Stereovision on Mobile Devices for Obstacle Detection in Low Speed Traffic Scenarios

Alexandra Trif, Florin Oniga, Sergiu Nedevschi
Computer Science Department
Technical University of Cluj-Napoca
Romania
alexandram.trif@gmail.com, Florin.Oniga@cs.utcluj.ro, Sergiu.Nedevschi@cs.utcluj.ro

***Abstract*—Since smart mobile devices having capabilities of synchronous stereo image acquisition have been released on the market, the topic of real-time 3D environment reconstruction by stereovision on such mobile platforms has become of a greater interest among researchers. In this paper we continue the sparse stereovision approach proposed in [15], while focusing on improving the reconstruction results by refining the disparity computation accuracy to a sub-pixel level and by using the available sensors to gain more information about the position of the device relative to the world. After the 3D points are reconstructed by triangulation, a correction is applied on them to compensate for a possible pitch rotation of the device. Moreover, we present a fast approach for detecting the obstacle on the estimated trajectory of the vehicle. A series of experiments have been conducted which proved that although mobile development is constrained by the available features of the device and its operating system, sensor information is beneficial, and more importantly, both reconstruction accuracy and obstacle detection at short-medium distances and real-time processing can be achieved. Thus, developing driving assistance functions with such devices is possible for low vehicle speeds / short range scenarios, which often occur in urban environments.**

I. INTRODUCTION

The most important issue of a driving assistance application is to understand the environment in which the car is moving, more specifically to know where various objects are and whether they may interfere with the vehicle on its moving trajectory. Thus, with the aid of the stereo-cameras and the sensors available on the mobile device the objects existent in the field of view can be reconstructed and then transposed to the world reference frame. Moreover, the high speed and low power consumption requirements of mobile applications impose an efficient implementation of the stereovision algorithm. Thus a tradeoff must be made between the performance of the algorithm (accuracy, maximum depth estimation, etc.) and the complexity of the processing (speed, power consumption, etc).

Many stereovision approaches have been proposed in the last decade, and next, we will briefly revisit some of the most relevant ones. In [1] Hirschmüller et al. present a real-time correlation-based stereovision system, by analyzing and comparing the results of various matching cost functions. They propose two methods of reducing the number of matching errors and also a solution to improve matching at object borders. In [2] an edge-based stereo-reconstruction system is presented, with focus on improving the accuracy of obstacle detection at far distances. A dense stereo solution with sub-pixel accuracy is described in [3], based on the Semi-Global Matching (SGM) method [4] and using the Census transform as the matching metric. The SGM was introduced in 2005 by Hirschmüller, and it was proved to provide accurate stereo-matching results in real time, by searching for pixel correspondences based on Mutual Information and also approximating a global cost [4]. In [5] Stefano proposes another area-based matching approach, the Single-Matching Phase (SMP) local algorithm, which eliminates redundant differences and sums while computing the Sum of Absolute Differences matching cost. Even though extensive work has been done in the stereovision domain, none of these previously mentioned solutions were intended to run on smart mobile devices.

Recently some research has been directed towards the implementation on smart mobile devices of some computer vision algorithms, 3D reconstruction being an example. A very good analysis of the advantages and limitations of mobile devices regarding computer vision and augmented reality applications is presented in [9]. As it was already noted in [9] and [12], most of the solutions either use the mobile device only as a client, the processing being performed on a server [9], or the processing is applied offline on a set of previously captured images, the computational complexity thus playing a less important role [12]. However, there are a few mobile applications that perform the processing entirely on the device, some of which are shortly described in what follows.

In [10], Langguth and Goesele describe a solution for sparse 3D reconstruction of objects using structure from motion. Multiple pictures of the scene are captured from different view angles using the camera of a mobile phone, the user being guided by the application to move the device in the best position for a better reconstruction. In [11] another approach is presented, in which two smart phones are used to create a master-slave photometric stereo system. Thus, both devices capture images, but the slave device also uses its flash to illuminate the scene, while the master applies the photometric reconstruction algorithm. In [12], Pan et al. describe their solution of generating a 3D model of the scene from panoramic images in real time. An application for fast 3D reconstruction of house plans is described in [13]. The user is guided to capture data needed for the later creation of a virtual tour of the indoor environment.

In the last years, there has been a significant progress in terms of technical capabilities and processing power on the market of smart mobile devices (phones and tablets). Pushed forward by the entertainment industry, some manufacturers (so far LG and HTC, to our knowledge) have released smart mobile devices that have two forward-facing cameras, which can acquire synchronous stereo images.

In this paper we investigate whether such a smart device can be used as both acquisition and processing platform for stereovision based driving assistance, while also using the gravity sensor available on the device to improve the reconstruction results by eliminating the rotation applied on the 3D points by a potential pitch angle of the device. We also propose a real-time obstacle detection algorithm that selects the maximum agglomeration of points on the vehicle's trajectory and computes the mean distance towards it.

In section II an overview of the entire stereovision algorithm is presented, with emphasis on the sub-pixel interpolation methods used in stereo-matching and on the pitch angle correction applied to the reconstructed points. In section III the algorithm used for obstacle detection is explained. Finally, some experimental results are given in section IV.

II. Algorithm Overview

This paper presents a continuation of the algorithm presented in [15]. The major steps of the algorithm are based on the standard stereovision approach, as depicted in Fig. 1. Furthermore, some constraints are imposed in order to reduce the search space in the stereo-matching step and to improve the reliability of the found matching point. Also, the disparity computation in the matching step is refined to sub-pixel accuracy. All these steps have already been detailed in [15]. Additionally, a correction is applied on the reconstructed points in order to compensate for a possible pitch rotation. Moreover, different sub-pixel interpolation methods have been tested and analyzed.

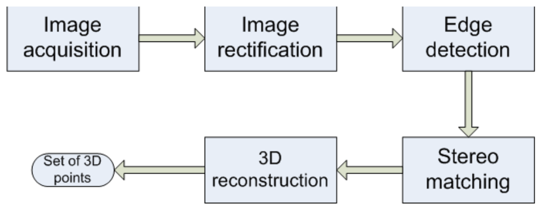

Fig. 1. Algorithm overview

A. Stereo-matching

Due to the low computational complexity requirements of the application we considered an edge-based stereovision approach to be the most appropriate. Therefore, after the images are rectified using the approach described in [6], edges are detected in the left image using the well-known Canny algorithm. The left image is then used as reference in the stereo-matching step, and the Sum of Absolute Differences is used as matching metric. A pseudo-code of the stereo matching function is presented below, and it was detailed in [15]:

Function Stereo-matching

```
1: for each edge-point in the left image do
2:   compute the gradient magnitude $m_L$
3:   set gradient threshold $t \leftarrow m_L/2$
4:   for each point $p$ in the disparity range in the right image do
5:       compute the gradient magnitude $m_R$
6:       if $m_R > t$ then
7:           compute $SAD_p$
8:           add $SAD_p$ to SAD-list
9:           if $SAD_p < SAD_{min}$ then $SAD_{min} \leftarrow SAD_p$ end if
10:      end if
11:  end for
12:  apply constraint for repetitive patterns
13:  if not repetitive pattern then
14:      apply sub-pixel interpolation
15:  end if
16:  add current match to matches-list
17: end for
18: return matches-list
```

B. Sub-pixel accuracy

Depth computation depends on the disparity between the pixels in the left and right images. Therefore, if the matching results are improved to reach a sub-pixel accurate level, this will also be reflected on the refinement degree of the depth estimation results. Sub-pixel accuracy is necessary because the matching point in the right image is not always located on an integer pixel, but rather between two pixels.

Sub-pixel accuracy can be achieved by interpolating the matching cost of the best-matching pixel and its two left and right neighbors. There are a few interpolation methods described in literature, which have been implemented and tested in our solution. One of them is parabola interpolation, in which a parabola is fitted on the values of the correlation cost function corresponding to the best-matching pixel (SAD_{min}) and its left (SAD_{left}) and right (SAD_{right}) neighbors. The displacement d_s from the integer coordinate of the previously found correspondent pixel is computed as:

$$d_s = \frac{SAD_{left} - SAD_{right}}{2(SAD_{left} + SAD_{right} - 2SAD_{\min})} \tag{1}$$

Another interpolation function that can be used is the symmetric "V". According to [8], the displacement d_s is computed as:

$$d_s = 0.5 - \frac{MIN\left(SAD_{left} - SAD_{\min}, SAD_{right} - SAD_{\min}\right)}{2MAX\left(SAD_{left} - SAD_{\min}, SAD_{right} - SAD_{\min}\right)} \tag{2}$$

Furthermore, in order to improve the accuracy of sub-pixel estimation, we implemented the histogram equalization approach presented in [14]. As already noted in [14] sub-pixel displacements tend to be biased towards integer values, which can be observed from the histogram of sub-pixel displacements. Thus, the histogram equalization approach tries to distribute the sub-pixel values on the entire interval of displacements [-0.5, 0.5]. Due to the fact that the histogram is symmetric around 0, this solution considers the sub-pixel displacement as a random variable x in the range [0, 0.5] and estimates its probability density function as a linear function when the histogram of sub-pixel displacements was generated using symmetric "V" interpolation. The transformation function is then given by the cumulative distribution function of *x*. With this method, the obtained function for computing the sub-pixel displacement is the following:

$$(3)$$

where

$$(4)$$

The histograms we obtained can be seen in Fig. 2 and a comparison between the effects the different sub-pixel estimation methods have on the reconstructed points will be presented in the *Experimental results* section.

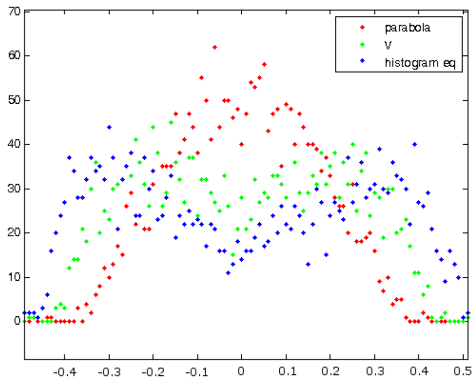

Fig. 2. Histograms of sub-pixel displacements obtained using parabola interpolation, symmetric "V" interpolation and histogram equalization

C. 3D reconstruction

The last step of the algorithm is represented by the 3D reconstruction operation. Because the images are rectified, we can use the well-known formulas for 3D reconstruction in a canonical configuration, which are widely available in literature [7].

Due to the fact that the mobile device is mounted on the dashboard in the car every time the application is used, a different pitch angle may be introduced every time the device's position is adjusted. Thus, considering the fact that the points are reconstructed relative to the camera reference frame, they may appear tilted in the 3D space, as it can be seen in Fig. 4b. As a consequence, an auto-calibration of at least the pitch angle is necessary in order to rotate the points back to their correct position in the 3D space, as in Fig. 4c.

Although the definition of a pitch is a rotation about the y-axis, we call a pitch a rotation about the horizontal axis of the device, which is the x-axis.

In order to apply the pitch correction on the 3D points, we first need to know the pitch angle α. For this we make use of the gravity sensor available on the mobile device. This sensor provides the force of gravity that is applied to the device on the x-, y- and z-axes, in m/s^2. Because the rotation is performed about the x-axis, the pitch angle is computed as:

$$(5)$$

where *Gz* and *Gy* are the components of the gravity on the z- and y-axis, respectively, as depicted in Fig. 3. Further, once we know the pitch angle, the new coordinates of the points are computed using the following formulas:

$$(6)$$

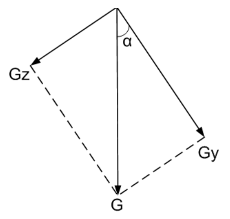

Fig. 3. The components of the gravity G on the y- and z-axes

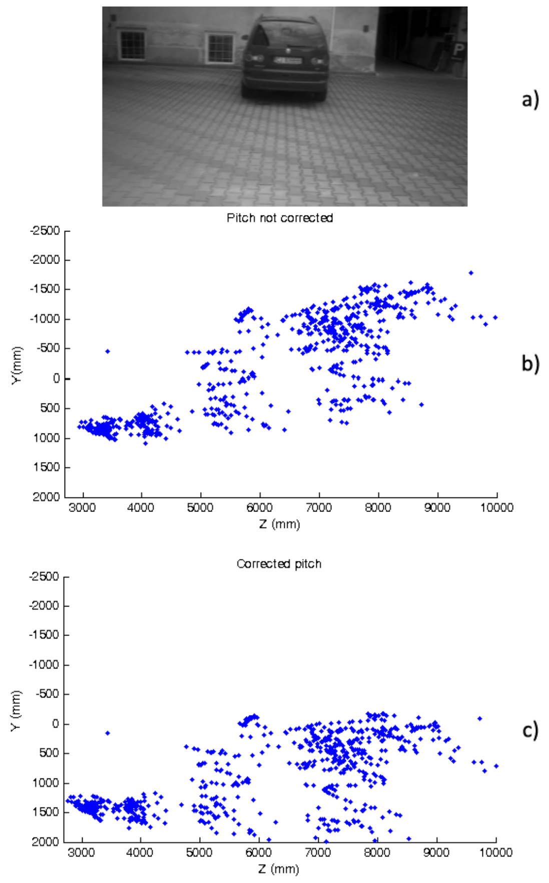

Fig. 4. a) The left image from the stereo-pair, acquired with a 10 degree pitch angle. b) Left view of the reconstructed 3D points, before applying the pitch angle correction. c) Left view of the reconstructed 3D points, after applying the pitch correction.

III. Obstacle detection

Once we have a better sense about the location of object points in the 3D space, we can eliminate those points that are outside the vehicle's trajectory, and we can also determine whether there is an obstacle on that trajectory. Our obstacle detection approach is using the histogram of point depths in the scene to determine at what distance the maximum agglomeration of points is located. Thus, the algorithm can be split in the following steps: vehicle trajectory estimation, building the histogram of depths and locating the obstacle. These steps will be described in more detail in what follows.

A. *Trajectory estimation*

Assuming that the vehicle is moving along a curvilinear path, in order to estimate its trajectory we need to know the radius of the curve on which it is moving. Thus, from the physics of circular motion we know that the car has a centripetal acceleration, whose formula is the following:

— (7)

where a_c is the centripetal acceleration, v_t is the velocity of the car and R is the radius of the curve on which the car is moving. Therefore, if we know the acceleration a_c and the velocity v_t, from (7) we can compute R.

The velocity v_t can be determined by using the GPS available on the mobile device. Along with the current location coordinates, the GPS also provides information about the moving speed in m/s. The velocity could also be taken from the vehicle's CAN bus via Bluetooth. Beside the speed, the yaw-rate of the vehicle can also be obtained from the CAN bus, which can be used in the computation of the radius R.

In order to find out the acceleration a_c, we can use the linear acceleration sensor of the device. This sensor provides the acceleration on the three axes (x, y, z) without taking into consideration the acceleration due to gravity (as opposed to the accelerometer which does not discard the gravity from its results). Due to the fact that the device is placed in the car, we know that the centrifugal force is acting on it. The centrifugal and centripetal forces and also their acceleration components are equal in magnitude, so we can approximate the centripetal acceleration that we need with the lateral acceleration applied on the x-axis of the device. Once we know the radius R, we can select all the points that lay in a tunnel along the moving path by computing their distance towards the center of the curve.

B. *The histogram of point depths*

In order to simplify the histogram computation process, all the points on the curved path are unrolled along a straight line. We consider the vehicle to be positioned in the point $V(0,0,0)$.Thus, for every point $P(x_p, y_p, z_p)$ in the tunnel, we compute the angle α determined by the center of the curve, the point P and the point V using the law of cosines in the PCV triangle, as depicted in Fig. 5 on the left. Then the length of the arc determined by V and the projection of P on the circle is computed as:

(8)

Further, the new position of point P along the transformed straight path is computed as:

$$P(x_p,\ y_p,\ z_p) \rightarrow P(-\delta,\ y_p,\ L_\alpha) \qquad (9)$$

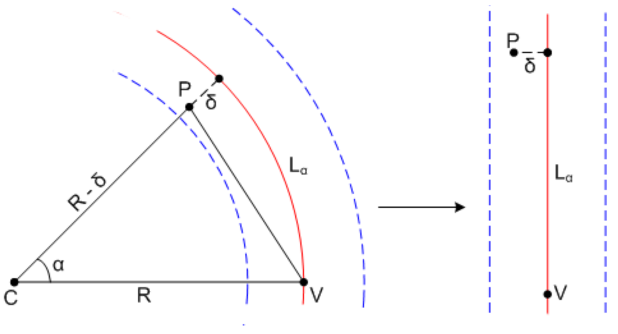

Fig. 5. Unrolling the points in the tunnel along the path of the vehicle. The red continuous line is the curved trajectory of the vehicle. The blue dashed lines are delimiters for the tunnel in which obstacles are searched for. V is the vehicle, C is the center of the curve, P is a point in the tunnel.

The histogram is then computed by using windows of 10 cm in length along the straight trajectory of the car, and counting the points which lay in each window. Due to the fact that objects which are more far away have less corresponding reconstructed edge points, the histogram is normalized to compensate for this difference in the number of reconstructed points. Considering that most obstacles on the road are vehicles, the normalization function we use depends on the area of the projected object in the image:

—— (10)

Thus, assuming that the sizes of the vehicles are somewhat constant, the area will depend on f^2/Z^2 and we can use the inverse of this coefficient to normalize the histogram.

C. *Obstacle detection*

In order to detect the obstacle on the road, we traverse the normalized histogram and search for the maximum agglomeration of points. Moreover, based on our observations from the *Experimental results* section which state that the dispersion of the points around the mean increases with the distance towards the object, we considered using an adaptive window in our search for the obstacle in the histogram. Thus, the size of the window depends on the distance at which we are currently searching, and a formula for computing the window size at every step is given by:

—— (11)

where Δd is the disparity error, Z is the distance, b is the baseline and f is the focal length of the cameras. We considered Δd to be 0.25.

IV. Experimental results

The application was tested on an LG V900 Optimus Pad device, which has an Nvidia Tegra 2 chipset including a dual-core ARM Cortex A9 CPU. The baseline of the stereo camera system is 4.5 cm.

In order to test the reconstruction accuracy of the system, we acquired a sequence of left-right pairs of images of a car located at measured distances in the range 2 m – 9 m. The 3D coordinates were computed in the left camera reference frame, and the mobile device was mounted horizontally using a laser level (to ensure alignment with the ground plane) and aligned with the test vehicle longitudinal axis.

Firstly, we tested the depth estimation accuracy of our system when using different sub-pixel interpolation methods and we compared the effects they have on the reconstructed points. For this test we selected only the points that lay on the car and computed their mean and standard deviation on the z-axis. The reconstruction results and the images used in our experiments can be seen in Fig. 6.

In Table I the results obtained using parabola interpolation are presented. The results obtained using symmetric "V" and histogram equalization can be found in Tables II and III, respectively. When analyzing the mean of the reconstructed points, we observe that the chosen interpolation method does not affect it significantly. Up to a distance of 6 m the mean distance is accurately determined, at 7 m we have an error of only 30-40 cm, at 8 m the error is 50 cm and at 9 m the error increases to 1.3 m.

The standard deviation of the reconstructed points describes how far away from the mean they are scattered. We noted in [15] that as the distance grows, the points are more spread out on the z-axis, as it can be seen in Fig. 6.

TABLE I. ACCURACY OF DEPTH ESTIMATION WHEN USING IMAGES OF SIZE 384X216 AND THE PARABOLA SUB-PIXEL INTERPOLATION FUNCTION

Measured distance (m)	**Mean (mm)**	**Standard deviation (mm)**	**Number of edge points**
2	2051	94	769
3	2923	188	557
4	3968	375	360
5	4997	534	296
6	5942	828	194
7	6778	996	204
8	7530	508	131
9	7717	626	90

TABLE II. ACCURACY OF DEPTH ESTIMATION WHEN USING IMAGES OF SIZE 384X216 AND THE SYMMETRIC "V" SUB-PIXEL INTERPOLATION FUNCTION

Measured distance (m)	**Mean (mm)**	**Standard deviation (mm)**
2	2042	88
4	3997	415
6	5913	779
7	6694	927
8	7536	692

TABLE III. ACCURACY OF DEPTH ESTIMATION WHEN USING IMAGES OF SIZE 384X216 AND THE HISTOGRAM EQUALIZATION SUB-PIXEL INTERPOLATION

Measured distance (m)	**Mean (mm)**	**Standard deviation (mm)**
2	2035	85
4	4017	444
6	5879	761
7	6603	877
8	7522	791

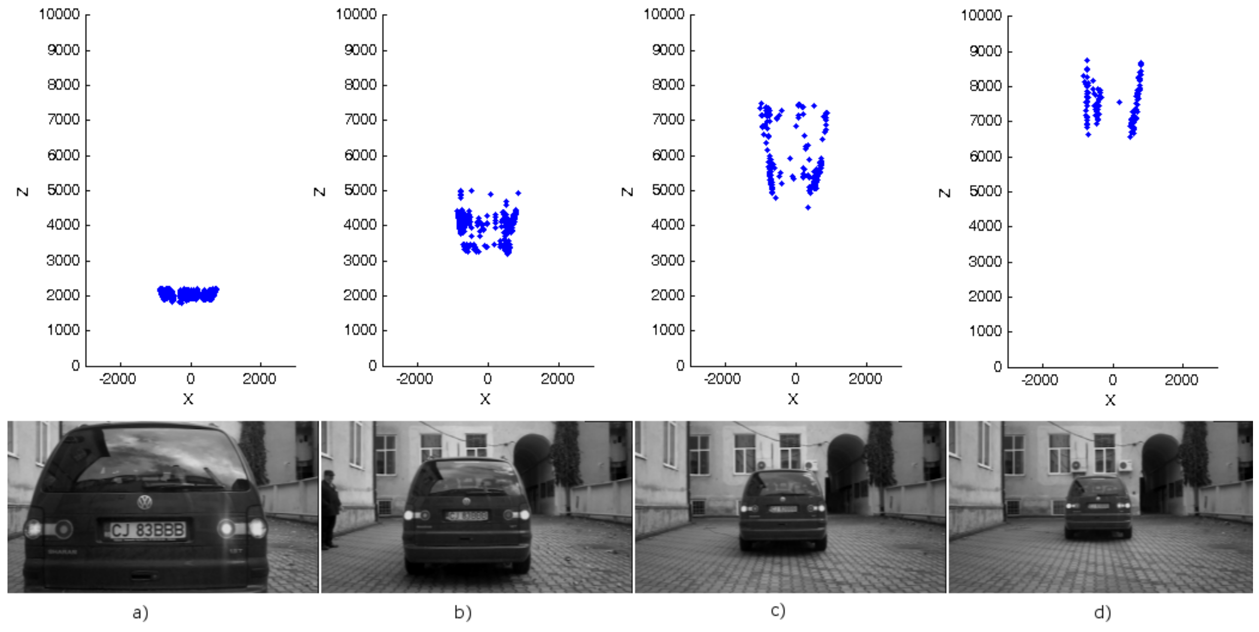

Fig. 6. Depth estimation results (top) and the corresponding 384x216-sized left image from the stereo-pair used for reconstruction (bottom): a) the car is at 2 m; b) the car is at 4 m; c) the car is at 6 m; d) the car is at 8 m.

From the tables we observe that when we use the symmetric "V" interpolation function we achieve an approximately 6% reduction of the points' dispersion around the mean, while the histogram equalization method improves this percentage to about 9-10%. However, in some cases, the standard deviation of the points worsens, the 4 m case being such an example.

The main causes for this dispersion of the points around the mean include the blurring introduced by the rectification step, as the final pixel intensity is obtained by interpolation, the limited accuracy of the sub-pixel interpolation (1/4 to 1/6 pixels, as noted in [2]) and the small resolution of the images. Thus, a small sub-pixel error could lead to greater errors in depth estimation.

For testing the obstacle detection algorithm, we acquired a sequence of images in traffic. The results we obtained can be seen in Fig. 10 and Fig. 9. In Fig. 10 the estimated trajectory of the vehicle is drawn with a yellow line, as it was moving along a curved road. In Fig. 9 the obstacle detection results are depicted, while also measuring the distance towards the detected object on the vehicle's path.

Concerning the performance of the application, we managed to achieve an average speed of 6.5 frames per second on 384x216 grayscale images when using a window of 7x7 in the stereo-matching function. Our solution for detecting the obstacle on the vehicle's trajectory proved not to be time-consuming, as it only reduces the processing speed to an average of 6 frames per second. Taking into account that recent smart mobile devices (including our test device) feature dual core processors, we performed an analysis of a potential multi-threaded implementation. Due to the fact that the affinity of a thread cannot be set to a certain processor programmatically, and moreover, there are a number of other services running in the background, the average improvement in speed that can be obtained by splitting the processing on two threads is approximately 20%.

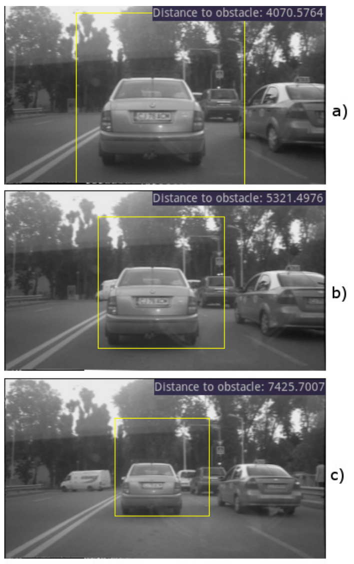
Distance to obstacle: 4070.5764
a)
Distance to obstacle: 5321.4976
b)
Distance to obstacle: 7425.7007
c)

Fig. 9. Obstacle detection results: the yellow rectangle marks the slice of the tunnel where the obstacle was detected:
a) obstacle detected at 4 m;
b) obstacle detected at 5.3 m;
c) obstacle detected at 7.4 m.

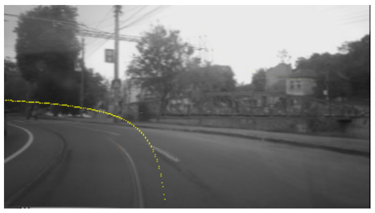

Fig.10. The estimated trajectory of the vehicle (the yellow line) while moving along a curved road

References

[1] H. Hirschmüller, P. R. Innocent, J. M. Garibaldi, "Real-time correlation-based stereo vision with reduced border errors", *International Journal of Computer Vision*, 47(1/2/3):229-246, April-June 2002.

[2] S. Nedevschi et al., "High Accuracy Stereo Vision System for FarDistance Obstacle Detection", *IEEE Intelligent Vehicles Symposium,* June 14-17, 2004, University of Parma, Parma, Italy, pp. 292-297.

[3] I. Haller, C. Pantilie, F. Oniga, S. Nedevschi, "Real-Time Semi-Global Dense Stereo Solution with Improved Sub-Pixel Accuracy", *Proceedings of 2010 IEEE Intelligent Vehicles Symposium*, June 21-24, 2010, University of California, San Diego, CA, USA, pp. 369 - 376.

[4] H. Hirschmüller, "Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* CVPR'05, vol. 2, pp. 807-814, June 2005.

[5] L.D. Stefano, M. Marchionni, S. Mattoccia, "A fast area-based stereo matching algorithm", *Image and Vision Computing*, vol. 22, 2004, pp. 983-1005.

[6] C. Vancea, S. Nedevschi, "Analysis on Different Image Rectification Approaches for Binocular Stereovision Systems", in *Proceedings of 2006 IEEE ICCP*, September 1-2, 2006, Cluj-Napoca, Romania, vol. 1, pp. 135-142.

[7] E. Trucco, A. Verri, "Introductory Techniques to 3D Computer Vision", Prentice Hall, 1998.

[8] J. Woodfill et al., "Data Processing System and Method", U.S. Patent 6,215,898 B1, April 10, 2001.

[9] C. Arth, D. Schmalstieg, "Challenges of Large-Scale Augmented Reality on Smartphones", *ISMAR 2011 Workshop: Enabling Large-Scale Outdoor Mixed Reality and Augmented Reality*, October 26, 2011, Basel, Switzerland.

[10] F. Langguth, M. Goesele, "Guided Capturing of Multi-view Stereo Datasets", *Eurographics*, 2013.

[11] J.H. Won, M.H. Lee, I.K. Park, "Active 3D Shape Acquisition Using Smartphones", *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, June 2012.

[12] Q. Pan et al., "Rapid Scene Reconstruction on Mobile Phones from Panoramic Images", in *Proceedings of 2011 IEEE ISMAR*, 2011, pp. 55-64.

[13] A. Sankar, S. Seitz, "Capturing indoor scenes with smartphones", in *Proceedings of the 25th annual ACM symposium on User interface software and technology (UIST '12)*, 2012.

[14] C. D. Pantilie, S. Nedevschi,"SORT-SGM: Subpixel Optimized Real-Time Semiglobal Matching for Intelligent Vehicles", *IEEE Transactions on Vehicular Technology*, 61(3): 1032-1042, March 2012.

[15] F. Oniga, A. Trif, S. Nedevschi, "Stereovision for Obstacle Detection on Smart Mobile Devices: First Results", accepted at *IEEE Intelligent Transportation Systems Conference 2013*, Netherlands, 6-9 October 2013