Stereovision for Obstacle Detection on Smart Mobile Devices: First Results

Florin Oniga, Alexandra Trif and Sergiu Nedevschi

Abstract — **Recent market deployment of smart mobile devices which feature synchronous stereo image acquisition has raised the question whether such devices can be used for real-time 3D environment reconstruction by stereovision. In this paper, we propose a stereovision approach that can run in real-time on smart mobile devices, and we evaluate its potential for developing driving assistance functions. The stereo approach is a sparse approach: edges are detected in the left image, correspondent right image points are determined using area-based matching, and each left-right pair of points is mapped in 3D by triangulation. Our experiments have proved that despite the limitations imposed by the mobile device, both reconstruction accuracy 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

Depth information about the objects in the environment is of a great importance in driving assistance applications, and more specifically in obstacle detection. Moreover, the algorithm needs to be efficiently implemented in order to meet the high speed and low power consumption requirements of mobile applications. In order to achieve widespread deployment of a stereovision based system, a tradeoff must be made between the accuracy and maximum depth performance of the stereo-reconstruction algorithm and the required hardware complexity (power, stereo system size, etc.) of the cameras and the processing unit.

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 obstacle detection at far distances with high accuracy. 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]. 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.

As already noted in [9] and [12], there are very few mobile device applications that perform the entire processing on the device. The majority of the solutions are either client-server applications, in which most of the processing is done on a server [9], or they perform an offline reconstruction from a set of previously captured images, without taking into consideration the computational requirements [12].

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. We propose a solution for edge-based stereo-reconstruction tailored for mobile devices, and, after the experimental evaluation of the system,

A. Trif, F. Oniga and S. Nedevschi are with the Computer Science Department, Technical University of Cluj-Napoca, Romania (corresponding author phone: +40-264-401457, e-mail florin.oniga@cs.utcluj.ro).

This work was supported by the Romanian Executive Agency for Higher Education, Research, Development and Innovation Funding (UEFISCDI), project number PNII – PCCA 18/2012.

we discuss the possibility of using such devices for driving assistance.

In section II an overview of the entire stereovision algorithm is presented. In section III the stereo-matching function is described in more detail. The choice of the correlation cost function, the constraints applied on the discovered matches and the method used for sub-pixel accuracy are explained in further subsections. Experimental results are then given in section IV. Finally, a critical discussion about the system benefits and limits is presented in section V.

II. Algorithm Overview

The major steps of the stereovision algorithm are depicted in Fig. 1 and are further detailed in what follows. The algorithm is more or less based on the standard steps, with an additional measure for reducing the search space for the right image correspondent.

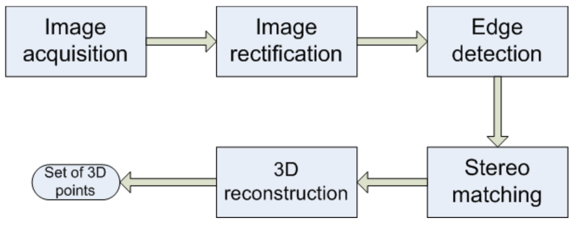

Figure 1. Algorithm overview

First of all, we calibrated the cameras in order to obtain the intrinsic and relative extrinsic camera parameters. The left camera was considered to be the world reference frame. These parameters are later needed for the image rectification and 3D reconstruction steps. The calibration was performed using the Caltech Camera Calibration Toolbox for Matlab.

Although the built-in cameras of the device are almost canonical, rectification is needed on the acquired images in order to simplify the correspondence searching step and to reduce the amount of computations required for 3D reconstruction, thus improving the processing speed. We chose to implement the rectification approach presented in [6], which divides the algorithm in two steps: an offline step, performed only once, and an online step applied to each frame. In the offline step, the new parameters of the canonical system and the rectifying matrix are computed, and four look-up tables are built, two for each image, having the size of the image. For each image, one look-up table will store the integer coordinates of the correspondent pixel in the original image, the actual coordinates being computed with sub-pixel accuracy, and the other will store the interpolation coefficients that will be used in the online step. Thus, the online step is reduced to the computation of the rectified pixel intensities using bilinear interpolation of the four neighboring pixels in the original image with the values stored in the look-up tables.

Together with the rectification step, a un-distortion operation is applied on the images, which is needed to correct the distortions introduced by the camera lenses. This operation is included in the offline computation of the look-up tables, and thus no overhead is added to the online processing of the frames.

The next step consists of selecting the relevant left image features for 3D reconstruction. Considering the need for low computational complexity, we use an idea already present in literature, which is to select edges as relevant features. Extracting edges in the left image was done using the Canny edge detection algorithm. The left image will be then used as reference in the correspondence searching step. For every edge point in the left image, its homologous point in the right image is searched on the horizontal epipolar line, and then an interpolation function is applied on the matching costs in order to achieve sub-pixel accuracy. Stereo matching will be described in more detail in the next section.

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]:

$$Z = \frac{b * f}{d} \tag{1}$$

where d is the disparity, b is the baseline and f is the focal distance.

III. Stereo Matching

As previous research has shown [1][3-5], stereo matching is the most important and computationally intensive step of the stereo-reconstruction algorithm. Therefore, a series of constraints need to be applied in order to reduce the search space and also ensure that the number of false matches is reduced without affecting the number of good matches. A pseudo-code of the stereo matching function is the following:

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
```

The presented pseudo-code is explained in more detail in what follows.

A. *The matching cost function*

Many correlation metrics exist in literature: the normalized cross-correlation (NCC), the sum of squared differences (SSD), the sum of absolute differences (SAD) and so on. According to [1], SAD is the fastest and also provides better results compared to SSD or NCC. Therefore, we considered SAD as being more appropriate for a fast implementation. The function is applied on a window of size n x n around the pixels to be matched. For every pixel, the differences between the intensities of the corresponding pixel positions in the window in the left I_L and right I_R images are added to a matching cost variable. The best matching pixel in the right image is chosen to be the one which minimizes this cost.

The choice of the window size is very important: a small window (e.g. 3x3) implies a smaller number of computations for the matching cost, but it will yield less accurate results and a lot of false matches [1], whereas a larger window (e.g. 7x7) will affect the processing speed but will provide more accurate results. We chose to use a 7x7 window for matching.

B. *Constraints*

In some cases, due to differences in illumination or repetitive patterns, the pixel in the right image which minimizes the correlation function might not be the correspondent of the edge point detected in the left image. Therefore, a series of constraints need to be applied to reduce the number of false matches. Moreover, these constraints also have the benefit of reducing the computational time by eliminating from the search space those points which will most likely not be a valid correspondent.

Firstly, we reduce the search space by looking for the correspondent in a limited range of disparities from the current point. As the baseline is 4.5 cm and the obtained (calibration) focal distance is approximately 404 pixels, our chosen disparity range of 13 pixels ensures matching of objects at distances greater than 1.5 m from the mobile device. This minimum distance is sufficient for driving assistance applications, as the objects in the environment are usually located at greater distances (if the mobile device is placed on the dashboard, near the windshield, while the closest objects can be located near the front bumper).

Secondly, the left image features are edge points, therefore exhibiting a significant value of the gradient magnitude. The right image correspondent should present a similar gradient value, slightly different (if any) due to contrast differences between the stereo pair. Thus a condition based on the gradient magnitude is applied. We first compute the gradient in the x and y directions of the pixel in the left image by convolving the rows and columns with the mask:

$$[1\ 0\ {-1}].$$

Then we compute the gradient magnitude m_L using the Manhattan metric (faster than the Euclidian metric):

$$m_L = |g_{xL}| + |g_{yL}|, \tag{2}$$

where g_{xL} and g_{yL} are the gradients in the x and y directions, respectively. We then set a threshold value at half of the previously computed magnitude and we choose to match only those points in the right image which have the gradient magnitude greater than this threshold. This constraint is graphically represented in Fig. 2.

The next constraint tries to eliminate the ambiguous matches of repetitive patterns. We compare the matching cost of the best match with all the other costs in the disparity range. If the global minimum is not smaller with at least 20% than all local minimums, the match is rejected.

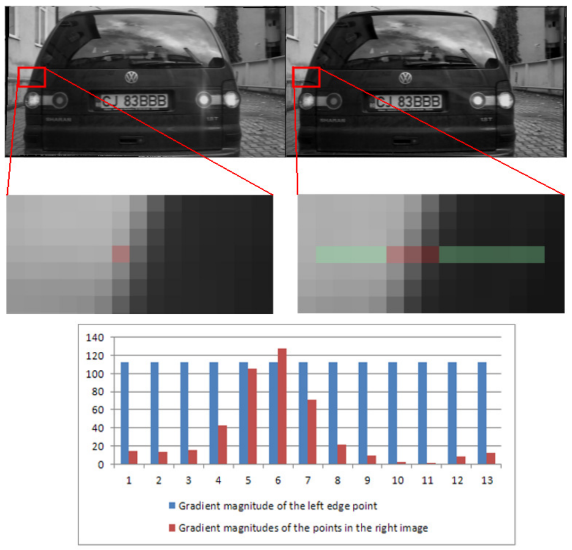

Figure 2. Gradient-based constraint for search space reduction. The red point in the middle-left picture is the detected edge point. In the middle-right image, the green points represent the search space for the correspondent, and the red points are those pixels that pass the gradient magnitude constraint. The diagram depicts a comparison between the gradient magnitude of the left edge point and the gradient magnitudes of all the points in the disparity search range in the right image. Only pixels 5, 6 and 7 have a gradient greater than the threshold.

C. *Sub-pixel accuracy*

Because the matching point in the right image might not always be located on an integer pixel, but rather between two pixels, we perform a sub-pixel interpolation to achieve a better accuracy. We chose to implement the parabola interpolation, in which a parabola is fitted on the values of the correlation cost function corresponding to the best-matching pixel and its left and right neighbors. The displacement d_s from the integer coordinate of the best match is computed as:

$$d_s = \frac{SAD_{left} - SAD_{right}}{2(SAD_{left} + SAD_{right} - 2SAD_{\min})} \tag{3}$$

Another interpolation function that can be used is the symmetric "V", presented in [8]. Our experiments have shown that the two interpolation methods provide almost similar results on the short-medium depth range perceived by the system.

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, 1 GHz. The baseline of the stereo camera system is 4.5 cm. The operating system of the tablet is Android 3.1 Honeycomb. For performance reasons, the entire stereo-reconstruction algorithm was implemented in native code using Android NDK.

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. First of all, we checked if the 3D points are re-projected back on the edge points that were used in their reconstruction. The re-projection result in the left image can be seen in Fig. 3.

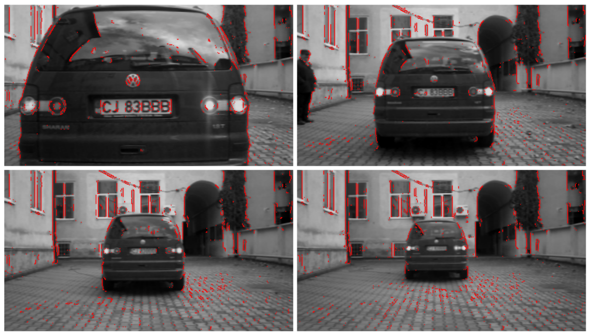
Figure 3. The re-projection result of the 3D points in the left image of the stereo-pair: top-left – the car is at 2 m; top-right – the car is at 4 m;

Furthermore, in order to test the depth estimation accuracy, we selected only the points that lay on the car and computed their mean and standard deviation on the z-axis. Some obtained results can be seen in Table I. We observe that up to a distance of 6 m the mean distance is accurately determined, at 7 m we have an error of only 30 cm, at 8 m the error is 50 cm and at 9 m the error increases to 1.3 m. The reconstruction results and the images used in our experiments can also be seen in Fig. 5.

In Table I the standard deviation of the points reconstructed in our experiments is also represented, which describes how far away from the mean they are scattered. We can see that as the distance grows, the points are more spread out on the z-axis. This phenomenon is more visible in Fig. 5, where at 2 m the reconstructed points are very close to the mean, while at 8 m the points are in the range [6.5 m, 9 m]. However, when the distance gets larger, the standard deviation becomes less reliable due to the decreasing number of edge points, as seen in Table I for distances greater than 7 m.

TABLE I. ACCURACY OF DEPTH ESTIMATION WHEN USING IMAGES OF SIZE 384X216

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

There are more causes for these deviations from the real distance. First of all, the small resolution of the images does not allow an accurate reconstruction at far distances. The size of the images is limited by the device and operating system.

Even though the stereo imager of the device has 5 MPixel cameras, so far the Camera API provides only a small set of supported sizes for the acquired images. The sizes supported by the camera are the following: 720x408, 720x480, 720x576 and 768x432. We chose the last one, as it has the largest horizontal size, relevant for stereo-matching. However, in the dual-camera mode, the two left and right images are scaled down horizontally so that both of them fit in the same picture of size 768x432. As a consequence, the images appear elongated on the y-axis and a further vertical resizing is necessary to bring them back to the natural aspect ratio. Thus, an image of size 768x216 is obtained, which contains both left and right images placed side by side, as in Fig. 4. Individually, the left and right images have a resolution of 384x216 pixels (the equivalent focal length of 404 pixels).

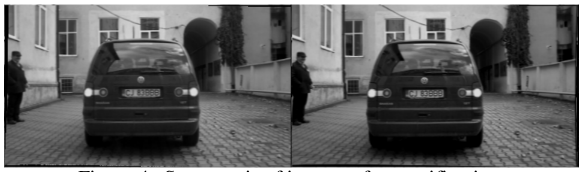
Figure 4. Stereo-pair of images after rectification

For comparison purposes, we captured a set of larger images using a different function of the Camera API (the *takePicture* method), which allows capturing pictures having a combined size of 1200x680. However, for now, this function does not allow sequence mode acquisition. We applied the same reconstruction algorithm on these larger images, which, after vertical resizing, have a size of 1200x340 pixels (each image from the pair has a size of 600x340 pixels). As expected, the reconstruction results are much better than in the case of smaller resolution images. These results can be seen in Table II and Fig. 6. The first improvement worth noticing is that the depth can be estimated with small errors up to greater distances. But more importantly, the reconstructed points are not so spread out around the mean, as it can be observed from the standard deviation. For example, when the object is at 6 m, the standard deviation is only 12.2 cm compared to a deviation of 82.8 cm in the case of smaller resolution images. Moreover, the standard deviation when the object is at 10 m is similar to the one obtained when the object is at 5 m, with the small image configuration.

TABLE II. ACCURACY OF DEPTH ESTIMATION WHEN USING IMAGES OF SIZE 600X340

Measured distance (m)	Mean (mm)	Standard deviation (mm)	Number of edge points
4	4053	78	2645
6	5818	122	1197
8	7351	198	765
10	9493	465	801
15	15102	1009	337

Another cause for the reconstruction errors is the blurring introduced by image rectification, because the intensity value of the new pixel is computed by interpolating the values of the 4 neighboring pixels in the original image. This blurring leads to inaccuracies both in edge detection and in sub-pixel interpolation.

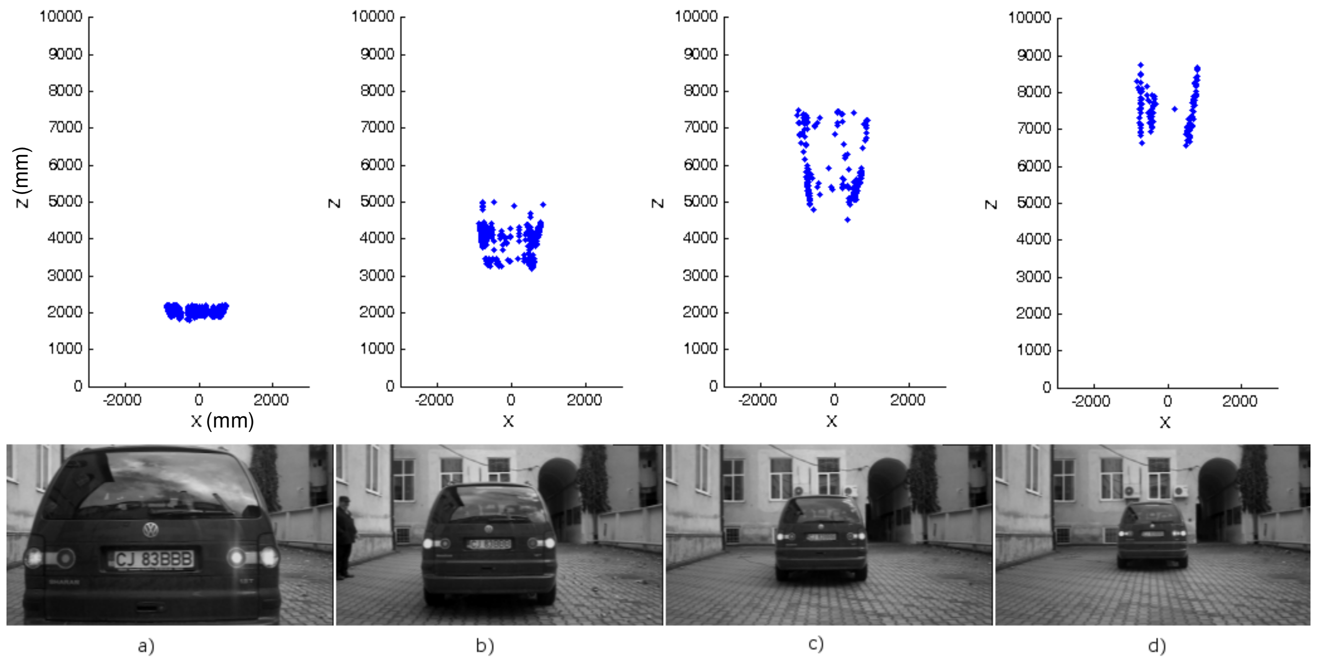

Figure 5. Depth estimation results (top) and the corresponding 384x216-sized left image from the stereo-pair used for reconstruction (bottom): a) the car is at 2 m; b) the car is at 4 m; c) the car is at 6 m; d) the car is at 8 m.

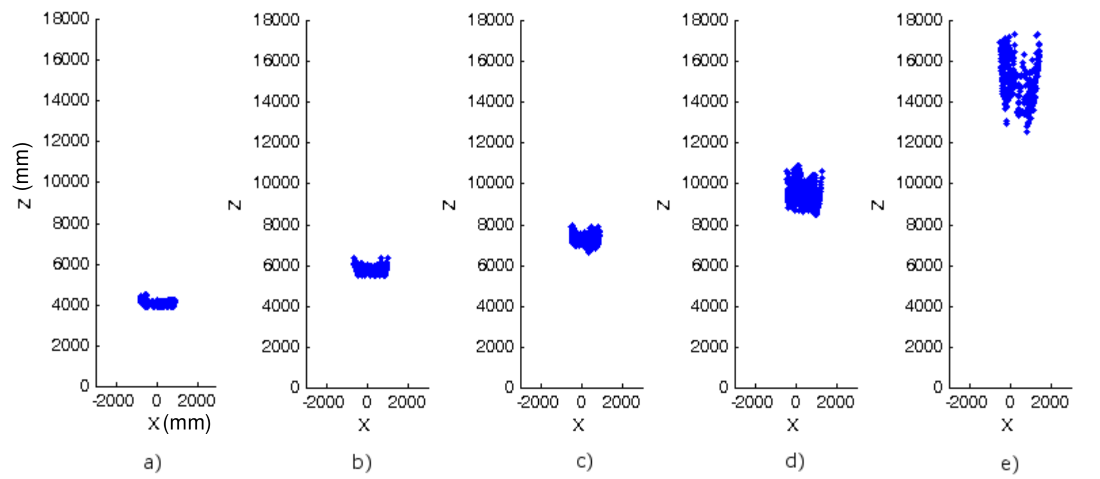

Figure 6. Depth estimation results when using 600x340-sized images and the car is at: a) 4 m; b) 6 m; c) 8 m; d) 10 m; e) 15 m;

Furthermore, sub-pixel interpolation performed by fitting a parabola to a 3-valued neighborhood around the global minimum has a limited accuracy of 1/4 to 1/6 pixels [2]. Due to the small resolution of the images, a small sub-pixel error leads to greater errors in 3D.

Last but not least, false matches could be another cause for reconstruction errors. However, our experiments have shown that pixels are generally well matched, as seen in Fig. 7. This figure depicts the matching result when the car is at 3 m, but this correct correlation behavior is preserved among all test images.

The processing time depends on the number of edge points, as it can be seen in Fig. 8. The analysis was performed on a number of 445 different image pairs, each image with 384x216 pixels and a window of 7x7 for the stereo-matching function. On a scene with lots of edges (9850 edge points, about 12% of the whole image) the whole processing time is about 130 ms (out of which the stereo matching takes about 81 ms). For scenes having the same number of points, the processing time has certain fluctuations, due to the operating system policy for allocating the processor for various tasks. However, these fluctuations can be minimized by proper management of the process priority.

Concerning the speed performance of the whole system, including the image acquisition and visualization of the results, we managed to achieve an average frame rate of 6.5-7.5 fps.

Currently the implementation of the stereo algorithm is single-threaded. Most smart mobile devices feature multi-core processors, so, as a future improvement, we intend to split the processing on two threads in order to increase the speed performance.

Furthermore, when the camera API will allow acquiring larger stereo images in sequence mode, multi-resolution stereo matching can be used to keep the processing time low, while benefiting from the better accuracy of larger images.

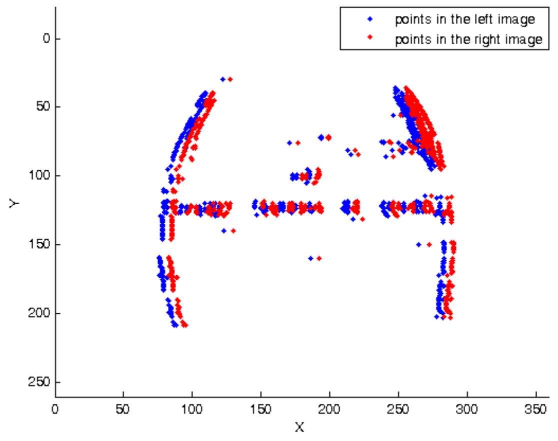

Figure 7. Stereo-matching result (the image space) when the car is at a distance of 3 m

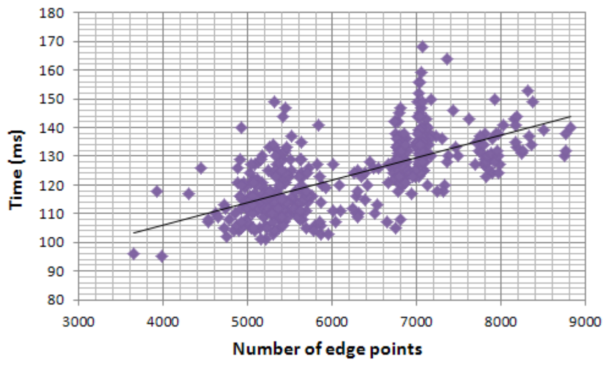

Figure 8. Analysis of the processing time vs. number of edge points

V. Discussion

Several conclusions can be drawn from the conducted experiments and evaluation.

The current sequence aqusition mode supported by the camera API (384x216 stereo images) limits the maximum depth reliable for obstacle detection at about 7-8 meters. Without the sequence mode, when using 600x340 stereo images, the reliable depth is up to 15 meters. By reliable depth we are refering to the maximum depth where, through feature grouping techniques applied on the 3D points and/or the intensity image, the obstacle can be located as a cuboid.

The question still remains: Will this limited range/processing time allow the development of driving assistance functions? This obviously depends on the targeted scenarios, and the main issue is related to the stop distance of the ego vehicle. With the current maximum range for online processing, the system can perceive obstacles up to 7-8 meters. The stopping distance is given by the reaction time of the driver (usually around 1 second for undistracted drivers) and the braking distance. The braking and stopping distances, assuming a dry flat road, and average tires (friction coefficient 0.7), are shown in Table III for speeds typical to urban environments.

In low urban speeds (5-15 km/h), warning the driver can help avoiding accidents involving pedestrians or vehicles. Such low speed scenarios are quite often in modern crowded urban areas, where a lot of stop-and-go maneuvers are performed, or the average speed is low on some road sectors due to the high traffic volume. Even at medium urban speeds (20-30 km/h), automatic warnings might help the driver to mitigate potential accidents. As the reliable depth will likely increase with the future software API, so will the range of speeds where the system is beneficial.

TABLE III. Braking and stopping distance for typical urban speeds

Speed km/h	5	10	15	20	25	30
Speed m/s	1.3	2.7	4.1	5.5	6.9	8.3
Brake Dist. m	0.14	0.5	1.2	2.2	3.5	5.0
Stop Dist. m	1.5	3.3	5.4	7.8	10.4	13.3

Future developments of the stereovision system based on the mobile device include the development of obstacle detection and tracking, finding an easy way to mount the mobile device on the dashboard and to autocalibrate the pitch and roll angles (the device accelerometer will help), and further investigation of the software API for aquiring higher resolution stereo images in continuous mode.

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 Far Distance 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.