Real-Time 3D Environment Reconstruction Using High Precision Trinocular Stereovision

Sergiu Nedevschi[1], Silviu Bota[1], Tiberiu Marita[1], Florin Oniga[1], Ciprian Pocol[1]
[1]Technical University of Cluj-Napoca,
{Sergiu.Nedevschi, Silviu.Bota, Tiberiu.Marita, Florin.Oniga, Ciprian.Pocol}@cs.utcluj.ro

Abstract - **This paper presents an implementation of a 3D environment reconstruction system, using trinocular (3 camera) stereovision. The system does not use rectification, in order to improve precision. Sub-pixel accuracy correlation is used. Feature extraction and correlation use MMX and SSE2 optimizations. Reconstruction correctitude tests were conducted using both synthetically generated images and camera acquired images.**

I. Introduction

Artificial vision systems can supply great amounts of information, useful for autonomous navigation systems, driving assistance systems and video teleconferencing systems. In order for this information to be effective, it has to capture the true 3D structure of the environment.

Due to the projection effect of the camera's lenses system, the 3D environment is transformed into a 2D image. In the projection transform the depth information is lost, i.e. the true position of each projected point along its corresponding projection ray cannot be recovered (Fig 1). All points on the OM_L projection ray are projected in the same point M_L, thus the real position of point M that generated the projection M_L is unknown. The loss of depth information prevents 3D environment reconstruction using a single camera.

Stereovision is a field of computer vision, which studies methods for recovering depth information by using two or more cameras instead of just one. The theoretical basis on which stereovision is build is epipolar geometry [1]. Epipolar geometry describes the configuration of stereo camera systems, using the pinhole camera model [1] and provides methods for 3D reconstruction.

Fig. 2. shows the geometry of a two camera system.

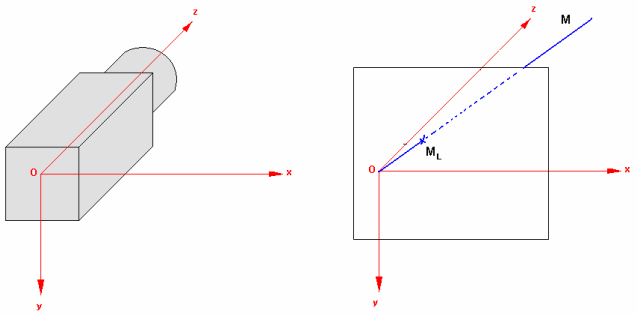

Figure 1. Left: perspective camera with its attached coordinate system. Right: Point M is projected into point M_L.

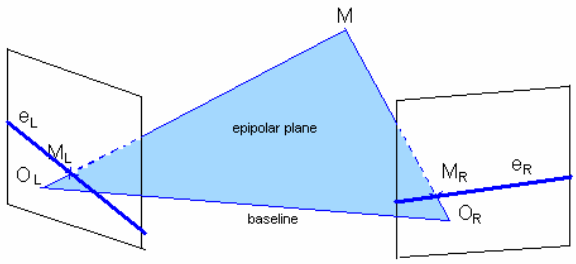

Figure 2. Epipolar geometry of a binocular system.

The projections of point M in the left and right camera's image planes are points M_L and M_L respectively. It is always possible to reconstruct the position of M if M_L and M_R are known.

The plane O_LMO_R is called the epipolar plane associated with point M and is determined by the optical centers of the cameras, O_L and O_M and the point M itself. The epipolar plane of course, contains ML and MR. The intersections of the epipolar plane with the left and right image planes are two lines e_L and e_R, the epipolar lines corresponding to the point M. The usual 3D reconstruction procedure is this [1]:

```
for each point M_L in the left image
  1. determine e_R, by intersecting plane O_L M_L O_R
     with the right image plane
  2. search for M_R on e_R (this step is usually
     known as left-right correlation)
  3. determine M by intersecting O_L M_L with O_R M_R
end for
```

The correlation step (2) of the reconstruction procedure is usually the most computational intensive (because it involves a search) and error prone. The instability of this step is caused by the relatively short baseline O_LO_R as compared to the lengths of the projection rays M_LM and M_RM which causes the projection rays to be almost parallel [2]. With a typical stereo system, for a point M situated at 100m in front of the stereo system, an error of just 1 pixel in the positioning of its determined projection point M_R yields a 20m error in the positioning of point M along the projection axis. The bottom line is this: for a stereo reconstruction algorithm to be precise, the correlation step must produce very precise results.

The precision of the correlation is affected by the presence or absence of high frequency image intensity components along the epipolar search line[2]. It is known that the higher the

1-4244-0361-8/06/$20.00 ©2006 IEEE

intensity frequency along the search line, the better the position of M_R can be determined, and thus the higher is the reconstruction precision. Lack of high frequency information, specific to un-textured areas, yields imprecise search matches.

There are situations where even the presence of high frequency intensity components is not sufficient. Repetitive patterns cripple the search process, by providing more than one possible candidates for the search [1]. Some areas that are visible in the left image are not visible in the right one, because other objects occlude them. Searching for an occluded point yields no match or, even worse, a wrong match. Specular surfaces tend to look different depending of the viewing angle, so they do not look the same in the left and right images [2].

Man made environments tend to consist in surfaces delimited by more or less straight edges. The surface itself has markings that have straight edges [3]. An edge has high frequency components on the direction perpendicular to the edge, and very low frequency components on the direction along the edge. As Fig. 3 suggests, searching on epipolar lines perpendicular to an edge gives precise results, while searching on epipolar lines parallel to the edge has poor results.

Usual stereo systems have a horizontal baseline and parallel optical axes, which means that epipolar lines are horizontal. These systems are capable of reconstructing vertical edges, but cannot precisely reconstruct horizontal edges.

Some stereo systems [4] generate their own artificial texture by projecting a pattern on the reconstructed area. These active reconstruction methods do not work well in uncontrolled environments and are not suitable for the purpose of autonomous navigation or driving assistance.

In this paper, we present a stereo system with three cameras, in a configuration that allows the reconstruction of all edges, irrespective to their orientation. The 3-camera configuration, as shown in Fig. 4, consists of two binocular pairs, one vertical and one horizontal. The vertical stereo pair has vertical epipolar lines and the horizontal pair has horizontal epipolar lines. The idea behind our 3 camera configuration is to reconstruct each edge using the appropriate stereo pair, i.e. the pair for witch the high frequency image intensity components along the direction perpendicular to the edge will be used most effectively [5]. Thus, horizontal edge points are reconstructed using the vertical (left-bottom) camera pair while the vertical edge points are reconstructed using the horizontal (left – right) camera pair (see Fig 5).

By comparing the experimental results of our trinocular 3D reconstruction method to the results obtained by using a similar 3D reconstruction method but only two cameras we will show the advantages of using 3 cameras.

The rest of the paper is organized as follows: Section II presents the feature extractors used to supply features needed for the correlation. Section III describes the sub-pixel accuracy correlation procedure. Section IV describes the 3D reconstruction method. Section V presents the experimental results we obtained. In section VI, we draw conclusions and present possible future work.

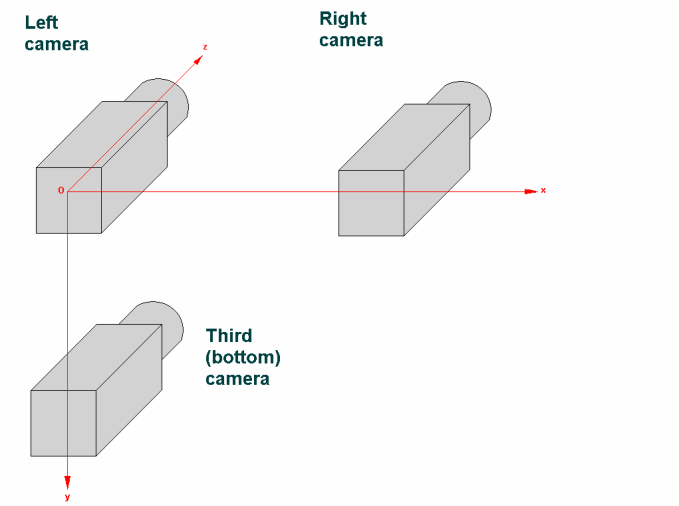

Figure 4. Our trinocular system configuration

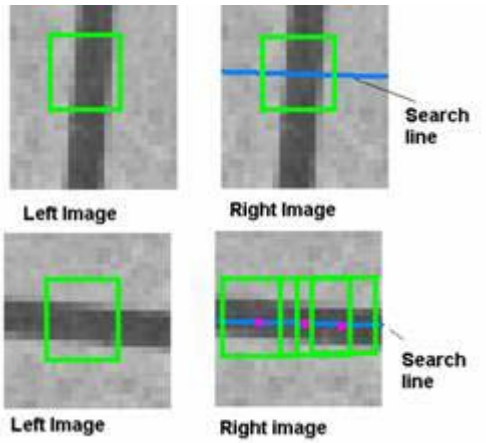

Figure 3. Searching on epipolar lines Top: epipolar line perpendicular to the edge generates a precise matches. Bottom: epipolar line parallel to the edge, poor matching

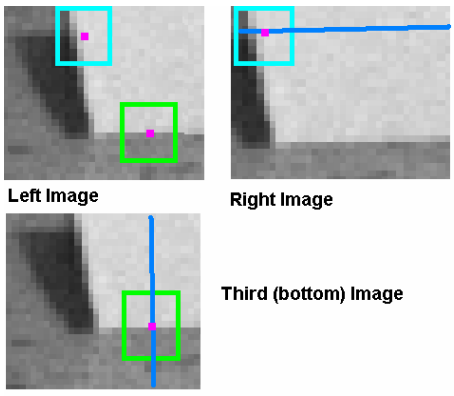

Figure 5. Correlation and reconstruction of edge point with different orientations using the appropriate camera pair

II. FEATURE EXTRACTION

The first step in any 3D stereo reconstruction algorithm is the selection of features to be matched (correlated) using a stereo camera pair. Features can be high level (line segments, ellipse arcs) or low level (points) [1]. We used edge points for our correlation features, as they are easily extracted and can be found in great numbers in most images. In addition, as shown in section I., the correlation is most precise in areas near edge points.

Because edges that are mostly horizontal are correlated in the top-bottom stereo camera pair and edges that are mostly vertical are correlated in the left-right stereo camera pair, we implemented two detectors, one for mostly horizontal and one for mostly vertical edge points. The image feature that gives the orientation of an edge is the image intensity gradient orientation.

Our two edge detectors are based on the Canny filter [11]. The steps of this filter are [2] (Fig. 6):

1. High-pass filtering for edge enhancement, using horizontal and vertical Sobel derivative masks, depending on the desired edge orientation. The results of this step are vertical and horizontal image intensity gradient components.
2. Non-maxima suppression, for thinning edges to a pixel width. Suppression is done horizontally for vertical edges and vertically for horizontal edges
3. Double thresholding for both edge orientations
4. Connection along vertical or horizontal direction, depending on the desired edge orientation.

The results of the edge detection algorithm are shown in Fig. 7. Some oblique edges are detected both as vertical and as horizontal.

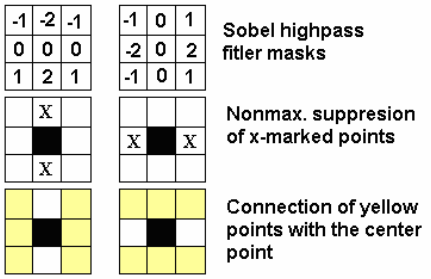

Figure 6. Edge detection, left for horizontal edges right for vertical edges

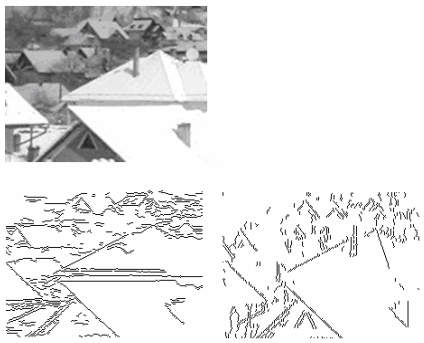

Figure 7. Top: original image. Left: horizontal edges. Right: vertical edges

That does not pose any problems, because these oblique edges can be reliably reconstructed using either the vertical or the horizontal camera pair, with the other pair used for validation.

Although edge detection is not the most computational intensive step in the 3D reconstruction algorithm, it is still a step worth optimizing. Because the algorithm's steps are straight forward, and no obvious algorithmic optimizations can be made, we turned our attention towards implementing the algorithm in assembly language, using, where possible, the SIMD features of the MMX and SSE2 equipped microprocessors [6], [7].

The following steps of the edge extraction algorithm where optimized:

1. Computing the horizontal and vertical image intensity gradients: 6 gradients are computed at once using MMX (Fig. 8 & 9) and 14 gradients at once using SSE2.
2. Non-maxima suppression for vertical edges: Again, computations are done for 6 points at once using MMX and for 14 at once using SSE2. The optimized algorithm is based on the fact that the maxima of the first derivative (gradient) correspond to positive to negative sign changes of the second derivative. Non-maxima suppression for horizontal edges is more easily parallelized and can be done straight forward, for 8 points in parallel using MMX and 16 points in parallel using SSE2.
3. Double thresholding is easily done for 8 points in parallel using MMX and for 16 points in parallel using SSE2, the same procedure being used for both horizontal and vertical edges.
4. Edge connection cannot be parallelized easily, so we implemented a highly optimized breadth first search procedure in assembly language, using a fixed sized, statically allocated queue.

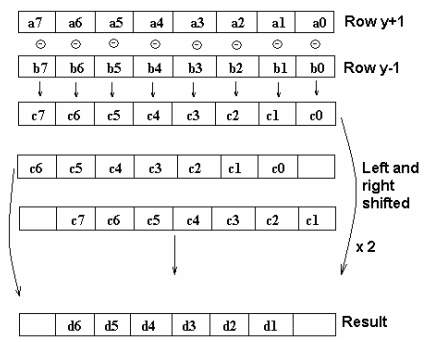

Figure 8 Vertical gradient computation using MMX

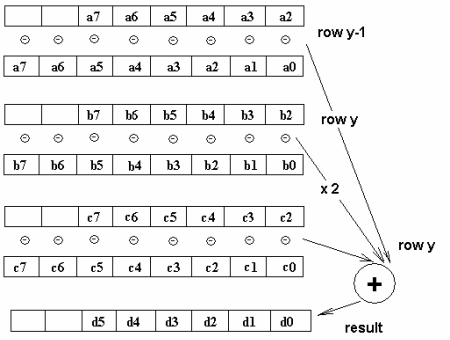

Figure 9. Horizontal gradient computation using MMX

III. CORRELATION

As discussed in section I, correlation is the most difficult step of the 3D environment reconstruction. It is highly computational intensive and error prone. Because we aimed for a real-time system, we made efforts to optimize the correlation module, even from the design phase. A number of choices must be made when designing the correlation module:

A. The features to correlate. As discussed in section I, we chose horizontal and vertical edge points as correlation features for the left-bottom camera pair and for the left-right camera pair respectively.

B. The measure of similarity between features. Comparing pairs of pixels is highly ambiguous, because there are many pixels with the same intensity value along the epipolar line and pixel intensity is also affected by noise [1]. The usual workaround is using a fixed-sized or variable-sized window, centered at the pixel of interest [1]. The use of variable-sized windows, although more precise, is also slower, so we chose fixed-sized windows. We also had to choose the size of the window. Small window sizes can give more precision near occlusion areas, but are also more ambiguous. Large windows are less ambiguous, but are also less precise near occlusions and comparing them is slower [2]. We chose a 7x7 pixel window size. Because of SIMD implementation issues, the window size is actually 8x7 pixels, extended one column to the right.

There are 3 widely used measures of similarity between pixel windows. In the order of accuracy these are: the normalized cross-correlation value, the sum of squared differences (SSD) and the sum of absolute differences (SAD). Although it is not the most precise, SAD is very fast, and has an MMX instruction to support it [6]. SAD ignores the brightness and contrast differences between images, but our acquisition system is capable of compensating brightness and contrast differences. The formula for SAD is:

$$SAD_{12}(x,y) = \sum_{i=y-3}^{y+3} \sum_{j=x-3}^{x+4} |I_1[i,j] - I_2[i,j]|, \quad (1)$$

where I_1 and I_2 are the images containing the windows to be compared.

C. The geometry used to compute epipolar lines and do correlation. The most widely used is the canonical configuration [1], in which the cameras are aligned so as the epipolar lines correspond to rows or columns of the images. In this configuration, the right image point corresponding to a left image point can be found on the same row in the right image as the left, and the bottom image point corresponding to the left image point can be found on the same column in the bottom image as in the top. The canonical configuration greatly simplifies the searching procedure. However, because the cameras and their sensors can never be perfectly aligned, a rectification step [2] is required, before the correlation. Rectification distorts the images, reprojecting them, to look as acquired by a canonical stereo system. Because rectifying the images requires re-sampling, sampling noise is introduced in the process. If advanced interpolation techniques (e.g. bi-cubic) are used, rectification becomes slower.

Our choice was to use a general configuration. In this configuration the epipolar lines have arbitrary orientations, deduced from the intrinsic and extrinsic camera parameters. The fundamental matrix is the transformation that links points in one stereo image to epipolar lines in the other. Each camera pair has its own fundamental matrix [1]:

$$\begin{aligned} p_L F_{LR} p_R^T &= 0 \\ p_L F_{LB} p_B^T &= 0 \end{aligned} \quad (2)$$

In (2), p_L is a point in the left image, p_R one in the right image and p_B a point in the bottom image, both expressed in homogenous coordinates. Therefore, the equations implicitly define epipolar lines in any image (by removing one point from any of the (2) equations we are left with the 3 coefficient vector of the corresponding epipolar line. A simple choice would be to clip such lines on the image edges and to search all along them. However, it is wasteful to search on the whole line. Usually, the geometrical locus of the correlated point is just a segment of this straight line. For example, if the optical axes of the cameras are not convergent, there is no point in searching the correlated point in the right image to the right of its corresponding point in the left image. An example of the true extent of the geometrical locus of correlated points is depicted in Fig. 10. We call the geometrical locus of the correlated points a *search line*, because that is the line on which correlated points are searched.

In order to compute the search line's extent we take the following steps, for each point p_L in the left image:

1. Back-project p_L, obtaining its associated projection ray, r_L
2. Clip r_L by 2 planes, of equations $z=z_{min}$ and $z=z_{max}$ (expressed in the left camera's coordinate system). This step is useful, because any camera has a minimum and maximum range, depending on the setting of its focus. We used a high z_{max} (cameras are usually accurately reproducing points at infinity). However, the choice of z_{min} has a big effect on the length of the search line. We used values in the range 2m-5m.
3. Transform the clipped r_L in the world's coordinate system, obtaining R_L.

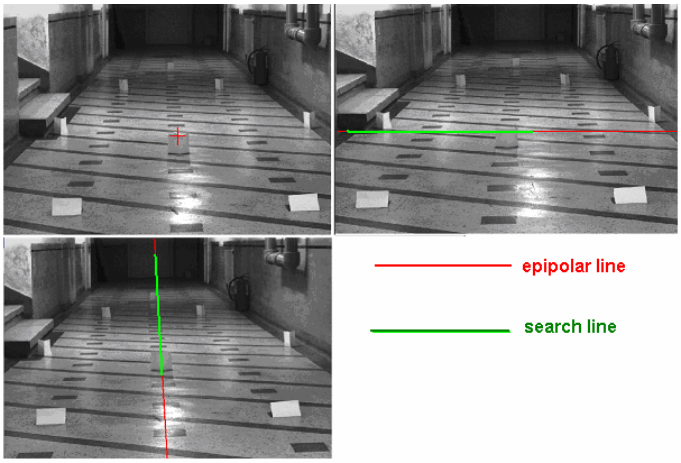

Figure 10. Epipolar and search lines in the right and bottom images corresponding to the marked point in the left image

4. Clip R_L with a six planes, of equations (in the world coordinate system): $x=x_{min}$, $x=x_{max}$, $y=y_{min}$, $y=y_{max}$, $z=z_{min}$, $z=z_{max}$. The clipping uses a Cohen-Sutherland [8] algorithm in 3D space. With this clipping, we restrict the reconstructed space to a subspace of interest
5. Transform R_L in the right camera's and bottom camera's coordinate system, obtaining r_R and r_B.
6. Clip r_R and r_B with a z_{min} and z_{max} plane, as in step 2.
7. Project r_R and r_B in their corresponding images, obtaining the right and the bottom search lines, s_R and s_B.
8. Clip s_R and s_B with a rectangle a little smaller than the image, so as the correlation window will not cross the image boundary.

Of course, all the above steps are computationally intensive. However, our algorithm works with image sequences, and the intrinsic and extrinsic parameters [1] that allow all the projections and coordinate transforms to be made are determined offline, and so are fixed for the duration of the image sequence. This means that the search lines can be pre-computed. We do not actually pre-compute the search lines, we compute them as needed, but we store the result, so we do not have to compute it again.

D. The last choice of the correlation algorithm is how precise the correlation should be. A correlation precision of one pixel is not suitable, as it yields large errors for distant points. We chose to use sub-pixel accuracy in our search. The correlation of edge point p_L in the left image is conducted as follows:
1. Determine if it is a vertical or horizontal edge point. If it is a vertical edge point, use the right camera for correlation. If it is a horizontal edge point, use the bottom camera for correlation. If it is an oblique edge point, use the left camera for correlation and the bottom (spare) camera for validation.
2. Obtain the (pre-computed) search line, as described in the above algorithm.
3. Use the Bresenham algorithm [8] to follow the search line. For each point on this search line, compute the SAD of the (fixed) left image window and the other (mobile) image window.
4. Search for the minimum SAD. If the minimum is not small enough, fail the search. This step deals with occlusions, i.e. the actual correlated point may not be visible at all.
5. If the global SAD minimum is not sufficiently lower than the other minima, located at least 3 pixels apart from it are, fail the search. This step deals with repetitive structures, and requires that there should be only a single pronounced minimum.
6. Use quadratic interpolation to refine the minimum position to sub-pixel accuracy. Fig. 11 shows the interpolation procedure.

The search across the search line is the bottleneck of the whole 3D environment reconstruction algorithm. Because of this, the whole procedure was written using assembly language. We used an MMX instruction capable of computing the SAD of 8 pixels at once [6]. This is the reason behind using an 8 pixel-wide window. Computing the SAD for the whole window is made using 7 such instructions, one for each row. Once the sub-pixel accuracy correlation is found, the only step remaining is to do the actual 3D reconstruction.

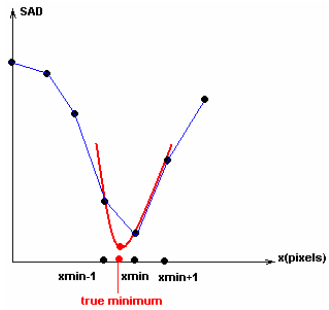

Figure 11. Sub-pixel accuracy interpolation

IV. 3D RECONSTRUCTION

In this section, we describe the actual 3D reconstruction algorithm. The algorithm is based on the fact that the 3D point that generated 2 correlated projection points must be located at the intersection of their projection rays [3]. Unfortunately, the projection rays do not usually intersect, because of the reconstruction error, i.e. the linear system of 4 equations, having the coordinates of the reconstructed point as unknowns is usually incompatible. There are many methods for dealing with this incompatibility. The most widely used methods are solving the system using least squares [1] and finding the middle-point of the common perpendicular of the 2 projection rays [2]. As computational complexity is the same in both methods, we chose the second method, because it has a more obvious geometrical interpretation (Fig 12). For oblique edge points, a further test is made, the reconstructed point is projected into the spare (bottom) camera image and its correlation SAD is evaluated. If the value is not small enough, the reconstructed point is discarded.

Once reconstructed, the points are passed to further processing modules, like grouping and tracking. However, these processing modules are not the scope of this article, and will not be discussed here.

We must say few words about the implementation of all the feature extraction, correlation and 3D reconstruction modules. Because the system is very complex, we made a C++, object-based implementation. For each processing class where assembly language optimizations were possible, we also developed a strict C++ reference implementation, which is easier to understand. Wherever possible, test classes where developed, to ease the error identification and to allow easier re-factoring of the system.

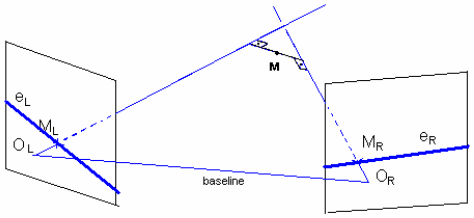

Figure 12. Point M is found at the middle of the common perpendicular of the 2 projection rays

V. EXPERIMENTAL RESULTS

We used 2 types of test images for our algorithm. One type were real images, the other were synthetic images, generated with the aid of a tool we developed. Using both type of images, reconstruction errors along the z-axis were measured (using measured controlled points for real images and rendering depth for synthetic images). The average reconstruction error is 3% along the z-axis. The reconstruction speed, for 752x576 pixels images was 20 frames/second on a 2.6GHz P4 system.

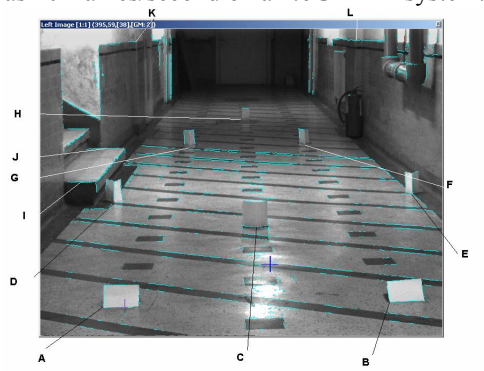

Figure 13. Corridor image with control points

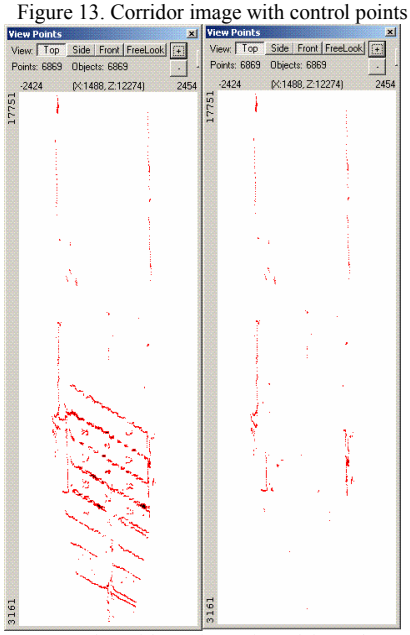

Figure 14. Reconstructed corridor points, viewed from the top. In the left image trinocular reconstruction was used, in the right only binocular reconstruction was used. The horizontal edges are clearly visible in the trinocular reconstruction, but not visible in the binocular reconstruction

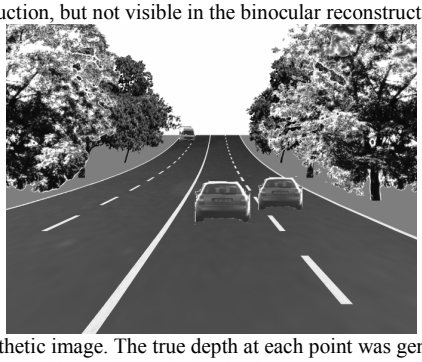

Figure 15. Synthetic image. The true depth at each point was generated in order to be compared with the depth obtained using the reconstruction algorithm

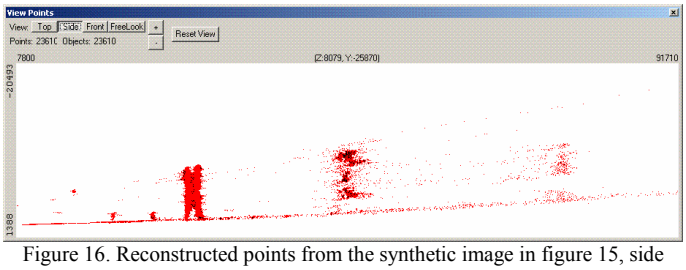

Figure 16. Reconstructed points from the synthetic image in figure 15, side view

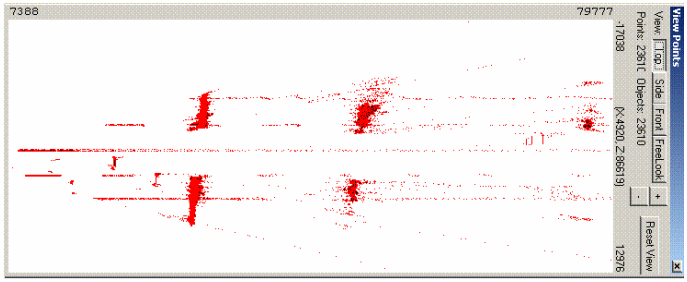

Figure 17. Reconstructed points from the synthetic image in figure 15, top view. One can see the lane markings, the trees and the car backs.

VI CONCLUSION AND FUTURE WORK

We developed a high precision, real-time 3D environment reconstruction system, using trinocular vision. We demonstrated the advantages of using trinocular stereo as opposed to the classic binocular stereo. We used non-rectified images and sub-pixel correlation accuracy in order to improve precision. We used MMX and SSE2 technologies to improve speed.

Future work on the system will be concentrated on multi-scale resolution correlation, using the motion field for correlation prediction and validation and reconstructing more points (dense stereo). Also, other trinocular system configurations should be tested [9][10], in order to evaluate their potential advantages.

REFERENCES

[1] E. Trucco, A. Verri , *Introductory Techniques for 3-D Computer Vision,* Prentice Hall, 1998

[2] D. A. Forsyth, J. Ponce, *Computer Vision: A Modern Approach,* Prentice Hall, 2002

[3] L. Shapiro, G. Stockman, *Computer Vision*, Prentice Hall, 2001

[4] S. M. Boersma, F. A. van den Heuvel, A. F. Cohen, R. E. M. Scholtens, *Photogrammetric Wound Measurement with a Three-Camera Vision System*, IAPRS, Vol. XXXIII, Amsterdam, 2000

[5] D. Murray J. Little, *Using real-time stereo vision for mobile robot navigation*, Workshop on Perception for Mobile Agents, CVPR'98

[6] *Intel Architecture Software Developer's Manual*, vol. 1 & 2

[7] *IA32 Intel Architecture Optimization*

[8] J. D. Foley, A. van Dam, S. K. Feiner, *J. F. Hughes, Computer Graphics: Principal and Practice in C*, 2nd Edition, Addison-Wesley 1995

[9] J. Mulligan, K. Daniilidis, *Trinocular Stereo for Non Parallel Configurations*, Proc. International Conference on Pattern Recognition (ICPR'00), Vol 1, pp 567-570, 2000.

[10] J. Migdal, *Depth Peception Depth Perception Using a Trinocluar Camera Setup and Sub-Pixel Image Correlation Algorithm,* Technical report, Mitsubishi Electric Research Laboratories Cambridge Research Center, May 19, 2000.

[11] J. Canny, *A Computational Approach to Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, June 1986, pp. 679-698