# ON-THE-WAY CITY MOBILE MAPPING USING LASER RANGE SCANNER AND FISHEYE CAMERA

**Xavier Brun, Jean-Emmanuel Deschaud and François Goulette**

Mines Paris
60, boulevard Saint Michel
75272 PARIS Cedex 06, FRANCE
xavier.brun@ensmp.fr, deschaud@ensmp.fr, francois.goulette@ensmp.fr
http://caor.ensmp.fr/

**KEY WORDS:** mobile mapping, laser range sensor, fisheye camera, real time, 3D textured models

**ABSTRACT:**

We present a new Mobile Mapping System mounted on a vehicle to reconstruct outdoor environment in real time. The scanning system is based on two sensors, a laser range finder and a camera equipped with a wide-angle fisheye lens. We can produce 3D coloured points or 3D textured triangulated models of the nearby environment of the vehicle. We explain our choice of the fisheye lens for its large aperture angle covering a large part of the world surrounding the car ($180^\circ$). An important part of the process is to calibrate the system: we need to get the rigid transformation between the two frames, and the fisheye projection model which is different from the usual "pin-hole" model. We present a new and faster way to determine the fisheye projection parameters. Once the system has been calibrated, the data acquisition and processing to create models are done "on-the-way". Results illustrating in city streets the possibility of the new scanning system are presented.

## 1 INTRODUCTION

Models of outdoor environments such as cities and roads are useful for various applications, such as architecture planning, 3D cartography for car and pedestrian navigation, virtual tourism, Virtual Reality, video games, and so on. For realistic modelling, a precise digitizing of 3D geometries and textures is useful, at the level and scale of the desired use (eg, ground level for car and pedestrian navigation). However, given a desired level of detail, and a desired size and surface of environments to be digitized, one has to face the issues of quantity of information to be acquired, processed and stored, and of overall processing time. To obtain 3D geometries of outdoor environments at the ground level, one can use passive methods such as photogrammetry or stereovision to deliver the location of specific points or features in the scene, or active methods using lasers or structured light to deliver 3D points all over a scanned surface. The active methods usually give more dense results. Several people have explored the use of laser range sensors mounted on vehicles, in order to scan large areas of environments. One possibility is to define several scanning spots, and to make a fusion of the 3D data clouds with adapted frame transformation between the locations, thus requiring post-processing ((Allen et al., 2001); (Stamos and Allen, 2000)). Another possibility is to use the vehicle itself as a scanning device, its movement defining one of the scanning directions ((Frueh and Zakhor, 2003); (Zhao and Shibasaki, 2003)). In the reported approaches however, the processing of data collected during scanning is done afterwards and is quite heavy, of the order of several hours, and would not allow on-board processing.

We have developped an integrated on-board laser range sensing system, which has been designed specifically for dense 3D digitizing of the geometry and color of cities and roads, at the vehicle speed (see (Goulette et al., 2006) for previous presentation and analysis of the system for 3D points alone). Our system (Figure 1) is an integrated platform, consisting of a car equipped with localization sensors (GPS, IMU) and perception sensors (laser range scanner and cameras). The laser range sensor, placed at the rear of the vehicle, performs scanning over a vertical plane perpendicular to the direction of the car. During the movement of the car, at low or normal speed, the laser is thus scanning the whole environment driven through. Geometry and color are captured together by a coupled fish-eye CCD camera and laser range scanner. Combined with geo-referencing information, we are able to create 3D points alone, colored, faceted models alone or textured, of the geometry of surroundings. For the processing of sensor data, we use integrated real-time software designed specifically for our platform, which realizes the functions of generic 3D digitizing and of 3D modelling adapted to cities and road environments. The software design has been done to allow on-board real-time processing along the movement of the vehicle.

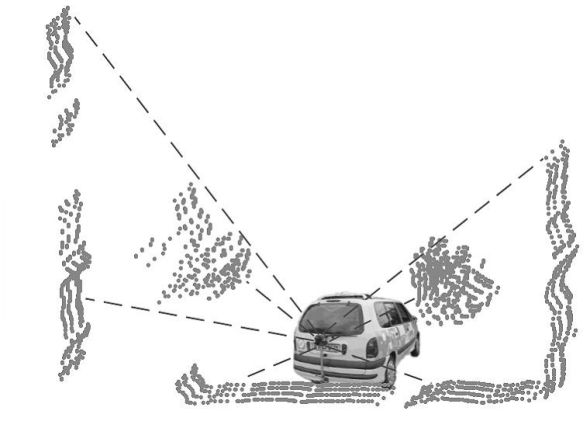


Figure 1: LARA-3D, on-board laser range sensing prototype

We present in this paper the integrated system, the methodology and algorithms developped, and we discuss our choices. In Section 2 is presented the experimental platform, named LARA-3D; Section 3 is dedicated to the description of processings regarding geometry reconstruction, while Section 4 details the algorithms used to add color and texture.

## 2 THE LARA-3D PLATFORM

### 2.1 Functional analysis

In order to produce the various results of 3D points alone, colored, faceted models alone and textures, from the information of range scanner, camera and localization sensors (IMU and GPS), a functional analysis leads to the definition of several functions to be realized by the on-board system (reported in Figure 2 as a data-flow diagram):

- Localization: need for precise, high-frequency and real-time position, speed and orientation of the vehicle;
- Geo-referencing: transforming the range data (distance and angle of the laser scanner) into 3D points expressed in a common geo-referenced frame;
- Triangulation: producing triangle facets from the 3D points;
- Coloring: finding the corresponding colors for the 3D points;
- Texturing: adding textures (extracted from photographs) to triangle facets.

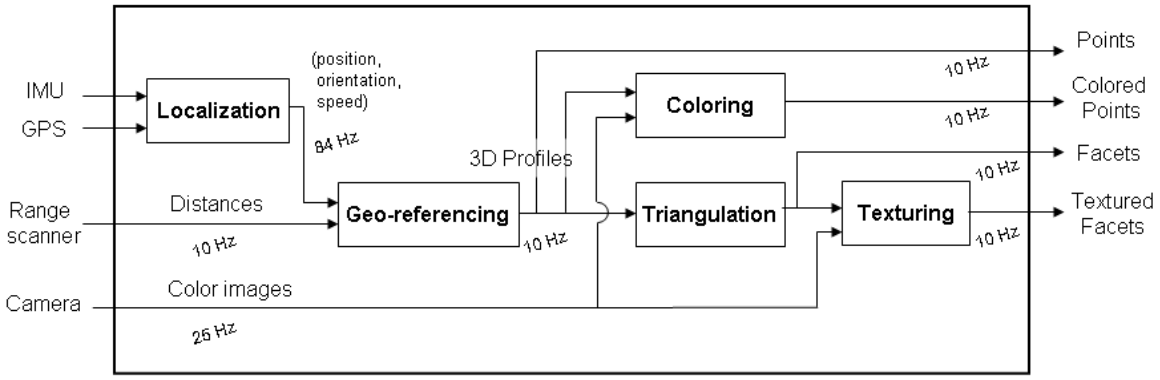



Figure 2: Framework for construction of 3D points.

### 2.2 Hardware and Software Setup

We have implemented the functions described on a prototype vehicle, called LARA-3D. It consists of a car (Renault Espace) equipped with localization sensors (GPS, IMU, odometers), perception sensors (rotating laser range sensor, cameras), on-board PC computer with adapted software environment and data storage capacity.

For localization, we use a GPS antenna AgGPS132 from Trimble (AgGPS 2000), giving non differential absolute positions at at a rate of 10 Hz, and a latency time comprised between 30 ms and 150 ms. The Inertial Measurement Unit is a strapdown VG600CA-200 from Crossbow (DMU 99), delivering data at a rate of 84 Hz. The laser range sensor is an LD Automotive from IBEO (LD IBEO), delivering 1080 distances per rotation at the rate of 10 rotations per second. On our platform we have an on-board computer with bi-processor Pentium III at 750 MHz using the Windows 2000 operating system.

In order to add color and textures, we considered various possibilities and focused on the use of fish-eye CCD camera with wide-aperture lens. In order to produce photometric redundancy for texture map creation further in the processing, we have chosen matricial CCD that give a complete scene area for each frame (differently from linear CCD). As data capture must be done during the movement of the vehicle, we need to have a rate of several frames per second. The resolution chosen is typical of usual products (Table 1).

The range scanner covers an area of $270^\circ$. We have made a comparison of angles that would be covered by our camera with three lenses with different focal lengths. Fish-eye lenses are specific lenses using different optics principles than regular ones, that allow to obtain very short focal length and then to cover up to more

Table 1: Characteristics of elements

| | | |
|---|---|---|
| Laser scanner | Number of points | 1080 per profile |
| | Angle covered | $270^\circ$ |
| | Profiles rate | 10 Hz |
| Color CCD | Number of pixels | 780x582 pixels |
| | CCD size (1/2"') | $6.47mm \times 4.83mm$ |
| | Frames rate | 25 Hz |
| Fish-eye lens | Focal length | 1.4 mm |
| | Aperture angle | $185^\circ \times 185^\circ$ |

than $180^\circ$ in all directions. Using a camera with fish-eye lens allows to capture a complete half-space with only one camera. To cover the same area as the scanner, only two cameras with fish-eye lenses would be needed, whereas in other caseswe would require 3 or more cameras. This motivated us to use a coupled system with fish-eye lens mounted on CCD camera for images, in addition to the laser range scanner capturing geometry (Figure 2.2).

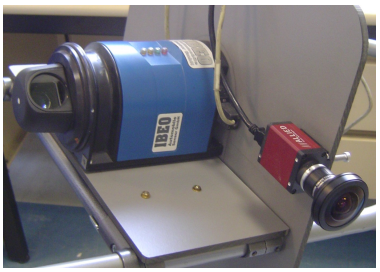


Figure 3: Coupled vision system

To design the dedicated software, we use a specific environment

for prototyping of real-time on-board applications, called RTMAPS (Intempora). It allows to design and implement processing algorithms in C++, and then test them on sets of real data acquired before, re-played at speeds lower (or faster) than the original ones. It allows an easy determination of computation times. This environment makes possible to execute non optimized algorithms, measure computation times and determine whether it is possible to speed up replay or necessary to perform optimization. One considers that an algorithm supports real-time when a replay at normal speed allows processing without loss of data or synchronization. In that case it is possible to install the software on-board for simultaneous acquisition and processing. For software prototyping, we used a simple powerful desktop PC with dualcore processor. The various functions described above in the functional analysis have been implemented as separate RTMAPS components, which can be linked together in a data-flow interface. We also use components to decode raw input (sensors) data, which are in specific formats.

## 3 3D GEOMETRY

### 3.1 Vehicle localization

For precise, real-time, localization we use a GPS-INS coupling (Abuhadrous et al., 2003)). Inertial Measurement Units (IMU) systems alone give only differential information (accelerations and rotation speeds), which need to be integrated in order to give position, orientation and speed of the vehicle (Inertial Navigation System, INS). These information are precise and high-rate yet subject to drift along time. GPS information give absolute position (and speed), but at lower rate, and they can sometimes be missing. A combination of both, by fusion of data, can be performed and give fast, accurate and absolute positions, orientations and speeds.

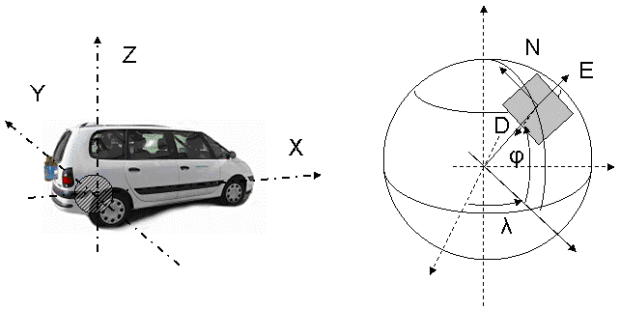



Figure 4: referentials

The vehicle referential frame is defined with the $X$ direction along the vehicle, and its center located above the rear wheel axis (Figure 4). Its position is given in a referential fixed to the Earth, with the representation $r$ (latitude, longitude, altitude). Its orientation and speed are given with respect to the local tangent plane (NED - North, East, Down). We use an additional referential $\Re^0$ , defined with the position and local tangent plane of the vehicle at some initial point, and we use for convenience the position $s(X, Y, Z)$ of the vehicle in this referential. The orientation of the vehicle is expressed in NED frame with 3 Euler angles $\rho$, or equivalently with a rotation matrix $R$. The speed is denoted $v$.

To implement the Inertial Navigation System equations, we use the differential equations, relating accelerations $f$ and angular speeds $\omega$ to orientation $\rho$, speed $v$ and position $s$ of the vehicle ((Farrell and Barth, 1998)). In order to perform fast computations, we make simplifying assumptions in the equations (eg. neglecting Coriolis terms that are low compared to the sensor sensibility, but expensive in time to compute). We use for numerical integration a first-order Euler scheme.

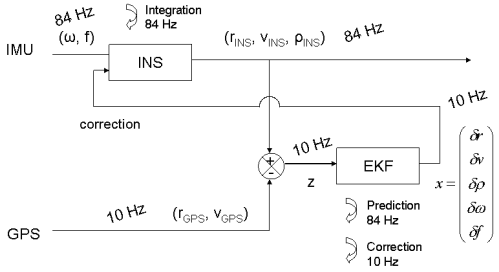



Figure 5: GPS-INS coupling

In order to correct the drift of INS, we use an Extended Kalman Filter (EKF) with feedback correction (Grewal et al. 2001), as represented on Figure 5. The observation of the filter is the vector of differences between GPS and INS positions and speeds. The state vector is the difference between actual values and INS values for position, speed, orientation, IMU values for angular speed and acceleration. Considering small variations around the actual values, a linearized equation can be written to perform prediction and correction steps of the filter. Prediction runs at the rate of INS computation, while correction runs when GPS data are available. When a correction has been computed, the IMU terms are fed forward as new IMU offsets in the INS equations. Predictions and corrections use noise models for IMU and GPS, which need numerical values determined through calibration or experimentation.

### 3.2 Geo-referencing of range data

The laser range data are given as distances and angles of measurement points in a vertical plane moving with the vehicle. It is possible to combine this data with the vehicle localization information, in order to produce sets of 3D points in a common terrestrial referential ((Abuhadrous et al., 2003)). The frame change between range sensor and vehicle referential is constant, it needs to be determined through calibration. As the range data are given as a set of points, a simple interpolation is performed, to evaluate the exact position of the vehicle for a given point in the profile (Figure 6).

### 3.3 Precision of points

One has to make the distinction between absolute and relative precision of data. The absolute precision is related to systematic bias in the values, while the relative precision is related to random noise affecting the data. In our case, we have several sources for absolute and relative precision. For the localization function, the best relative precision is given by the INS data, the noise coming from the IMU noise; the INS gives bad absolute precision, because of a drift that is corrected by the GPS; hence the best absolute precision is given by the GPS (however not very precise, because of the use of simple GPS without corrections such as DGPS, RTK...). In a series of outdoor experiments on the road scene (track), we have compared our results with reference measurements, and we can give rough estimates of the precision of our GPS-INS localization system: for relative precision, standard deviation $\sigma \approx 1mm$, and absolute precision (bias) less than 4 m.

We have verified on a workbench the correct calibration of the range sensor, and the standard deviation of 5 cm in distance announced by the manufacturer. In the geo-referencing, additional

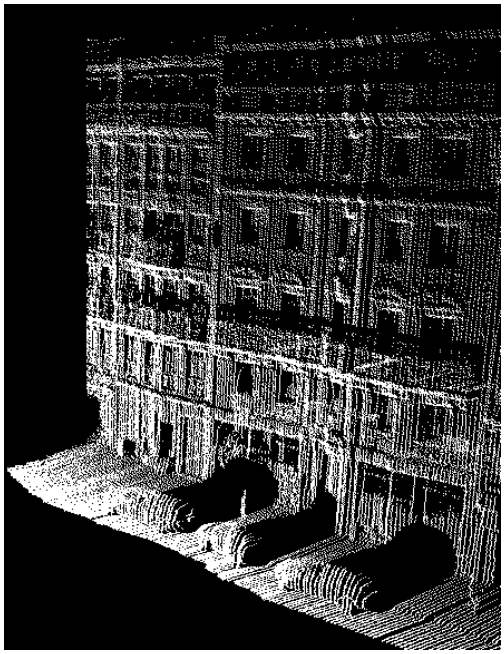


Figure 6: Points Cloud of a Street

Table 2: Estimated precision of the 3D points

| Precision | Relative (std deviation $\sigma$) | Absolute (bias) |
|---|---|---|
| Points 3D | $\approx 5cm$ (mainly in laser dir.) | $< 4m$ (when GPS available) |

sources of inaccuracy are in laser-vehicle calibration, and propagation of the vehicle orientation random noise coming from the EKF. Calculations show that, for the distances considered in our experiments, the order of magnitude of the effect of the orientation noise on the geo-referenced 3D points is more than 10 times lower than the range noise. Hence we can consider that the random noise for the positions of 3D points is mainly in the laser direction, with the magnitude of the range random noise. Concerning laser-vehicle calibration, we assume that a correct calibration has been done. For absolute precision, the remaining element is the absolute precision of position, coming from the GPS. This leads to Table 2 which summarizes the absolute and relative precision of the 3D points delivered by the LARA-3D system (Goulette et al., 2006).

Indeed, to reach the results presented above, additional processings and calibration are needed. For the localization function, we have added a first-order low-pass filter for IMU data, to eliminate high frequency noise; at the beginning of acquisition, we stay with the vehicle immobile a few seconds in order to determine the IMU initial offsets. The initial orientation of the vehicle, needed for INS integration initialization, is determined with a compass. For the GPS, we take into account the latency time between position acquisition and delivery to correct the position; we also take into account and perform correction for a "lever-arm" effect, related to the distance between actual GPS antenna and frame centre. For the Kalman filter, we need to give numerical values for the noise models: we use values coming from calibration (IMU) or experimentation (GPS). The frame change between laser and vehicle referentials is determined with a hands-on procedure.

### 3.4 Triangulation

We only use the not too far points (typically less than $30m$) because we have not enough points on object that far and so the geometry would not be representative. After that filtering, we process the profile of the laser data points with the last one. We use a incremental 2D delaunay algorithm to compute the triangles. To do this, we need to parameterize the 3D points to get 2D points, we use the cylindric coordinates of the points $r$ and $\theta$ (cf. Figure 7). The result on the urbain scene is show on Figure 8.

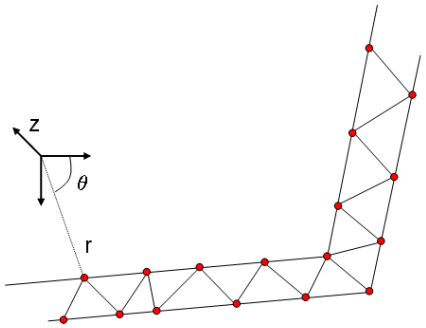



Figure 7: Schema of the triangulation

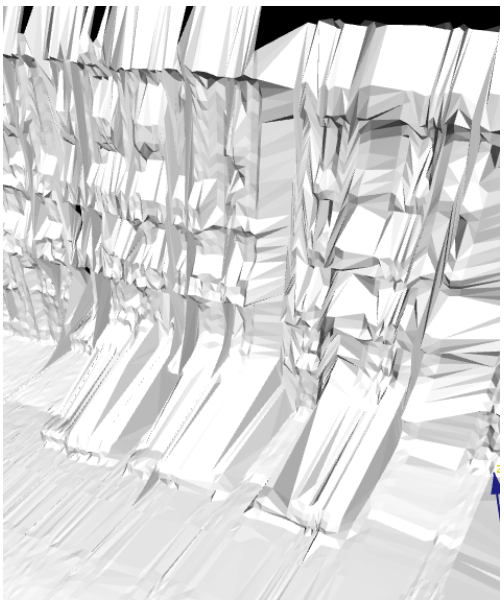


Figure 8: Triangulation of a Street

## 4 ADDING COLOR AND TEXTURE

### 4.1 Camera and Calibration

In this part we want to determine the rigid transformation between the camera and the laser frame, a rotation matrix $\Phi$ and a translation vector $\Delta$. If $P_c$ is a point in camera coordinate system and $P_l$ the same in the laser coordinate system, we have :

$$P_l = \Phi * P_c + \Delta \quad (1)$$

We use the method described by Pless and Zhang (Pless and Zhang, 2004) and improved by (R. Dupont and Fuchs, 2005). The basic idea is to view a planar calibration pattern painted as

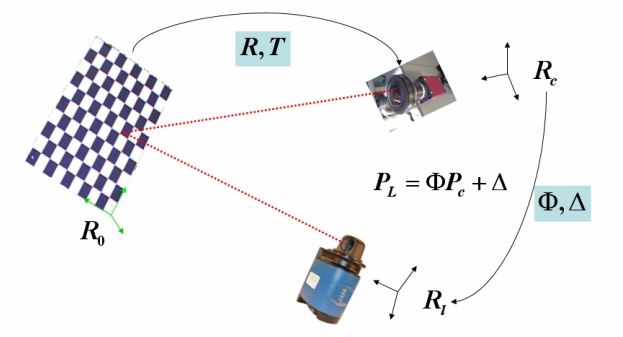



Figure 9: Camera - Laser exintrinsic Calibration

a chessboard with the camera and the laser at the same time as shown by the figure 9. The laser points must lie on the calibration plane estimated from the camera, we get a geometric constraint on the rigid transformation between the camera coordinate system and the laser coordinate system. We get the measure of each sensors at the very least fifteen times and each times, the pattern position is differents. For this calibration process, we are using a "classical" lens because we only need the extrinsics paramters. We will discuss of this point in the next paragraph. At first, we do the Tsai Camera Calibration (Tsai, 1987) to get the camera position in the world reference system defined by the pattern. For each configuration, we now have the position of the pattern in the camera reference. Those positions are defined as planes : for the $i$th pose, $N_i$ is the normal vector of the plane and $d_i$ the distance to the origin. If $P_{ij}$ are the laser points of the $i$th configuration which lie on the calibration pattern, the rigid transformation to pass to the camera reference must put them in the plan of the calibration pattern. We used the Levenberg Marquardt optimization algorithm to minimize the reprojection errors of the laser points for all configurations. The function to minimize is :

$$\sum_i \sum_j \left( \frac{N_i}{\|N_i\|} (\Phi^{-1} P_{ij} - \Delta) - d \right)^2 \quad (2)$$

Pless and Zhang(Pless and Zhang, 2004) have proposed a algebraic solution which can be used as initial solution and then the convergence is very quick (approximately 10-20 iterations). Something important needs to be noticed, the lens used in this process is not a wide-angle lens. Theoretically it does not matter because we only use the extrinsic parameters of the camera but in practice, we use the *Camera Calibration Toolbox for Matlab* for doing the Tsai Caliabration. In this method, the intrinsic and extrinsic parameters are estimated together. We assume than this estimation will be better because with the same camera, with a fisheye lens, a pixel represents a larger area of the world (typically 20 cm for something 10 meter in front of the camera in our case). The position estimation depends directly of this resolution so we do this extrinsics calibration with normal lens and after that we equipped the camera with the fisheye lens. We now need to estimate the intrinsic parameters of the vision system and this would be the concerning of the next section.

### 4.2 Adding a fisheye lens

We equipped the camera with a fisheye lens in order to get a wide angle of aperture. Consequently, the Gauss conditions are no longer available : the light touch the lens far from its center and the light is not quasi-parallel to the optics axis. So the pin-hole model can not be used, we need another projection model. We have chosen to use the equidistance model where the position of

Figure 10: Example of a fisheye image

the projected point depend on $\theta$ and not $tan(\theta)$ as in the pin-hole model (cf. Figure 11).

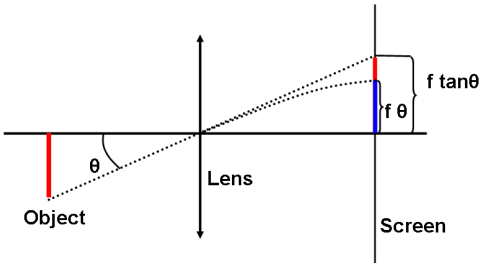



Figure 11: Differents models for the projection

After this projection, we have to pass into the pixel units. We need to know four parameters, $u_0$ and $v_0$ the center position and the size $\mu_x$ and $\mu_y$ of one pixel.If we considere a 3D point $P(\rho, \theta, \varphi)$ in the camera refence system $R_c$, we get its projection position in the pixel reference system $R_p$ by :

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \begin{pmatrix} \mu_x & 0 \\ 0 & \mu_y \end{pmatrix} r(\theta) \begin{pmatrix} cos(\varphi) \\ sin(\varphi) \end{pmatrix} \quad (3)$$

Our calibration method is based on the form of a image from a fisheye lens (cf. Figure 10). We assume the world projection on the CCD sensor to be a circle. Others pixels are black which allows us to determinate easily which pixel belongs to the circle. We first use a a canny filter to get to circle border ans we use differents classical image processing methods to estimate the parameters of the circle :

- the barycenter of the no black pixels,
- mean squared estimation to find the best fitting circle
- hough circular transformation (Renato M. Hadad, 2001)

The center of the circle is the position of the optical center projection on the image $(u_0, v_0)$ and we use the circle radius $r_c$ to estimate the $f$ parameter. The radius $r$ between the pixel position

and the center of the image is proportional to the angle $\theta$, $r = f\theta$ (cf. Figure 11) so we can estimate this ratio for the extreme value :

$$f = \frac{r_c}{\theta_{max}}. \quad (4)$$

$\theta_{max}$ is equal to the half of the aperture angle of the fisheye, known by the constructor value. The CCD cell size parameters ($\mu_x$ and $\mu_y$) are given by the previous calibration. We have observed constant values of those parameters if we used one or more images which means we get those parameters with only one picture.

### 4.3 Colored Points

We have now all the necessary parameters to project a points from the laser to its corresponding pixel in the image with the fisheye lens. We have installed this rigid and calibrate system on a vehicle geolocalized by a extended kalman filter presented in the second section. All the date processing are done in real time, each time we get laser data, we choose the temporally nearest image, we project those data in the picture and thus we affect the pixel colour to the laser point. Thanks to the localization we convert the laser data into a earth fixed reference, we get a 3D coloured points cloud (cf. Figure 12).

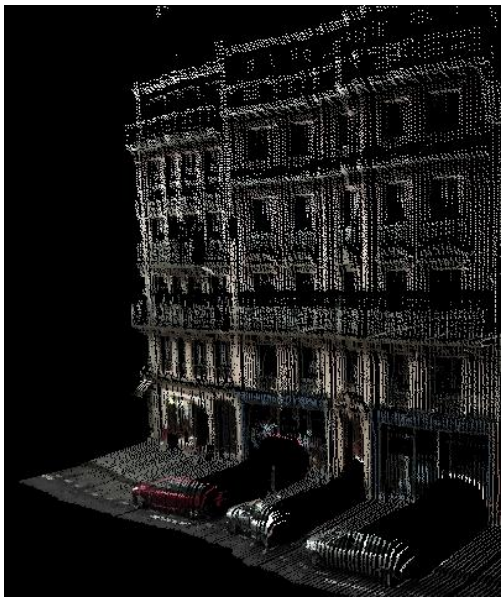


Figure 12: Result of the coloring of the laser range data with the fisheye image

### 4.4 Textures

**4.4.1 Texture Map Creation** With a 3D model using triangles to represent our urban environment, we can now add textures using an on-board camera. The car speed is typically 5km/h and the laser telemeter gives 1080 points in each profile at a $10Hz$ frequency. Two adjacent profiles can create 300 triangles with a decimation algorithm and a Delaunay triangulation. The car will have moved 15cm during this time. We proceed with triangle vertices in world coordinates including groups of 300 triangles at a frequency of 10Hz, with fish-eye images at a frequency of 25Hz and with the car position at a frequency of 10Hz. The car position is represented by rotation and translation matrices, which transform world coordinates into camera coordinates. In practice, GPS/INS fusion gives us matrices to transform laser coordinates into world coordinates. We have to invert those matrices and to add the transformation between laser coordinates and camera coordinates (it is done once with camera/laser calibration).

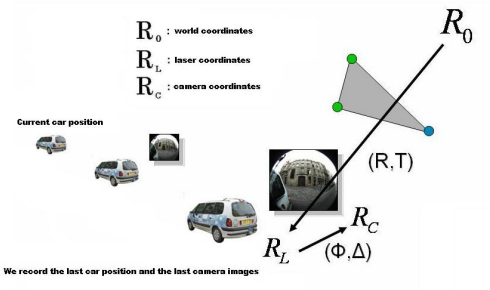
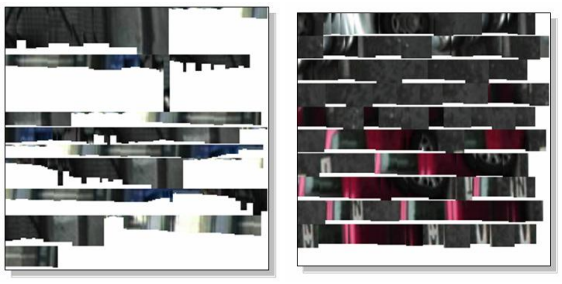



Figure 13: Projection of the texture on triangles

The same iterative algorithm is applied for every triangle. With world coordinates of vertices, we calculate coordinates in a chosen camera reference. In our application, for a triangle whose vertices belong to some different profiles, we have chosen to keep the image whose acquisition date is closest to the acquisition date of the profiles. This gives us the best camera resolution. All this is done in postponed real time (we calculate at the same speed of data acquisition but we have a delay of a few profiles). This requires recording the last car position and the last camera images. Then, we project the 3 vertices from camera coordinates on the fish-eye image using the principle of equidistant projection. We record the texture in texture maps and OpenGL can now display triangles and their associated textures in real time.

**4.4.2 Refinement and Storage optimization** Actually, we record textures in texture maps (BMP files) by surrounding triangles with rectangles. To save memory (it is important to display models faster), we make a classification of rectangles by grouping them into classes. A class is all rectangles with around same height (C. Rocchini and Montani, 1999). For example, with classification, we use only 108 texture maps instead of 302 texture maps for a street model (Figure 14).



(a) (b)

Figure 14: Texture file without (a) and with (b) classification

We use a technique to add precision in estimation of texture coordinates. We work with floats instead of integers with pixels. In the figure, we keep the blue triangle instead of the black triangle for the texture. Actually, with a $776x580$ fish-eye image, $10cm$ on a building front at a distance of $5m$ will be represented by 3 pixels on the image. Then we can understand why it is important to be in sub pixel resolution to calculate texture coordinates (inspired by (K. Nakamura and Ozawa, 2000), Figure 16).

**4.4.3 Results and Vizualization** To improve our results in visualization, we have smoothed triangle borders with a linear filter. Actually, we see a difference in luminosity between textures of two adjacent triangles. Errors in camera parameters can also give

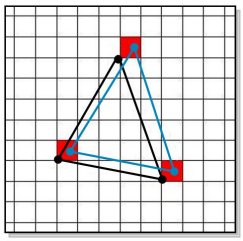


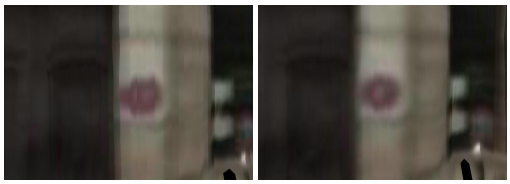
Figure 15: Subpixel resolution for textures

bad and unrealistic borders. When we apply a linear filter using textures of the two adjacent images we obtain a blurred texture that gives a less visible border (Niem and Broszio, 1995).

(a) Original image of the camera

(b) 3D textured model

Figure 16: Comparaison between the image and the textured model



(a) Detail without filters (b) Detail with filter

Figure 17: Details of the textured model

## 5 CONCLUSION

We have presented a new Mobile Mapping System mounted on a vehicle to reconstruct outdoor environment in real time. The scanning system is based on two sensors, a laser range finder and a camera equipped with a wide-angle fisheye lens. We have presented a new and faster way to determine parameters for the data fusionning between the camera and the laser scanner. We showed the creation of 3D coloured points or 3D textured triangulated models of the nearby environment of the vehicle.

## REFERENCES


Abuhadrous, I., Nashashibi, F., Laurgeau, C., Chinchole, M. and Goulette, F., 2003. Multi-sensor data fusion for land vehicle localization using rtmaps. In: Intelligent Vehicles Symposium.

Allen, P., Stamos, I., Gueorguiev, A., Gold, E. and Blaer, P., 2001. Avenue: Automated site modeling in urban environments. In: In Proc. of 3rd Conference on Digital Imaging and Modeling (3DIM'01).

C. Rocchini, P. C. and Montani, C., 1999. Multiple textures stitching and blending on 3d objects. In: Proc. Eurographics Workshop Rendering.

Farrell, J. and Barth, M., 1998. The global positioning system and inertial navigation.

Frueh, C. and Zakhor, A., 2003. 3d model generation for cities using aerial photographs and ground level laser scans. Computer Vision and Pattern Recognition.

Goulette, F., Nashashibi, F., Abuhadrous, I., Ammoun, S. and Laurgeau, C., 2006. An integrated on-board laser range sensing system for on-the-way city and road modeling. In: The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. 34, Part A.

K. Nakamura, H. S. and Ozawa, S., 2000. Generation of 3d model with super resolved texture from image sequence. In: Systems, Man, and Cybernetics.

Niem, W. and Broszio, H., 1995. Mapping texture from multiple camera views onto 3d-object models for computer animation. In: International Workshop on Stereoscopic and Three Dimensional Imaging.

Pless, R. and Zhang, Q., 2004. Extrinsic calibration of a camera and laser range finder. In: Intelligent Robots and Systems(IROS).

R. Dupont, R. K. and Fuchs, P., 2005. An improved calibration technique for coupled single row telemeter and ccd camera. In: 3D Digital Imaging and Modeling (3DIM).

Renato M. Hadad, A. De A Araujo, P. M., 2001. Using the hough transform to detect circular forms in satelliteimagery. In: Brazilian Symposium on Computer Graphics and Image Processing.

Stamos, I. and Allen, P., 2000. 3d model construction using range and image data. In: In. Proc Int. Conf. on Computer Vision and Pattern Recognition (CVPR).

Tsai, R. Y., 1987. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. IEEE Journal of Robotics and Automation RA-3,, pp. 323–344.

Zhao, H. and Shibasaki, R., 2003. A vehicle-borne urban 3d acquisition system using single-row laser range scanners. IEEE Trans. SMC Part B: Cybernetics.