PAPER • **OPEN ACCESS**

Object recognition technology for autonomous unmanned control based on stereo data analysis and voxel representation of visible objects

To cite this article: L A Kotyuzanskiy *et al* 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **709** 033069

IOP Conf. Series: Materials Science and Engineering **709** (2020) 033069 doi:10.1088/1757-899X/709/3/033069

Object recognition technology for autonomous unmanned control based on stereo data analysis and voxel representation of visible objects

L A Kotyuzanskiy[1, *], **N V Chetverkin**[2] **and N G Ryzhkova**[3]

[1] Institute of Fundamental Education, Ural Federal University, 620002 Yekaterinburg, Russian Federation
[2] LLC "Nexus", 623780 Artemovsky, Russian Federation
[3] Ural Federal University, Institute of New Materials and technologies, 620002 Yekaterinburg, Russian Federation

*nexus077@gmail.com

Abstract. The paper represents a description of the object recognition technology on a road using moving stereo pair of cameras in real time. The developed algorithm is characterized by high processing speed and performs reliable detection of arbitrary shape obstacles. Calculation time for one iteration from receiving input images (size 1280 * 480) to detecting obstacles is less than 2 msec for a Geforce GTX 1080T video card.

1. Introduction

Object recognition is an important step in solving problems of identifying obstacles by unmanned vehicle control and movement of robots, in object detection at production processes automation.

One of the main parts of the object recognition system is stereo vision, which reconstructs a three-dimensional scene of the visible area with real metric values by a set of images. The restoration of a three-dimensional image is used to solve a plenty of applied problems in various activities [1, 2]. In the motor-car industry stereo vision is used to recognize obstacles and analyze the quality of paving. Approbation of the unmanned vehicle projects [3] determined limitations of object recognition algorithms such as movement in urban conditions, obligatory presence of road markings, recognition of a limited number of object classes require revision of the proposed solutions. A separate direction is the detection of moving objects, which accuracy is based on the analysis of programs working with open initial code does not exceed 60% [4].

Using stereo vision allows to solve two important tasks for unmanned control – object detection and obtaining an estimate of the distance to them, as well as determining their size.

A detailed review of the stereo vision algorithms based on the analysis of two images is represented in paper [5]. The resource capacity of the algorithms does not allow them to be used to calculate stereo data in real time. The method proposed in paper [6] is based on the control points searching, which reduces the amount of data processed. However, the described algorithm implementation by CPU does not allow it to be used to detect objects in real time when using input images with high resolution.

2. Goals and objectives of the work

The main requirements for a real time implementation of the stereo-vision algorithm are its speed and reliability of the received stereo data.

The main goals and objectives of the work are:

1) to obtain reliable object detection on the road of arbitrary shape sized to 10-15 cm; the system should work stably on the road with damage and heavy pollution;

2) to get a high-speed system, allowing it to be used to analyze the video in real time.

The basis of the proposed technology is a passive stereo vision system which consists of two cameras. It requires qualitative automatic or semi-automatic calibration of the stereo camera, as well as synchronized frame getting. Variants of active stereo vision based on structured backlighting [7, 8, 9] have limitations associated with road illumination and maximum range.

The output of the system are the voxel representation of objects for detecting obstacles, the "parktronic" lattice and the depth map.

1) The voxel world of obstacles in the cloud of points is a cubic approximation (or a model of the world visible by cameras), where each active cube is an obstacle. For each voxel coordinates are determined relative to the starting point - the stereo camera. The voxel world has borders and is situated inside a truncated pyramid. Closer to the front cut-off zone, the stereo pair does not define obstacles, the far edge is the maximum allowable visual range.

2) "Parktronic" lattice is the approximation of voxels in 1D representation (ruler) to simplify the control model of an unmanned vehicle.

3) Depth map for calculating the distance value at an arbitrary point of the frame from the stereo camera.

3. Object recognition technology for autonomous unmanned control based on the analysis of stereo data and voxel representation of visible objects

We shall consider the general processing algorithm carried out in 1 iteration.

3.1. Image acquisition

At the first stage, the transfer of images from cameras and their pre-processing takes place.

1) We get the orientation of the car – with the camera – relative to the road based on the gyroscope data (Fig. 1).

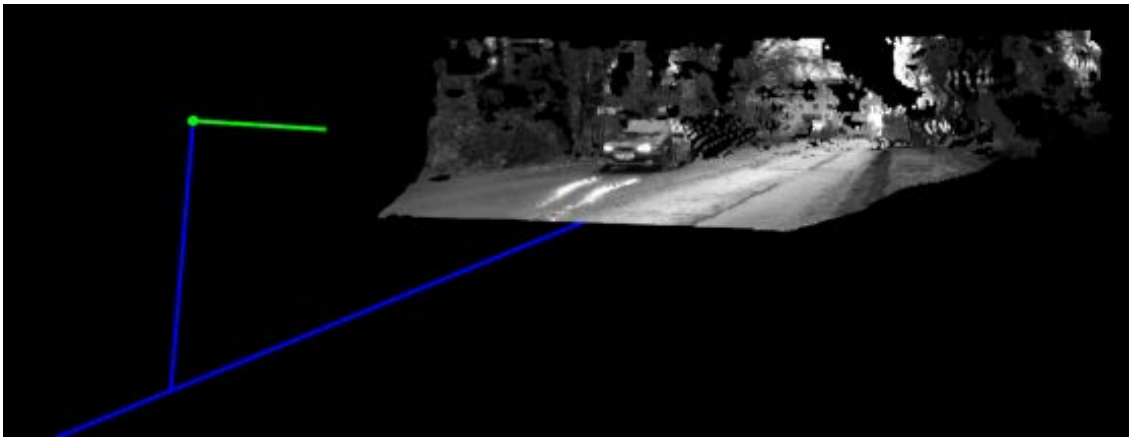


Figure 1. The orientation of the vehicle on the road. The orientation to the road is shown by a blue line parallel to the "road", the green line shows the angle of the camera to the road.

2) There goes a request for simultaneous acquisition of images from two cameras.

3) The resulting images are rectified to eliminate optical distortion according to the camera calibration and copied to the GPU.

3.2. Determination of support points

The next step is to determine the coordinates and values of the disparity of the strong points to increase the reliability and reduce the algorithm working time. It is important for identifying the obstacles on the road correctly. The algorithm was proposed in [6], but in the original paper it is performed by CPU and is resource-capacious for real-time application. Work on its implementation on OpenCL and obtaining acceleration of the order of 100 - 150 times (Gtx 780ti) as compared with the work on the CPU (Core i7 860) in one stream was carried out. Testing based on the rectified stereo images [10] showed high efficiency of the method.

A disjoint lattice with S-step is set to correspond to the input image, but describes it rarefied. By default, each node is set to -1 (no disparity is defined). The method is based on a sliding window approach for searching for pixels with the greatest similarity in the left and right images.

The GPU / OpenCL strong points searching algorithm includes 3 stages of OpenCL core processing.

Stage 1. Calculating descriptors $Desc_l$, $Desc_r$. Two black-and-white rectified images I_l, I_r with the dimension $W^I \times H^I$ are received at the input, $p^l_{x,y}$ – is the element – pixel of the left image, $p^r_{x,y}$ – is the element – pixel of the right image.

The Sobel operator is applied to the images using 3×3 cores. The results of convolution in the vertical and horizontal directions are arrays G^x and G^y. Convolution is calculated by GPU, OpenCL core is configured as a two-dimensional matrix of parallel elements, where each parallel element calculates its gradient element $g^x_{i,j}$ from the set G^x and $g^y_{i,j}$ from the set G^y.

There is a combination of the values of G^x and G^y in the image descriptor $Desc$, with a resolution of $2\,W^I \times 2\,H^I$, where each element of the image is a 4-component vector. OpenCL core runs as a two-dimensional matrix of dimension $W^I \times H^I$ of parallel elements, each of which performs 16 readings of values from single-channel images G^x and G^y and writes these values into 4 pixels of the output image $Desc$.

Stage 2. Calculating the values of the nodes of the disjoint lattice of the left image relative to the right using descriptors $Desc_l$, $Desc_r$ of dimension $4 \cdot (2\,W^I \times 2\,H^I)$. The output data is $DisparityGrid_{LR}$ of the dimension $\frac{W^I}{S} \cdot \frac{H^I}{S}$, where S – is the thinning step, which determines the number of nodes of the dispersion lattice and the pixel distance between the lattice nodes relative to the original input image. To do this, OpenCL core runs a matrix of dimension $lSizeX\,\frac{W^I}{S} \times lSizeY\,\frac{H^I}{S}$, where $lSizeX$ and $lSizeY$ – are the size of the workgroup of parallel elements for which local memory is available. or our task, both parameters are 16, so the size of the working group is 256 parallel elements that have a united local memory. The implementation of the algorithm by OpenCL was built in such a way that the size of the working group determined the maximum possible value of the disparity that can be determined for the lattice site. Consequently, each element of the working group is assigned its own value of disparity, for which it calculates the "energy" or the price of the conformity of "its". This happens in parallel for each node of the single grid. After this, the reduction is performed with searching and saving of 1 minimum and 2 minimum values of the energy of the disparity. In this case, the 1D coordinates of a parallel element in the coordinate system of the working group [0, 255], for which the minimum energy value of 256 elements was found, will be the actual value of the dispersed lattice node.

Stage 3. Calculation of the points of the right image relative to the left $DisparityGrid_{RL}$ and check of the correspondence of the disparities of the found points on the left and right image. The points are also checked for the presence of neighbouring points/nodes of the lattice that have similar values. Each parallel element of OpenCL core is responsible for analysing the two-dimensional neighbourhood of the corresponding node for the presence of nodes with similar disparity values, if the number of such nodes is less than the threshold value, then the node in question is punctured (the value is -1).

The found disjoint lattice is sparse, contains only those points that have undergone a multi-step test and contain reliable stereo data (Fig. 2).

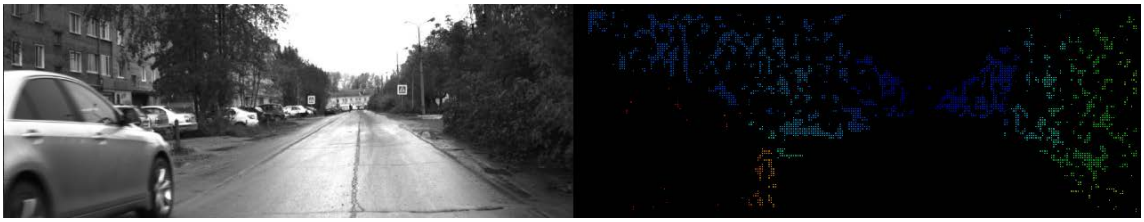


Figure 2. Construction disjoint lattice.

For objects searching, it is necessary to increase the density, and actually turn the rarefied lattice into a depth map, where a value is determined for each point of the map. To construct a depth map from the set of support points in [6], the Delaunay triangulation is used. However, this approach to the problem being solved has several drawbacks. With inaccurate / erroneous strong points found, triangulation can strongly distort the map and create a significant number of false obstacles. In addition, the Delaunay triangulation is resource-intensive in the context of a real-time task and is rather difficult to parallelize.

To remove the restrictions, an algorithm was developed that interpolates the sparse lattice to turn it into a disparate map, as well as the generation of a three-dimensional point cloud from the depth map.

3.3. Depth map computation and point cloud generation

Based on the matrix of disparity of the visible scene $DisparityGrid_{LR}$, where $gridW = \frac{W^I}{S}$ – is the number of lattice nodes horizontally, $gridW = \frac{H^I}{S}$ – is the number of lattice nodes vertically, $DepthMap_{LR}$ of the same dimension containing interpolated values is calculated disparate values.

Stage 1. Horizontal interpolation. To interpolate the disparity value, a separate parallel element of the OpenCL core is used.

Linear interpolation is performed for a pixel of the depth map for which there is no disjoint lattice node, or the value of the disparity lattice node is not defined. At a single moment in time t, each parallel element in the pos coordinate calculates a value using Eq. 1:

$$Dout_{pos} = DepthMap_{LR}[pos], \text{ если } D_{pos} = DisparityGrid_{LR}[pos] = -1. \quad (1)$$

A horizontal study of the pos element in the r- neighborhood is performed to search for the nearest nodes on the left (coordinate lp) and on the right (coordinate rp) with disparity values greater than -1, respectively D_{lp}, D_{rp}. If such points are not found, then $Dout_{pos} = -1$. Otherwise, the points RW_{lp} and RW_{rp} are calculated in the three-dimensional coordinate system, where the origin of the coordinate system is the stereo pair sensor.

In fact, 3D reprojection of the point is performed using well-known Eq. 2:

$$RW_{lp} = [X\ Y\ Z\ W]^T = Q \cdot \left[x_{lp}\ y_{lp}\ D_{lp}\ 1\right]^T, \quad (2)$$

where $Q_{4\times4}$ – perspective transformation matrix, x_{lp}, y_{lp} – coordinates of the node D_{lp}.

Since the dispersed array was thinned out, a factor S is required for correct conversion in order to return to the coordinate system of the frame received from the camera. The calculation of RW_{rp} is completely analogous.

Based on 2 points in the coordinate system of the visible world, the value of $Dout_{pos}$ is calculated using Eq. 3:

$$Dout_{pos} = \begin{cases} D_{lp} + \left(\frac{D_{rp}-D_{lp}}{lx_{rp}+lx_{lp}}\right) \cdot lx_{lp}, if\ \left|RW_{rp}(z) - RW_{lp}(z)\right| < \max Z, \left|RW_{rp}(x) - RW_{lp}(x)\right| < \max XY, \\ -1, otherwise. \end{cases} \quad (3)$$

Here $lx_{lp} = |x_{lp} - x_{pos}|$, $lx_{rp} = |x_{rp} - x_{pos}|$, $\max Z$, $\max XY$ – are the threshold values of the differences in the corresponding directions.

Stage 2. Similarly, interpolation is performed vertically. Input data consist of the values are interpolated horizontally.

Stage 3. Generation of a cloud of 3D points on a smoothed depth map. Each value of the depth map at DM_{pos} corresponds to $CloudP3D_{pos1D}$ using Eq. 4:

$$pos_{1D} = x_{pos} \cdot gridH + y_{pos}. \quad (4)$$

A two-dimensional scan is performed of the area centered at pos and the size of a square neighborhood of r according to Eq. 5:

$$\forall i,j: x_{pos} - r \le i < x_{pos} + r, y_{pos} - r \le j < y_{pos} + r, \text{если } DM_{pos} > 0:$$
$$numP = numP + 1, mlt = mlt + (1 - p), lSum = lSum + DM_{(i,j)}. \quad (5)$$

Here $numP$ – number of points selected, mlt, p – smoothing ratio, $lSum$ – local sum.

The smoothed value of the depth map at pos is calculated by Eq. 6:

$$SmoothDM_{pos} = \frac{mlt \cdot DM_{pos} + lSum}{numP}. \quad (6)$$

The algorithm was originally developed by GPU and has a simple but effective interpolation model for scenes with a characteristic horizontal arrangement of objects. The results of the work are presented in Fig. 3.

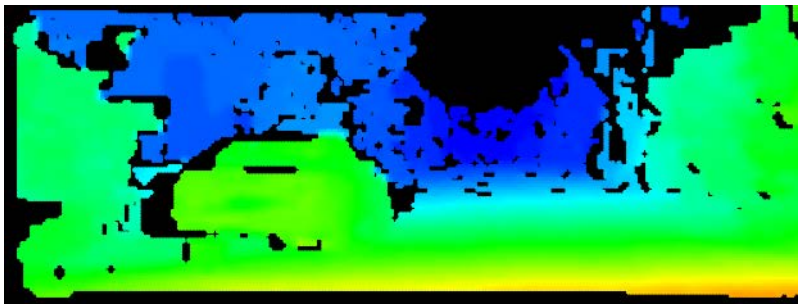


Figure 3. The resulting depth map according to the strong points.

2D blurring smoothes the depth map and point cloud and is an additional way to filter out unwanted noise.

3.4. Evaluation of three-dimensional points of the cloud

Outgoing cloud $outCloud$ – points which the evaluation are performed for.

Let $id0 = x_{pos} \cdot gridH + y_{pos}$ – 1D-number of the point being processed in the cloud of points by a parallel element with the coordinates pos. Then $outP$ – is a cloud point with $id0$ index from the $outCloud$ set, for which a parallel element with pos coordinates must perform its estimation. We form the neighbors of a three-dimensional point relative to the two-dimensional space of the depth map, performing calculations using Eq. 7:

$$id_1 = x_{pos} \cdot gridH + (y_{pos} + 1), id_2 = (x_{pos} + 1) \cdot gridH + y_{pos},$$
$$id_3 = (x_{pos} + 1) \cdot gridH + (y_{pos} + 1). \quad (7)$$

Having 4 nearest neighbouring points, you can calculate a new point of the cloud, determine in it the normal to the plane passing through 3 selected neighbouring points, and evaluate it relative to the three-dimensional $groundPlane$ vector describing the plane which the vehicle is currently located on.

Based on the nearest neighbours, an estimate is made in the range [0,1], where 0 is free space, 1 is an obstacle point. Since the orientation of the car to the road is known, each point of the cloud can be estimated by calculating the angle between the normal of the point in question and the vector of the orientation of the car. The score is 0 - the vectors are parallel, and 1 - are perpendicular.

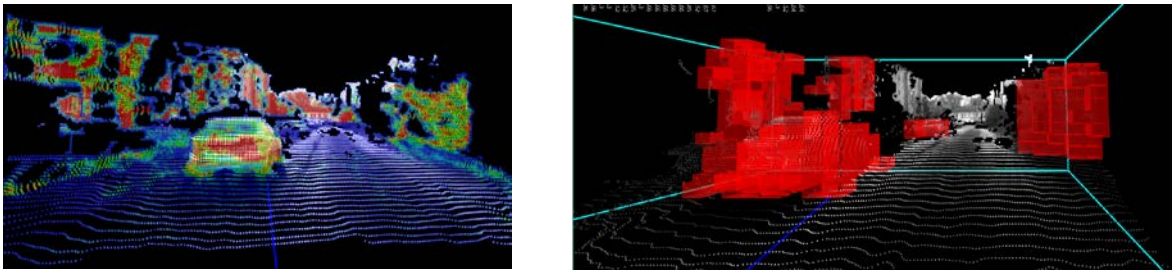
Similarly, the procedure is carried out for the calculation of distant neighbours, the output is $farCost$ estimate. Then the final price value of the output point $outP$ is calculated by Eq. 8:

$$Cost(outP) = (1 - NormalMix) \cdot Cost + farCost \cdot NormalMix, \quad (8)$$

where $NormalMix$ – the coefficient of mixing price of distant and near neighbors. This is a balance between the ability to find small obstacles and the resistance to noise from stereo data.

The final stage is to drop the value of the point price, if the distance of this point from the $groundPlane$ is higher than the threshold in question. The calculation of the distance of the $outP$ point from the ground plane of the $groundPlane$, given by its normal and the plane point, is performed.

For convenience of perception, each point in the processed image (Fig. 4) is colored in the format of a heat map, where blue corresponds to 0 and red to 1.



(a) (b)

Figure 4. (a) Estimation of point cloud, (b) Obstacle voxels filtered by distance are found.

3.5. Building a voxel model

To determine the obstacles, the activation level of each voxel is calculated. For all points included in the volume of this voxel, the sum of their estimates is the level of activation of this voxel. A voxel is considered enabled (active) when it or its neighbours were observable for several iterations in time, while the voxel activation level was higher than the established constant, and its z value also did not exceed the distance from the camera. Fig. 6 (b) shows an example of the voxels found filtered by distance.

Table 1 shows the performance comparison between CPU and GPU implementations. The calculation time of one iteration is considered. The CPU version used the original ELAS version using triangulation and SSE instructions. Stereo resolution: 1280 * 480 pixels.

Table 1. Comparison of the performance of the algorithm with the implementation on different hardware.

Work in progress	Xeon E5 1660 3.9 ггц (1 stream)	Geforce GTX 780TI	Geforce GTX 1080TI
Full signal processing cycle - from camera input images to active voxels	220 ms	5 ms	1.84 ms
Depth map calculation	110 ms	3.1 ms	1.14 ms
Cloud generation and pricing	65 ms	1.1 ms	0.412 ms
Voxel activity calculation	45 ms	0.8 ms	0.29 ms

4. Conclusions of the research and prospects for further development

The proposed system has a high detection rate, performs reliable detection of obstacles of arbitrary shape not only on asphalt but also, in the conditions of the snow-covered, polluted, dirt road. Managed to reach detecting obstacles of small size - about 8-10 cm, at the distance of up to 7-10 m (Fig. 5).

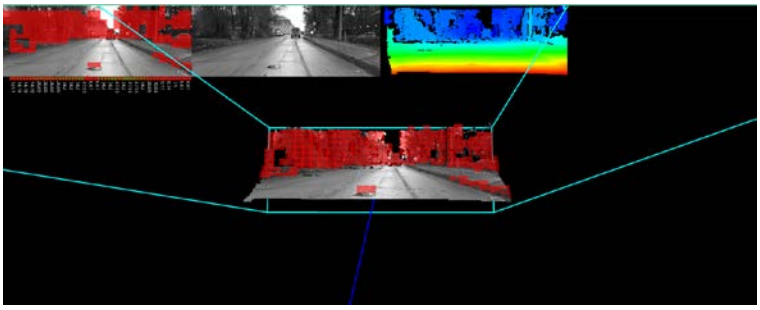


Figure 5. Border detection (right) is not more than 10 cm high and small debris on the road.

One iteration calculating time from receiving input images (size 1280*480) to the formation of obstacles with all related calculations (depth map, point cloud, finding obstacles) is 4 msec for Geforce GTX 780TI video cards and 1.4 msec for the Geforce GTX 1080TI video card, which corresponds to about 250 - 700 frames per second. This speed of calculation allows to adapt this system for high-speed low-flying unmanned vehicles to fly around obstacles.

However, the described system cannot detect minor road defects, car ruts and extensive puddles on the road can also be a problem.

To overcome the described limitations we could combine the system with a convolutional neural network.

References

[1] Kotyuzhansky L A 2014 *Contactless Interface for Human-machine Interaction Based On Sensor-range Finder* (Autoabstract of the Dissertation, Ekaterinburg: UrFU)

[2] Yurin D V 2009 Modern concepts of three-dimensional scenes restoration on a set of digital images: filling of virtual reality systems *Proc. of the First Int. Conf. "Three-Dimensional Visualization of Scientific, Technical and SocialRreality. Clustering Technology Modeling"* vol 1 (Izhevsk, Udmurt State University) pp 96-100

[3] Rusanov A D and Nekrasov D K 2016 Overview of the principles and algorithms of environmental objects recognition in unmanned vehicles *New Information Technologies in Automated Systems* № 19

[4] Shepelev K V 2017 Detection and classification of moving objects in video sequence *Technical Sciences. Computer Science and Engineering* № 4 (44)

[5] Vakhitov A T, Gurevich L S and Pavlenko D V 2008 Review of the stereovision algorithms *Stochastic Optimization in Computer Science* (SPb.: Publishing house of St. Petersburg State University) Issue 4 pp 151 – 169

[6] Geiger A, Roser M and Urtasun R 2010 Efficient large-scale stereo matching *Asian Conference on Computer Vision* (ACCV)

[7] Jang W, Je C, Seo Y and Wook Lee S 2013 Structured-light stereo: Comparative analysis and integration of structured-light and active stereo for measuring dynamic shape *Optics and Lasers in Engineering* vol. 51 Issue 11 pp 1255-1264

[8] Fanello S R, Rhemann C, Tankovich V, Kowdle A, Escolano S O, Kim D and Izadi S 2016 HyperDepth: learning depth from structured lightwithout matching *CVPR*

[9] Scharstein D, Szeliski R 2003 High-accuracy stereo depth maps using structured light *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (CVPR Madison, WI), vol. 1 pp 195–202

[10] Unified system for testing stereo algorithms, URL: http://vision.middlebury.edu/stereo/