

Real Time 3D Terrain Elevation Mapping Using Ants Optimization Algorithm and Stereo Vision

Andrea Cappalunga, Stefano Cattani, Alberto Broggi, Michael S. McDaniel, Susmita Dutta

Abstract—Reliable perception of terrain slope and terrain traversability is a key-feature for any off-road unmanned ground vehicle, as well as for any Driver Assistance Systems designed to work in extreme environments, like mining. In this paper we want to present an innovative technique to build a 3D elevation map of the *traversable terrain* from a world's 3D dense data set, in real time. The 3D points are grouped into lateral and longitudinal equally spaced slices, then they are projected onto the corresponding slices' reference planes. The projections are then analyzed by a biologically inspired Optimization Algorithm able to segment points into terrains *inlier* and *outlier*; the resulting 2D terrain slopes represent an *optimal* terrain approximation along each slice. Finally, the 2D approximations are merged together, to create the overall 3D terrain surface.

The algorithm has been successfully tested with 3D data provided by a stereo camera system mounted on a Cat[®] wheel loader operating in a mining environment.

I. INTRODUCTION

Processing of 3D data sets to obtain terrain mapping information has been widely explored in literature, both in Robotics and Unmanned Ground Vehicles fields. The most of these approaches, or at least those able to provide some real time capabilities, make use of Cartesian grid maps containing cells of uniform size. These cells can store various information about the corresponding world portion, depending on the approach and algorithm's scope. The simplest grid maps normally contain just *occupancy* information, useful for basic autonomous driving. Elevation grid maps typically add, to the occupancy information, a single height information for each cell, obtaining the so called 2.5D grid maps [1]. A more complex type of Cartesian grid map make use of adjacent *stacks of cubes* [2], each one classified occupied or not, depending on the presence of 3D data inside of them. Both in elevation and cubes maps, each cell has to be classified, at least, as to be part of the terrain or not.

All these Cartesian cells based solutions have in common the prevalent use of *local* considerations and analysis to classify each cell. Some authors [3] use visual information, like colors and texture, to understand if a cell is part of the terrain, of the vegetation, of a traffic isle or other road structures, applying pattern recognition techniques. Geometry-based cells classification is also performed, fitting them with local planes patches; the residual of this fitting is a measure of the roughness of the terrain. On the basis of the



Fig. 1. The Cat[®] 980H wheel loader

information contained in each cell, a traversability cost of traveling through that cell can be added [4].

A different approach was presented by Labayrade in [5]: here a global model of the terrain is obtained from Disparity Space Image, with the strong constraints of flat terrain. While this approach find a *global* fitting, it is applicable only in urban environment when the scene complexity is not too high (not too many obstacles, not big traffic isle, etc.). Also in [6] the terrain is estimated upon a flat plane approximation hypothesis, applying RANSAC fitting to segment the 3D points in road in-liers and out-liers. Then, a post processing phase will classify the outliers as obstacles, road structures and other high level patterns. This kind of global fitting requires a high fidelity terrain model and sufficient computational power to execute in real time. Unfortunately, these conditions may not yet exist for an off-road environment as investigated.

A. Toward a Different Approach

The algorithm described in this paper tries to take advantage of both local and global analysis and fitting. A set of 2D longitudinal and lateral models replaces the 3D global model. This greatly reduces the computational complexity of the problem. Each of these models represent the projection of 3D point lying of a world "slice" with a given width. The global slope of these projections is estimated applying an Ants Colony Optimizations (ACO) [7]: this biologically inspired evolutionary algorithm identifies the optimal slope as a shortest path problem, *cutting out* those 3D points that are not part of the *traversable* terrain, according to specific cost function. The longitudinal and lateral projections will then fused together, to obtain a 3D Cartesian grid map.

A. Cappalunga, S. Cattani and A. Broggi are with VisLab - Dipartimento di Ingegneria dell'Informazione, Università di Parma, Italy {kappa, cattani, broggi}@ce.unipr.com

M. S. McDaniel and S. Dutta are with Caterpillar Inc., Peoria, USA {McDanielMichael.S, dutta-susmita}@cat.com

In Section VI deep details are provided about the ACO algorithm. Key aspects of the terrain definition are encoded into the ACO cost function and movement rules.

So, what is *traversable terrain*? Our definition is pretty straightforward: *any surface where the vehicle can drive on it without problems*. In other words, *any surface not too steep for the vehicle capabilities*. In Section VI will be explained in details how to define the ACO's cost function and ant's movement rules on the basis of the maximum steepness allowed.

B. Test Bed

The host vehicle for testing and development of this algorithm was a Cat[®] 980H wheel loader (shown in Fig. 1) operating in a sand and gravel quarry.

II. HARDWARE AND SOFTWARE SETUP

The vehicle was equipped with 2 AVT Stingray FireWire color cameras, mounted on the back hood; the cameras were connected to an industrial PC (fanless 1.6GHz Intel Core Duo running VisLab's GOLD framework [8] on a Linux OS) and synchronized through RS232 connection, with a capturing frame rate of 10Hz.



Fig. 2. Stereo cameras on the wheel loader hood

In Fig. 2 is shown the cameras' mounting position: with an height of ~ 2.9 m above the ground, a baseline of 0.5 m, a pitch of ~ 45 degrees with 2.3 mm focal length lenses. The cameras cover a region of interest 10 x 8 meters with full stereo overlapping.

III. ALGORITHM OVERVIEW

The approach is pretty straightforward, and can be described as a 4 step procedure:

- 1) *3D world points cloud*. The algorithm needs a data set of 3D world points $p_{x,y,z}$. The data set must provide enough points to support the desired resolution over the area of interest. Similar to other optimization algorithms, the ACO exploits the richness of the data set to avoid problems of local minima. Implementation details in Section IV.
- 2) *2D projections*. The world is divided into longitudinal and lateral slices of fixed width, height and length. Each 3D point is projected onto its corresponding *2D reference planes* P_i and P_j (longitudinal slice and lateral slice) to capture the profile associated with each slice. The profile includes all points in the slice whether they represent terrain, an object, or spurious noise. The length and height of the slices are defined by the region of interest and the required resolution

constraining the slice width. The resulting map has Cartesian cells of area $w \bullet w$ meters squared. Implementations details in Section V

- 3) *2D projections' slope estimation*.

The goal is to extract, from each profile, the *traversable slope*: a 2D curve representing the terrain surface where it is possible to drive safely, as defined in Section I. Given the projection properties described in Section V, it will be shown in Section VI how the ACO algorithm performs this segmentation in most cases.

- 4) *3D grid mapping*. At this point a set of longitudinal and lateral terrain profiles are available, each one representing a limited portion of the region of interest. The orthogonal slices are then merged together at their intersecting points to produce a grid of the terrain as a set of (x,y) coordinates with their corresponding height z . Interpolating these points in the 3D grid map then gives an estimate of the terrain throughout the region of interest.

- 5) *Inliers and Outliers*. As a valuable side effect, each 3D point from the original point cloud can now be easily classified in the traversable terrain as an in-lier or out-lier.

IV. 3D WORLD POINTS ENGINE

The 3D world points engine selected was a stereo based Disparity Space Image (DSI). Stereo vision is a relatively low cost solution, considering the high density point cloud sets generated at each frame. Moreover, compared with laser or sonar based methods, it brings other information, like colors, texture, lighting, etc. that may be used in post processing phases for high level terrain classification (grass, dust, asphalt, etc.). However, as already mentioned, vision is not mandatory: any other engine, device or system capable of providing point clouds of data with sufficient density and spread can be used with this approach.

A. DSI Engine

VisLab has developed, as a part of its framework GOLD, an efficient single frame DSI engine, based on sum of absolute differences (SAD) technique [9]. The engine is capable of exploiting the inherent parallelism of modern multi-core, SIMD1 -capable CPUs, computing a disparity map of 320x240 pixels in ~ 20 ms on the hardware available on the vehicle.

This engine has been widely used to develop obstacle detector systems for automotive applications. It has been extensively tested during many autonomous driving session, such as DARPA Grand Challenge 2005 and Urban Challenge 2007 [10] [11].

Fig. 3 shows a DSI application, with its pre-processing.

V. 2D PROJECTIONS

Other methods present in the literature try to solve the terrain estimation problem by dividing the world in several cells, estimating, for each cell a value that in the simplest case is an elevation level, while in complex ones it represents



Fig. 3. Example of DSI engine processing: (a) left image, (b) right image (c) disparity depth map with the usual color coding. Note how the left and right images have been processed with *rectification* and lenses distortion removal procedure

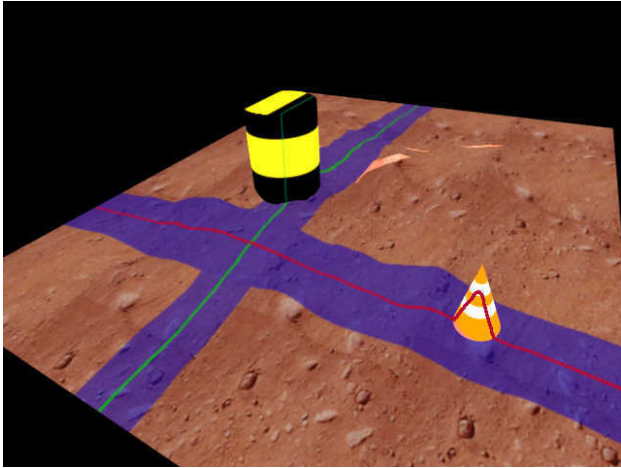


Fig. 4. Synthetic terrain scenario in which two example slices are shown together with the traces of the two vertical planes (lateral plane in red, longitudinal plane in green) in which the points belonging to the slice will be projected on

a stack of cubes. The limit the traditional algorithms can not overcome is bound to the local analysis approach they use: the correlation among the several adjacent cells is ignored while building the terrain model, and sometimes only used a-posteriori according to the goal they want to achieve, but unfortunately information has already been lost.

The proposed method instead bases its process on a global representation of the terrain as much as possible; this is the reason why the world is not analyzed cell-wise, but by analyzing bi-dimensional projections.

A. Projection definition

2D-projections are grey scale images representing the Cartesian projection of the 3D-points belonging to a slice of the world, onto their slicing plane; so they are actually bi-dimensional views of several cross-sections of the world.

B. Projections building

Figure 4 shows a typical scenario with terrain and outliers, and gives an example on how the world is divided in slices: at each frame, in fact, the algorithm creates $2 \bullet N$ two views: longitudinal and lateral (the image shows only 1 slice per

view). To better understand this process, imagine that all 3D points in the blue slices collapse onto the plane identified by their respective green/red trace; the result of the operation is shown in Fig.5(a) and (b). In this case, the shape of the outliers appears clearly in the projected images: in the lateral projection (a) a triangle on the right side represents the cone, while in longitudinal projection (b) the rectangle is relative to the barrel. All other projections are created in the same way, i.e. by projecting the points belonging to their slice on planes parallel to the red one for lateral projections and on planes parallel to the green one for longitudinal projections.

C. Greyscale

The output example shown in Fig.5 is binarized to better explain the projections creation; actually those images are created as greyscale, in order to improve the performance of the profile extraction step executed on the images. So the pixels in these images are “accumulators” that count the number of 3D points present in their relative area¹. In this way, the larger the number of 3D-points in the interest area of a pixel, the larger will be its value.

D. Realtime

The impact of the realtime constraint of this method on this algorithm step is determined by two parameters: number of projections and resolution of projections. It is important to highlight that this step implies the processing of $2 \bullet N$ images at each execution loop, so it is necessary to tune both parameters in order to find a good tradeoff between the computational time and the output accuracy according to the available processing hardware. The latest innovations on multicore CPU technologies indeed help in this kind of problems, because both the images creation and the following profiles extraction can be run on each projection independently of the others, thus highly parallelizing the process.

This process enables analysis of the world as a grid, splitting the problem of terrain estimation into the analysis of several ground profiles, one for each projection; after that,

¹The projection images will generally have very small size, i.e. if a world slice of 10m long, 6m high and 0.5m thick is projected on a 200x120 pixels image, each pixel represents a 5 cm square.

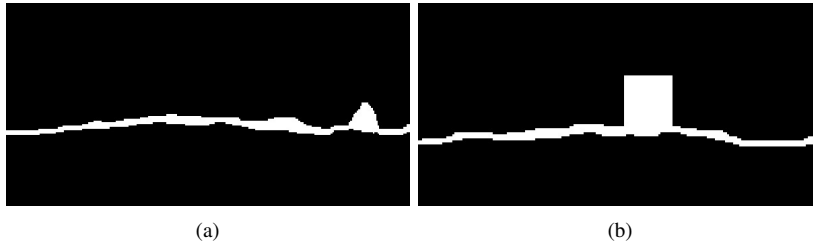


Fig. 5. Example of 2D projections relative to the scenario shown in Fig.4: (a) is a Front projection, relative to the slice with red line, (b) is a Side projection, relative to the slice with the green line

it will be possible to fuse all the information extracted from each 2D projection image, without losing information of the globality of the terrain in any step.

VI. PROJECTIONS' SLOPE ESTIMATION

Once the projection images are ready, it is possible to analyze those images and extract a terrain approximation at every projection. Several methods have been tried for extracting ground profiles from the projection images and the one that shows the best result is based on Ant Colonies Optimizations (well known within swarm intelligence algorithms). This method uses a distributed meta-heuristic for combinatorial optimization problems, inspired by the communication system of biological ants. Experimental observations show that a colony of real ants, after a transitory phase, always finds the shortest path from the nest to the food. This is achieved by depositing pheromone, an odorous substance, on the terrain to attract following ants. Arriving at a decision point, ants make a probabilistic choice based on the amount of pheromone they smell in correspondence to all the possible roads. Due to this *autocatalytic* process, the shortest path (the one covered in fewest steps) emerges as the one with the largest amount of pheromone. Returning to the slope estimation problem, the goal becomes that of building one ant colony for each 2D projection, moving pixel by pixel, trying to find the optimal slope curve for each projection.

A. The ACO approach

The complete and generic representation of a combinatorial optimization problem which can be solved by artificial ACO algorithms [7] falls out of the current paper's scope².

In fewer words, the ACO algorithms are implemented by building a colony of artificial ants that tries to approximate

²However, for readers convenience, it may be worth to recall some important ACO's mathematical entities:

1) a finite set Λ of *connections* λ_{ij} among components χ_i and χ_j in X ; 2) a *cost function* $\Gamma(\lambda_{ij}, t)$ with which assign to each λ_{ij} its own cost value, defined as $\Gamma_{\lambda_{ij}} \equiv \Gamma(\lambda_{ij}, t)$, where t is a time measure; 3) a finite set $\Omega(X, \Lambda, t)$ of *constraints* between X and Λ elements; 4) a set Σ of *states* defined as variable length sequences of components: $\sigma = \langle \chi_i, \chi_j, \dots, \chi_k, \dots \rangle$; 5) a set $\widehat{\Sigma}(t) \in \Sigma$ if *feasible states*, defined by $\Omega(X, \Lambda, t)$; 6) a state σ_1 is a *feasible neighbor* of the state σ_2 if : (i) both σ_1 and σ_2 are in $\widehat{\Sigma}(t)$; (ii) σ_1 can be reached from σ_2 in one single step: since χ_i is the last element of the σ_2 sequence, σ_1 is a neighbor of σ_2 if $\exists \chi_j \in X : \exists \lambda_{ij} \in \Lambda$ and $\sigma_1 = \langle \sigma_2, \chi_j \rangle$. The set of feasible neighbor of σ is called $N_\sigma(t)$; 7) a *solution* Ψ is an element of $\widehat{\Sigma}(t)$ that meets some *exit condition* e ; 8) an *objective cost function* Γ_Ψ which assigns a cost value to each solution reached.

the optimal solution of a shortest path problem in the following way:

- Each ant starts from its own starting state and moves through one of its own feasible neighbors, thus building a solution step by step. The processing ends when the ant reaches its own exit condition (i.e. final state).
- Often a *heuristic function* is supplied, to indicate the convenience of moving from a state to another, based on a priori knowledge.
- Each connection has an associated *pheromone trail* encoding the experience of the previous ants: the higher the pheromone deposit, the lower the costs of previous ants' solutions that included the connection.
- Each ant moves towards states of its own feasible neighbors applying a *probabilistic rule*. The probabilistic rule is a function of: (i) the heuristic function; (ii) the pheromone trail; (iii) the information yield by the ant during its past travel to the current state.
- Once an ant has reached the exit condition it increases the quantity of pheromone on its path in a way inversely proportional to the cost of the solution it constructed. This is a *reinforcement learning* strategy that modifies the way the following ants will perceive the problem, thus performing a distributed adaptive approach.

B. Implementation

Hence, the problem of finding an optimal projection slope estimation must be formalized in terms of *shortest path problem* through a graph, where each solution is represented by a path satisfying the constraints and meeting some exit condition. In particular we have:

- graph's nodes corresponds to projection's pixels; so ants move from pixel to pixel;
- the starting nodes are located on the left bound of the projection image;
- the exit nodes are located on the right bound of the projection image;
- the heuristic functions depends on the projection's pixels values, the higher the better;
- the cost function will be computed, again, on the basis of the projection's pixels values, the higher the better;
- ants' move from one pixel to another governed by a set of *motion rules and constraints*.

Focusing on motion rules and constraints, the rules determine the ants ability to *follow only a given range of*

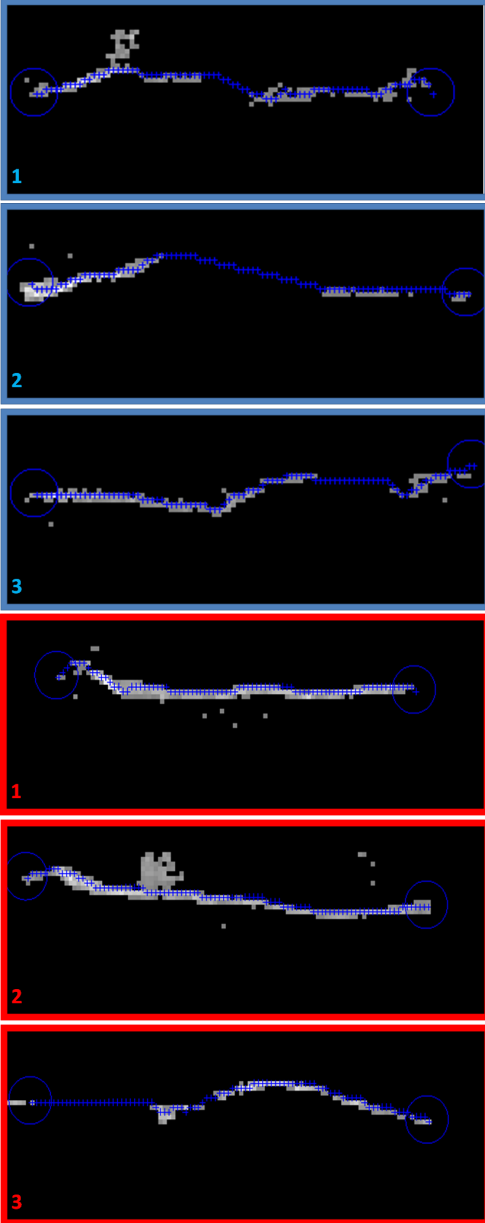
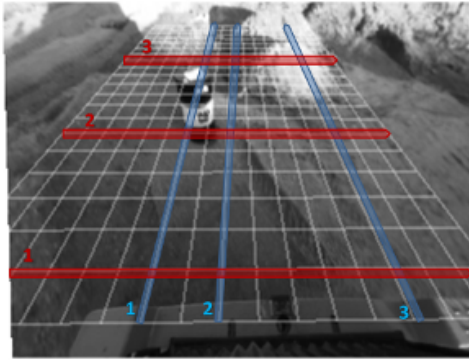


Fig. 6. From top to bottom: the right source image, with highlighted the longitudinal and lateral projection analyzed; 3 side projection with the resultant ant path superimposed with blue crosses; 3 front projection with the resultant ant path superimposed with blue crosses. Note how in (1 blue) and (2 red) the terrain outliers are cut off by the ants algorithm. Note also in (1-2-3 blue) and (3 red) how the ants are able to deal with missing terrain points, linking unconnected parts of the terrain

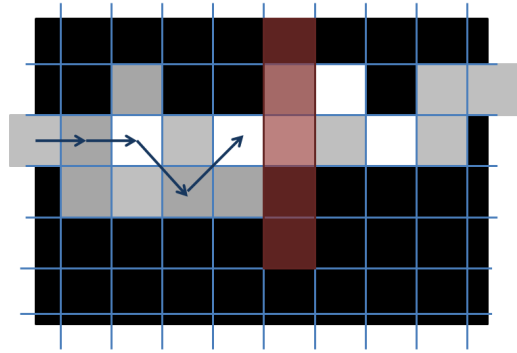


Fig. 7. Example of an ant's path: in red is highlighted the current set of neighbor pixels, placed on the next column

slopes crest' steepness. Hence, they implicitly define what is terrain and what is not. When an ant faces a slope too steep for its motion ability, it will inevitably cut it off. Since there is a direct mapping between the projection's pixels and 3D world points, defining the maximum and minimum bounds of steepness for the pixels slopes, automatically sets the corresponding 3D world points slope's steepness limits for the mapped terrain. This unique feature of the ACO algorithm perfectly fits the constraint of *traversable* terrain given in Section I.

Starting pixels, final pixels: according to the definition of projection images given in Section V, each row always represents a given height z in the world reference system, while columns represent a given x or y distance, respectively for longitudinal and lateral projections. Regardless of the particular orientation, the aim of the ant colony is to find a mapping (i.e. function) from each column to the corresponding row, represented as a sequence (i.e. path) of (x, y, z) points. These paths define the projection slope. Hence, each ant has to move, on the image, from *left* to *right*, trying to connect the starting leftmost pixel with the final rightmost one, while overlapping projections of 3D points.

Unfortunately it can so happen that the first and last columns are just blank, due to lack of 3D points in those regions. For this reason the starting pixels are chosen, projection by projection, by pixel *histograms*, using which it is possible to approximate the location of the first projected non blank pixel. A random walk process is used, if needed, to fill the gaps between the rightmost non blank pixel in the projection and the last column in the image. These random-walk tails are then discarded in post processing stage, since they do not provide any valuable information. In Fig. 6 the starting areas are located around the left blue circles, while the right blue circle marks the last best valid pixel before the random tail.

Motion rules: at each step, ants move from pixel to pixel. In particular, each ant can move from the current pixel to a limited subset of the neighboring (moving left to right) *column's* pixels (see Fig. 7). By varying the subset size it is

possible to bound the ants' motion ability. How ants choose the next pixel is determined by these motion rules, which can be divided into two levels. The first one makes the ants attracted to those pixels which have the highest brightness and/or highest pheromone levels, which is typically called *random-proportional*³. Recalling how the projection images are built, high levels of brightness correspond to a high number of 3D points accumulating on the pixel. 3D points on the terrain tend to accumulate on the same pixel generating higher brightness than the non-terrain points, thus making them more reliable. The previous ants' experience comes into the moving rule as well, in the form of the pheromone trails it leaves behind.

The second level consists of the so-called *pseudo-random-proportional*⁴ rule, using which it is possible to improve the exploitation behavior of the ants. This rule *reviews* the previous choice made with random-proportional, thus giving new chances to pixels with higher brightness. This is particularly useful in the final iterations, where the pheromone deposit is very strong. The destination pixel is chosen from among the same neighborhood set, made of the next column's subset pixels.

Cost function, pheromone update and pheromone evaporation: once the ants reach one of the final pixels, each ant has to evaluate its own path by the *cost function*⁵: the higher the traversed pixels' brightness, the lower the cost. So the path cost is inversely proportional to the average brightness

³Random-proportional:

$$a_{\chi_k} = \begin{cases} \frac{(\alpha) \cdot \tau_{hk}(t) + (1 - \alpha) \cdot \eta_{hk}}{\sum_{\chi_l \in \overline{N_{\sigma_h}}} (\alpha) \cdot \tau_{hl}(t) + (1 - \alpha) \cdot \eta_{hl}} & \text{if } \chi_k \in \overline{N_{\sigma_h}} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where:

- σ_h is the current state.
- a_{χ_k} is the probability of state $\langle \sigma_h, \chi_k \rangle$ of being the next state.
- $\overline{N_{\sigma_h}} = N_{\sigma_h} \cap \{(r, j+1), r \in N\}$ are the neighbors on the next column $j+1$.
- α is a parameter with which it is possible to tune the balance between edge-exploitation and pheromone-exploitation behavior of the ants.

⁴Pseudo-random-proportional:

$$\chi_{next} = \begin{cases} \chi_j : \eta_{\chi_k} = \max_{\chi_l \in \overline{N_{\sigma_h}}} (\eta_{\chi_l}) & \text{if } q \leq q_0 \\ \chi_k \text{ random based on Formula (1)} & \text{otherwise} \end{cases} \quad (2)$$

where q is a random variable with a uniform probability distribution, and q_0 is a threshold defined as follows:

$$q_0 = \gamma \cdot \frac{\max_{\chi_l \in \overline{N_{\sigma_h}}} (\eta_{\chi_l} - \eta_{\chi_{next}})}{\max_{\chi_l \in \overline{N_{\sigma_h}}} (\eta_{\chi_l})} \quad (3)$$

Parameter γ is used by the programmer to vary the balance between global *exploitation* and *exploration* behavior.

⁵Cost function:

$$L_k = \frac{\sum_{ij \in path_k} (255 - edge_{ij})}{\text{path length}} \quad (4)$$

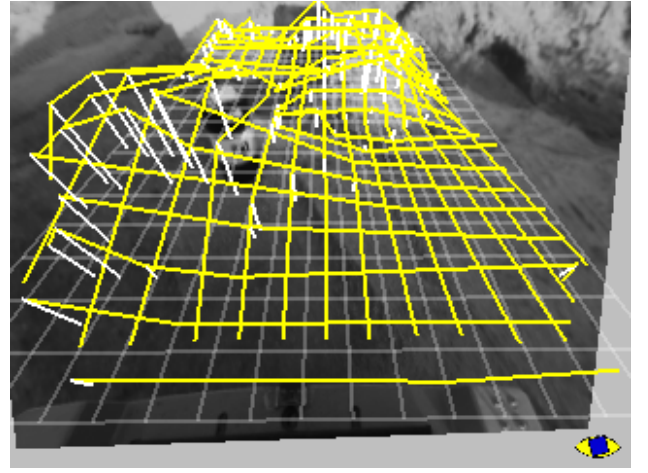


Fig. 8. The ants best paths are approximated by the yellow polyline; the profiles found on longitudinal and lateral projections are drawn on the input image

accumulated over the collection of its pixels. A path formed by bright pixels costs less than a path formed by dim pixels. On the basis of the value, each ant updates the pheromone deposit along its path, the lower the path's cost, the higher the ant's contribution.⁶

With time, the previously deposited pheromone *evaporates* at a given rate⁷, in this way early paths no longer covered by recent ants will be penalized and eventually forgotten, while giving more weight to those paths (and deposits) reconfirmed and reinforced by subsequent ants.

In summary, a colony of intelligent ants is created for each projection. As ants try to find the best path connecting starting pixels to the final pixels, adopting a collaborative behavior, the best path on each projection emerges as the optimal slope estimation. Fig. 6 shows results, with the best path highlighted in blue.

VII. TERRAIN MAPPING

The output of the ACO algorithm is a list of points representing the corresponding projection's slope. The points are further simplified with a polyline prior to fusion resulting in the final slice profiles.

⁶Ant pheromone deposit:

$$\Delta_{ij}^k = \begin{cases} Q / (L_k - L_{k-best}) & \text{if k-ant passed from pixel } i \text{ to } j \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where Q is a fixed parameter and L_{k-best} is the cost of the current best solution.

⁷Pheromone update rule:

$$\tau_{ji}(t+1) = (1 - \rho) \cdot \tau_{(ji)}(t) + \rho \cdot \sum_{k=1}^{N_{ax}} \Delta_{ij}^k(t) \quad (6)$$

- t is a time measure, and represents the evolution of the pheromone deposit.
- $\rho \in (0, 1]$ is the pheromone *evaporation ratio*.

Fig. 8 shows the output of the ground profile extraction from each slice which are not yet usable since they precede the fusion step that merges the results of both sets into a single terrain model for the region of interest. This fusion is done by fixing a grid of points $(x, y, 0.0)$ in the world, and evaluating their relative existing profiles (if present) in order to determine the surface z value for each point. If both profiles are present for that point, the z coordinate is set to the average value of the two. If only one is present z will coincide with the polyline value at the current (x, y) . If no profiles cover the current point, the location (x, y) is marked as “unknown” and the corresponding height remains to its default value $z = 0$. The output of this fusion is a model that can be queried at each (x, y) of the continuous world of the region of interest, to retrieve the z value at that point. This is done by bilinear interpolation of computed points as shown in yellow in Fig. 8(b).

VIII. RESULTS

The system has been tested in a mining environment on a Cat 980H wheel loader. The algorithm provides reliable terrain mapping as shown in Fig. 9 with 320x240 pixel images and 10 frames per second.

Fig. 9(a), as well as (c) and (i), shows how the 3D points cloud is segmented into inliers and outliers: the barrel and the pedestrian do not affect in any way the terrain mapping. In fact the ACO algorithm, operating under the movement rules/costs defined for this system, identifies them as not traversable and, consequently, are not included in the terrain approximation.

The differences between Fig. 9(k) and (l) result from the algorithm initializing terrain height $z = 0$. As the vehicle approaches the gravel hill, as in (l), the rightmost portion of the region of interest is occluded by the hill and there are no 3D points in the data set.

IX. CONCLUSIONS AND FUTURE WORKS

In this paper we presented a real-time approach for 3D terrain mapping, based on stereo vision and ACO algorithm. A cloud of 3D points is generated from a disparity depth map, then they are segmented in terrain *inliers* and *outliers*, on the basis of *traversability* consideration. The segmentation is done by the genetic algorithm Ants Colony Optimization.

The main advantages of this method, compared with others known in literature, are: 1) segmentation is done during the projection profile processing, before the terrain is estimated. This allows further analysis of visual data such as color and texture to reinforce estimates. 2) segmentation takes into account vehicle capabilities to traverse rough terrain. 3) segmentation does not require a global terrain model. The ACO algorithm accurately identifies localized terrain and the process of recombining the localized terrain estimates into a global model avoids local minima.

A. Future Works

Currently the algorithm is fully frame based, so it does not perform any temporal interpolation. Since the terrain has

to have a certain continuity, knowing the vehicle odometry it would be possible to integrate through the time the projections's slope computed, increasing the mapping robustness.

Exploiting the vision capabilities, another improvement may consist of taking into account also the corresponding color and texture of the 3D points, to improve the segmentation performances.

If the time constrains allows that, it is possible to compute mathematical models of each projections' slopes (e.g. cubic splines), looking forward to a more complex terrain analysis (e.g. curbs detection).

A vision based 3D engine implementation can be negatively affected by poor visibility conditions, leading to low disparity map densities. In general, more testing sessions are needed in 3D points lacks conditions, to characterize the system performance with respect of 3D points resolution and density.

Finally, it would be also interesting to evaluate the algorithm performances also with a 3D points engine different than stereo vision, like 360° multi-beam lasers scanners.

X. ACKNOWLEDGMENTS

The authors gratefully acknowledge the reviewers' comments.

REFERENCES

- [1] 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.
- [2] H. Moravec, “Robot spatial perception by stereoscopic vision and 3D evidence grids,” Technical Report CMU-RI-TR-96-34, CMU Robotics Institute, Tech. Rep., 1996.
- [3] 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*, 2009, pp. 733 210–733 210–17.
- [4] Q. C. Jiajun Gu and Y. Huang, “Rapid Traversability Assessment in 2.5D Grid-based Map,” *International Journal of Advanced Robotic Systems*, vol. 5, no. 4, pp. 389–397, 2008.
- [5] R. Labayrade and D. Aubert, “A single framework for vehicle roll, pitch, yaw estimation and obstacle detection by stereo vision,” in *Proc. of International IEEE Intelligent Vehicle Symposium*, Columbus, USA, June 2003.
- [6] F. Oniga, S. Nedeveschi, and M. Meinecke, “Temporal integration of occupancy grids detected from dense stereo using an elevation map representation,” in *Proc. of the 6th International IEEE Workshop on Intelligent Transportation*, Hamburg, Germany, Mar. 2009.
- [7] M. Dorigo and G. Di Caro, “The ant colony optimization meta-heuristic,” in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London: McGraw-Hill, 1999, pp. 11–32.
- [8] M. Bertozzi, L. Bombini, A. Broggi, P. Cerri, P. Grisleri, and P. Zani, “GOLD: A Complete Framework for Developing Artificial Vision Applications for Intelligent Vehicles,” *IEEE Intelligent Systems*, vol. 23, no. 1, pp. 69–71, Jan.–Feb. 2008.
- [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] Defence Advanced Research Projects Agency (DARPA), “Urban Challenge 2007,” available <http://www.darpa.mil/grandchallenge>.
- [11] A. Broggi, A. Cappelunga, C. Caraffi, S. Cattani, S. Ghidoni, P. Grisleri, P. P. Porta, M. Posterli, and P. Zani, “TerraMax Vision at the Urban Challenge 2007,” *IEEE Trans. on Intelligent Transportation Systems*, 2008, submitted.

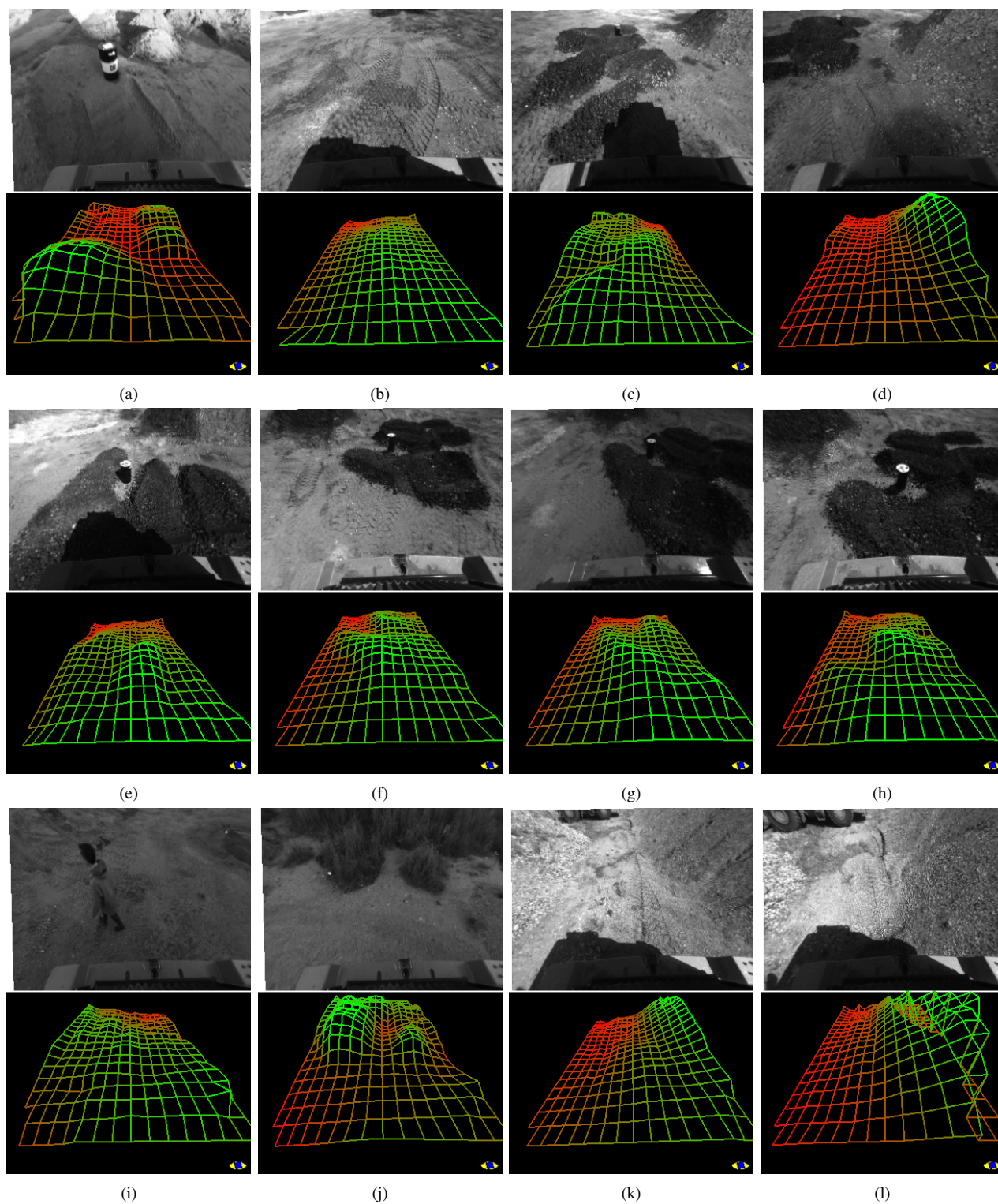


Fig. 9. Examples of terrain mapping and outlier segmentation: (a) a typical terrain mapping scenario with mounds of gravel, airborne dust, and common objects; (b) flat terrain with relative roll between vehicle and ground; (c)(d)(e)(f)(g)(h) mounds of gravel, some including a barrel; (i) flat terrain with a pedestrian; (j) terrain with bushes; (k)(l) a gravel "canyon." Note in every case the terrain model excludes the effects of objects.