Traversability cost for each (x, y) location is finally computed as a function of the corresponding slope and occupancy information: the steeper the slope, the higher the cost; similarly, the more occupied the terrain portion, the higher the cost of its traversability.

This paper focuses on terrain mapping only: steps 1 to 4.

III. 3D POINTS ENGINE

The 3D points engine is based on the processing of the Disparity Space Image (DSI). To perform the stereo reconstruction a modified Semi Global Matching (SGM) algorithm [11] with a multi-resolution analysis scheme [12] has been implemented. In order to reduce computational weight, a multi-threaded SIMD processing scheme has been devised, exploiting the parallel processing capabilities of the hardware platform.

Fig. 3. Dense Stereo 3D point cloud.

The engine has been extensively tested during VIAC, the VisLab Intercontinental Autonomous Challenge [13], in a variety of scenarios and in different conditions.

IV. GRID QUANTIZATION

The main focus of the sampling processing, applied among the 3D world points obtained by the disparity map, is simplifying the traversable terrain reconstruction, highlighting, in the 3D cloud, all points belonging to the terrain slope and, at the same time, attenuating the spurious noise contribution. Moreover, the grid quantization allows to reduce the problem of computational complexity, transforming the dense stereo depth map into a sparse grid representation.

Each world point (x_j, y_j, z_j) is projected, according to its position, on a $m_0 \times m_1$ 2.5D grid, whose cells are used to locally accumulate the points. As shown in figure 4, for each cell c_i all 3D points, belonging to its volume, are condensate in a single point $p(x_i, y_i, Z_i)$, with x_i and y_i equal to the cell centre coordinates (p_x, p_y), and Z_i calculated as:

$$Z_i = max(min(z_j), mean(z_j) - \sigma) \quad (1)$$

where:

- $min(z_j)$ is the height of the lower world point in c_i;
- $mean(z_j)$ and σ are, respectively, the average and the standard deviation of the 3D points heights belonging to the volume of each cell c_i.

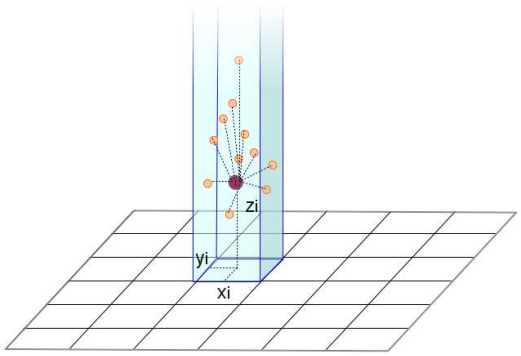


Fig. 4. 3D world points quantization processing.

When none of 3D points belongs to a given cell, its corresponding Z_i is computed averaging valid neighbours' heights. If no valid neighbours are available, cells are marked as *INVALID*, and $Z_i := 0$. Section V explains how B-Spline fitting handles invalid cells.

This mapping, through the local evaluation of the 3D points distribution, allows to maximize the contribution of the 3D points characterized by low height values, that are likely to represent the terrain slope. The determination of the mapped Z_i is more influenced by low z_j values, regardless of whether the corresponding world points heights vary in a small or in a large range. The standard deviation evaluation is used to attenuate the outliers contribution. Moreover all mapped heights are maintained in the range defined, locally, by the highest and lowest world point.

V. B-SPLINE FITTING

A rational B-spline tensor product surface of order d_0 and d_1 is a $R^2 \rightarrow R$ function defined as follows:

$$\mathbf{S}_{\mathbf{R},\mathbf{Q}}(x,y) = \sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} \frac{\mathbf{R}_{i_0,i_1} N_{i_0,d_0}(x) N_{i_1,d_1}(y) \mathbf{Q}_{i_0,i_1}}{\mathbf{R}_{i_0,i_1} N_{i_0,d_0}(x) N_{i_1,d_1}(y)} \tag{2}$$

where: $\mathbf{Q}$ is a 2-dimensional array of $(n_0+1) \times (n_1+1)$ control points; d_0 and d_1 are Spline's degrees along the two axes, with $1 \leq d_0 \leq n_0$ and $1 \leq d_1 \leq n_1$; $\mathbf{R}$ is the corresponding control points' weights matrix.

The set of $N_{i,d}(t)$ are the Spline's basis functions:

$$N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t)$$

with

$$N_{i,0}(t) = \begin{cases} 1 & t_i \leq t < t_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

where t is a non-decreasing sequence of scalars t_i for $0 \leq i \leq n_{0,1} + d_{0,1} + 1$ known as *knot vector* [9]. In this paper, the vectors (x, y) are world coordinates, while $\mathbf{S}_{\mathbf{R},\mathbf{Q}}(x, y)$ are the corresponding terrain's height z.

A. B-Spline surface fitting

Control points $\mathbf{Q}_{i_0,i_1}$ are typically obtained by *Least Squares Fitting*, from a set of $m_0 \times m_1$ sample points $\mathbf{P}$ with known z. In particular, our fitting module starts with *Weighted Least Squares Curve Fitting* [9] along the X world coordinate, where to each sample is assigned a proper amount of influence $\mathbf{W}(z_i)$ over the parameter estimates. Then the resulting control points are fitted across the Y direction, to compute the final $\mathbf{Q}_{i_0,i_1}$ control points matrix. As mentioned, here the weights refer to *sample points*, instead of control points, and are used to increase or decrease the influence of a given sample on the final square error.

B. Terrain estimation

Basically the algorithm consists of a repeated least-squares B-Spline fitting, using surfaces with a number of control points increasing iteration after iteration, and where control points' weights and samples' weights, at a given iteration, are adapted on the basis of the previous surface.

The idea is to fit sample points with a very simple B-Spline, not weighed and not rational, with very few control points, in order to understand the principal terrain slope and shape; then, the closer a sample point is to this surface, the higher its weights will be at the next iteration. Moreover, to speedup the procedure, samples are also *equalized*: they are moved towards the surface, the higher the distance, the higher the equalization. This procedure ends when the maximum number of control points n_{max} (always $\leq min(m_0, m_1)$) is reached or when no point has been equalized in the last step. The algorithm V-B.1 summarize the process in pseudo-code.

Algorithm V-B.1 Iterative least-squares fitting of $\mathbf{S}$

```
W ← identity_matrix(m0 × m1)
R ← constant_matrix(1, (n0 + 1) × (n1 + 1))
procedure Terrain_mapping(P)
 do
    P_prev ← P
    Q ← spline_fitting(P, W, d0, d1, δn0, δn1)
    ē ← linear_mean(‖P − S_R,Q‖)
    W ← compute_weights(P, S_R,Q, e)
    R ← remap_weights(W, δn0, δn1)
    P ← equalization_filter(P, S_R,Q, ē)
    δni ← δni + 1
 while (δni < n_max_i,  i = 0, 1) ∧ (‖P_prev − P‖ < ϵ)
 Q ← spline_fitting(P, W, d, n0, n1)
end
```

As mentioned $\mathbf{P}$ is a $(m_0 \times m_1)$ samples matrix containing the Z_i values of each sample, while $\mathbf{R}$ and $\mathbf{W}$ are control points and sample points weights matrices respectively. The $\mathbf{R}$ are computed downsampling and interpolating the larger $\mathbf{W}$ matrix.

1) Weight function: Function `compute_weights` in V-B.1 assigns to each sample point a fitting weight according to the following formula:

$$\mathbf{W}(Z_i) = 1 + \left(\frac{|\Delta z|}{th(\overline{e})} \right)^{-\alpha} \tag{3}$$

where $th > 0$ is the weighting threshold, proportional to the current average fitting error, while $\alpha \geq 1$ is the weighting speed, proportional to the current iteration number. Note how $\mathbf{W}(Z_i)$ is always greater than 1.

2) Equalization: The `equalization_filter` in V-B.1 moves sample points towards the last computed surface, according to this logic:

$$Z_i = Z_i - \Delta z \cdot (1 + \frac{1}{\mathbf{W}(Z_i)})$$

Actually not all sample points are equalized: valid points (see Sec. IV) with $\Delta z < 0$ are not modified. This is because our goal is to find a good *terrain* surface estimation, that must always be made of the *lowest* visible 3D points. Under this point of view, it is counterproductive to rise points to the current approximated surface. Note how adjustments are always smaller than Δz, hence points never overshoot the current surface.

In Fig. 5 it is shown the comparison between a simple B-Spline surface and the proposed iterative method, applied on the same samples.

VI. TESTING

The input 3D points cloud includes everything is visible in the images: terrain, obstacles standing on the terrain, sky, etc. The goal of our algorithm is to fit the terrain surface, removing all the remaining points (outliers). An easy scenario is when all the 3D points belong to the terrain (e.g. a flat road free of obstacles): in this case it is enough to fit the whole 3D point cloud into a simple spline. Now imagine the same scene, but with a pedestrian on the ground: a well-performing terrain estimation algorithm should detect the very same terrain surface as before, *without being affected by the presence of the object*. On the basis of this concept, the assessment procedure is the following:

1) select a scene where the terrain is perfectly estimated by the algorithm; e.g. a flat road;
2) add an *artificial obstacle*, of given width and height, and estimate again the terrain surface;
3) compare the two surfaces, calculating the maximum deviation from the original surface.

Fig. 6 shows, for each artificial obstacles' width and against obstacles' height, the maximum surface deviation in percentage of obstacles' height. We can notice that:

- deviations are more dependent on obstacles' size rather than height; in particular they are dependent on size *compared to the area of interest's size*: a tall and thin obstacles (e.g a pole or a pedestrian), will be easily cut off, leading to small differences between the two surfaces; a wide and short obstacle will be less effectively separated from the terrain below;
- under a terrain estimation point of view, the above concept can be rephrased: those parts of the terrain characterized by low height/width ratio turn out in high deviations with respect to a flat plane; i.e. obstacle are more likely to be included into terrain;
- with few control points the surface tents to be rigid, i.e. less affected by the presence of an obstacle, regardless of its height; at the same time, a rigid surface results to be less accurate in estimating rough terrains, since deviations never get close to obstacle height;

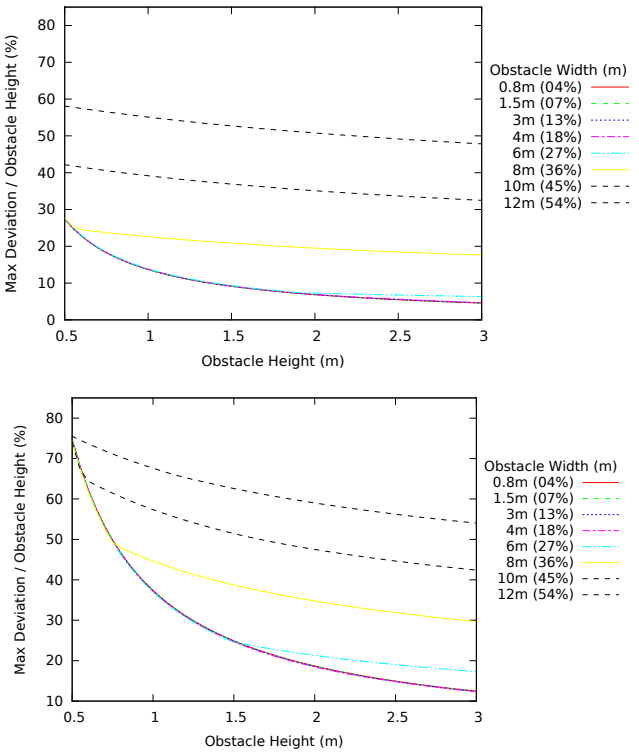


Fig. 6. Terrain deviation with artificial obstacle: the maximum deviation, as percentage of obstacle height, is plotted against the obstacle height. The interest area is $22m \times 22m$, control points are (above) 5×5 and (below) 25×25, splines order is 3, sampling ratio $50cm$. The obstacle is placed in $(x, y) = (11.0, 0.0)$, with height from 0.5m to 3m and width from 0.8m to 12m.

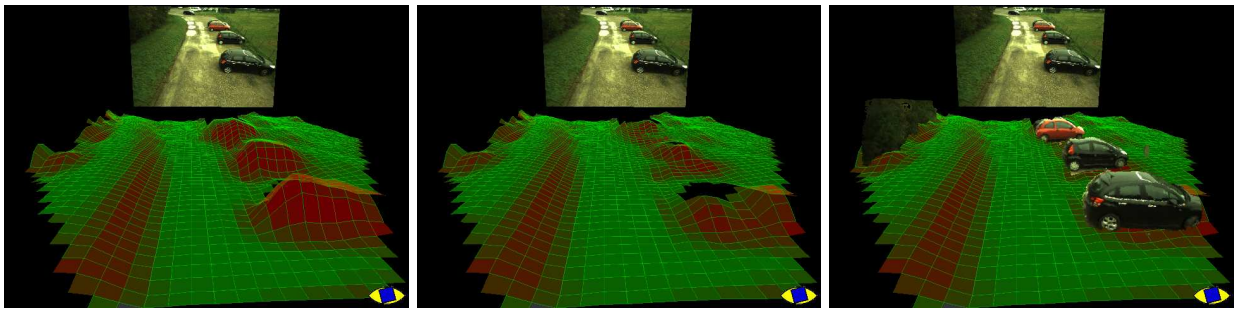
- with many control points there is a trade off between terrain fitting and obstacles segmentation: the surface is still able to cut off obstacles, but only when characterized by a minimum steepness; conversely, it is possible to fit terrains with a wider range of slopes.

Hence, as mentioned in Section I, it is very important to clearly define what an obstacles is in terms of size and *steepness*. Under a pure geometry based point of view, if the area of interest is limited to $22m \times 22m$, should an obstacle $15m$ wide still be considered an obstacle? Or would it be better to include it into the terrain? Probably the obstacle width is not enough to answer this question, and we should also look at its height: in many extreme scenarios, a $15m$ wide, $1m$ tall area, is not considered an obstacle; but a $15m$ wide, $8m$ tall area deserves more attention, even for a big mining machines.

In conclusion, the definition of the application scenario in terms of *maximum allowed terrain steepness* and *size limits* of typical obstacles compared with the area of interest is a key aspect. The right spline parameters pattern can be derived with the help of deviation plots like those shown in Fig. 6.

VII. RESULTS

The system has been tested in different configurations and scenarios (urban, off-road, mining), providing reliable



(a) (b) (c)

Fig. 5. Examples of simple B-Spline fitting (a) compared with the proposed iterative least-squares fitting (b) and the resulting obstacle segmentation (c). Note how the terrain is not affected by the equalization, while obstacles are noticeably lowered, allowing effective obstacle detection.

terrain traversability maps. With $34m \times 22m$ area of interest, 25×25 control points, spline order 3, and 640×480 pixel images, the algorithm runs at 12.5Hz (Preprocessing and DSI=45ms, Terrain Mapping=15ms, Obstacle Detection and Traversability Cost=20ms) on an Intel® Core™i7 3720QM.

Fig. 7 shows various results in different scenarios, with different cameras setup and lighting conditions. In particular, Fig. 7(g) and Fig. 7(h) show two paradigmatic processing results. First of all, the terrain estimation is not affected by the presence of a pedestrian, regardless of its position. Remember that this is a geometry based approach, so no analysis of the visual information (color, edge, features, histograms, etc.) is made to classify obstacles. Second, the pedestrian is always detected as an obstacle (outlier), wherever it is placed in the scene. Finally, each portion of the terrain surface is labeled with different traversability cost.

VIII. CONCLUSIONS

This paper presented a real-time approach for 3D terrain mapping, based on stereo vision and rational B-Splines surfaces computed by re-weighted iterative least square fitting and equalization. A cloud of 3D points is sampled into a 2.5D grid map; then grid points are iteratively fitted into rational B-Splines surfaces with different patterns of control points and degrees, depending on *traversability* consideration. The obtained surface also represents a segmentation of the initial 3D points into terrain *inliers* and *outliers*.

This method presents some advantages, compared with previous techniques: 1) obstacle/terrain segmentation is geometry based only, performed online during the surface fitting process. This allows further analysis of visual data such as color and texture to reinforce estimates. 2) as seen in Section VI, by selecting appropriate spline parameters it is possible to empirically set the maximum allowed terrain steepness; this could allow to adapt the terrain estimation to actual vehicle capabilities to traverse rough terrain, in terms of maximum slopes; 3) B-Splines minimize the terrain fitting error over the whole area of interest, providing robustness and computational efficiency; they are also locally controllable through control points, ensuring accuracy in identification of localized terrain.

The algorithm is fully frame based, so it does not perform any temporal interpolation. Knowing vehicle odometry and pose, it would be possible to integrate in time the surfaces obtained, increasing the robustness. In addition, more constrains on points' heights and derivatives could be added in the spline fitting phase.

The vision based 3D engine implementation can be negatively affected by poor visibility conditions, leading to low disparity map densities. More testing sessions are needed to characterize the system performance with respect to 3D points noise and resolution, especially at long distances.

Finally, a quantitative assessment on terrain reconstruction accuracy is now being considered, running the system on a proving ground where terrain elevations are known, or by means of simulators

IX. ACKNOWLEDGEMENT

This work has been developed in the framework of the Open intelligent systems for Future Autonomous Vehicles (OFAV) Project funded by the European Research Council (ERC) within an Advanced Investigators Grant.

REFERENCES

[1] F. Malartre, T. Feraud, C. Debain, and R. Chapuis, "Digital elevation map estimation by vision-lidar fusion," in *Robotics and Biomimetics (ROBIO), 2009 IEEE Int. Conference on*, dec. 2009, pp. 523 –528.

[2] J.-S. Gutmann, M. Fukuchi, and M. Fujita, "3d perception and environment map generation for humanoid robot navigation," *Int. J. Rob. Res.*, vol. 27, no. 10, pp. 1117–1134, 2008.

[3] H. P. Moravec, "Robot spatial perception by stereoscopic vision and 3D evidence grids," Technical Report CMU-RI-TR-96-34, CMU Robotics Institute, Tech. Rep., 1996.

[4] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *In Proc. of the ICRA 2010 workshop*.

[5] A. L. Rankin, A. Huertas, and L. H. Matthies, "Stereo vision based terrain mapping for off-road autonomous navigation," in *Proc. of SPIE, the International Society for Optical Engineering*, vol. 7332, 2009.

[6] D. Pfeiffer and U. Franke, "Towards a global optimal multi-layer stixel representation of dense 3d data," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 51.1–51.12.

[7] F. Oniga and S. Nedevschi, "Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 3, pp. 1172 –1182, march 2010.

[8] A. Cappalunga, S. Cattani, A. Broggi, M. McDaniel, and S. Dutta, "Real time 3d terrain elevation mapping using ants optimization algorithm and stereo vision," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, june 2010, pp. 902 –909.

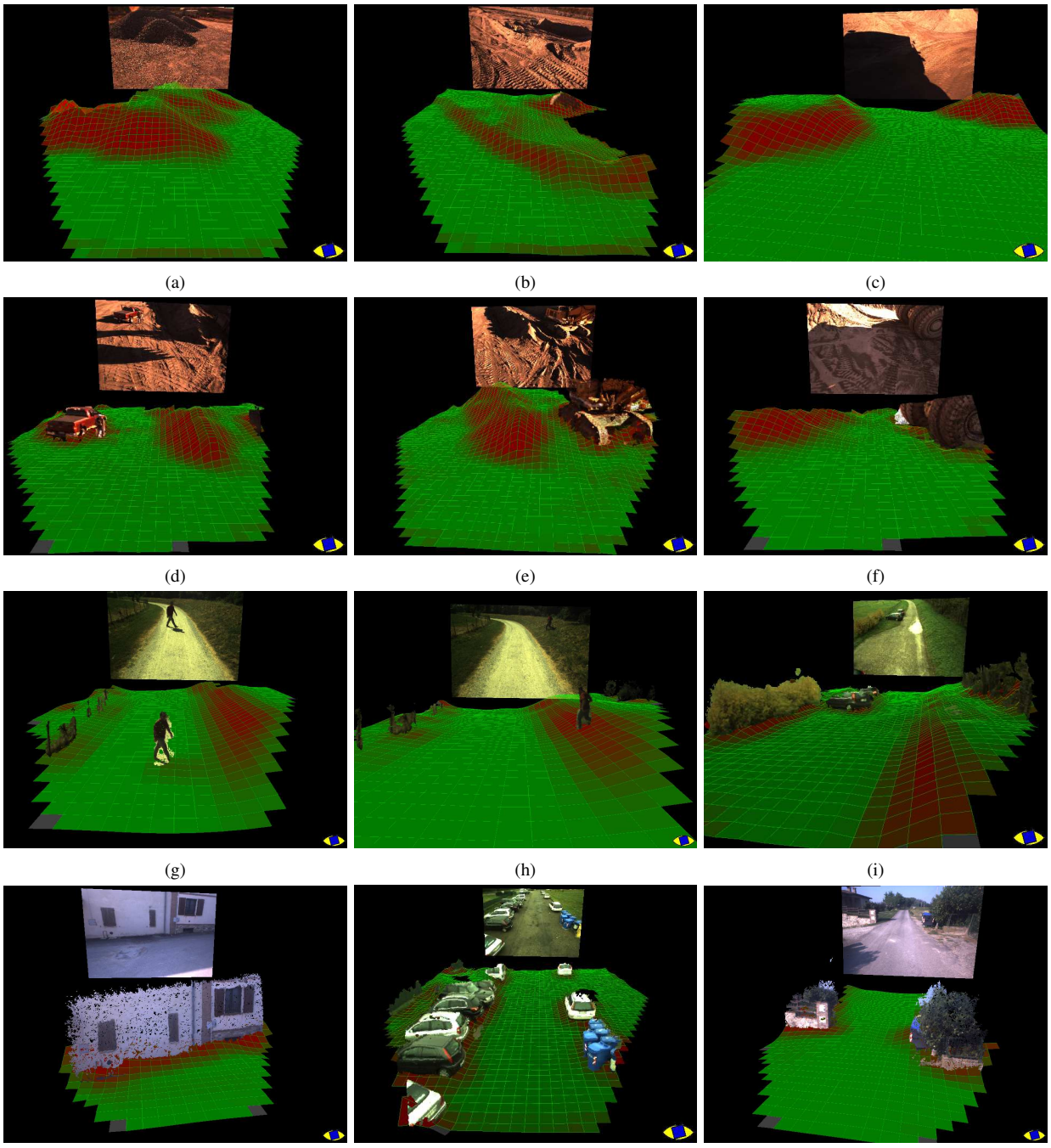


Fig. 7. Examples of terrain mapping on a variety of scenarios: terrain mapping in mining environment: (a) mounds of gravel, (b) ditch and berms, (c) gravel "canyons"; obstacle detection in mining environment: (d) car and pedestrian close to a berm, (e) excavator close to a berm, (f) truck close to a high berm; (g)(h) a pedestrian walking up and down a hill; (i) gravel road with ditch, cars, trees and bushes; (j) urban scenario with (k) parking lot and (l) houses. The area of interest is $34m \times 22m$ for all images but (g)(h)(j)(l), where is $22m \times 22m$.

[9] L. Piegl and W. Tiller, *The NURBS book (2nd ed.)*. New York, NY, USA: Springer-Verlag New York, Inc., 1997.

[10] A. Wedel, U. Franke, H. Badino, and D. Cremers, "B-spline modeling of road surfaces for freespace estimation," in *Intelligent Vehicles Symposium, 2008 IEEE*, june 2008, pp. 828 –833.

[11] 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.

[12] A. Broggi, M. Buzzoni, M. Felisa, and P. Zani, "Stereo obstacle detection in challenging environments: the VIAC experience," in *Procs. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, San Francisco, California, USA, Sept. 2011, pp. 1599–1604.

[13] M. Bertozzi, L. Bombini, A. Broggi, M. Buzzoni, E. Cardarelli, S. Cattani, P. Cerri, A. Coati, S. Debattisti, A. Falzoni, R. I. Fedriga, M. Felisa, L. Gatti, A. Giacomazzo, P. Grisleri, M. C. Laghi, L. Mazzei, P. Medici, M. Panciroli, P. P. Porta, P. Zani, and P. Versari, "VIAC: an Out of Ordinary Experiment," in *Procs. IEEE Intelligent Vehicles Symposium 2011*, Baden Baden, Germany, June 2011.