2017

System for creation and display of 3D maps of coal mines

Tomas Kot
University of Ostrava

Petr Novak
University of Ostrava, petr.novak@vsb.cz

Jan Babjak

Publication Details

Tomas Kot, Petr Novak and Jan Babjak, System for creation and display of 3D maps of coal mines, in Naj Aziz and Bob Kininmonth (eds.), Proceedings of the 17th Coal Operators' Conference, Mining Engineering, University of Wollongong, 8-10 February 2017, 117-124.

SYSTEM FOR CREATION AND DISPLAY OF 3D MAPS OF COAL MINES

Tomáš Kot[1], Petr Novák and Jan Babjak

ABSTRACT: This paper presents a system for creating, processing and visualisation of 3D maps of underground coal mines (old or new mine shafts with no other types of maps available, corridors affected by an accident). The maps in the form of point clouds are created by 3D laser scanning. The paper mentions the most important algorithms applied in point cloud data pre-processing, especially voxelization, outlier removing and smoothing. In more detail is described the rendering engine with its advanced visualization methods like shading and colouring and special distance measuring and shape highlighting features designed to help the user to orient in the virtual view of the mine.

INTRODUCTION

Accidents in underground coal mines are often very grave and lethal. Especially serious are underground explosions caused by dangerous concentrations of coal dust or methane. Typically, the area of the mine affected by an explosion is sealed by a thick stopping with a service hole and human rescuers are not allowed to enter the zone until the monitored parameters drop below critical limits. It would be extremely beneficial, if the rescuing operations could start as soon as possible, because every wasted human life is a huge tragedy.

A logical solution is to use mobile robots, which could be applied much earlier. Mobile robots can perform reconnaissance of the tunnels to measure concentrations of dangerous gasses and temperatures and provide information about the physical state of the tunnels. The research in this area has been already running for many years (for example Ray, *et al*, 2015; Gomathi, *et al*, 2015).

A new research project “System for virtual TELEportation of RESCUER for inspecting coal mine areas affected by catastrophic events (TeleRescuer)” has started in 2014 in the framework of an EU programme Coal and Steel under the grant agreement RFCR-CT-2014-00002. This project is solved by multiple teams from the Czech Republic, Poland and Spain and the goal of the project is to develop a mobile robot possessing all useful tools and features for this task in one body and – which is unique – to get the important certifications for deployment in coal mines without risks of causing additional damage (eg, explosion safety), as described by Novak, *et al* (2015a); Novak, *et al* (2015b); Moczulski, *et al* (2014).

PREPARATION OF A POINT CLOUD BY SCANNING

The mobile robot TeleRescuer is equipped with a huge variety of sensors, including a 3D scanning device consisting of the Sick LMS111 2D laser range finder mounted on a turning platform providing the third dimension (Figure 1). To make one 3D scan, the scanning device performs a series of 2D scans with increments of the turning angle. The result is presented as a set of points in spherical coordinates. These individual 3D scans can be made in multiple places and then can be merged into one complex map of the mine in the form of a point cloud (for example Blanco, 2013; Rusu, *et al*, 2011; Universität Karlsruhe, 2011).

Distance between the scans is typically a few meters (Olivka, *et al*, 2016). The density affects precision of the resulting map and the amount of details, but also the data size, which can be a limiting factor when the 3D map has to be transferred wirelessly to an operator in real time.

[1] *VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Robotics, Czech Republic. Email: petr.novak@vsb.cz, Mob: +42 0 59 732 3595*

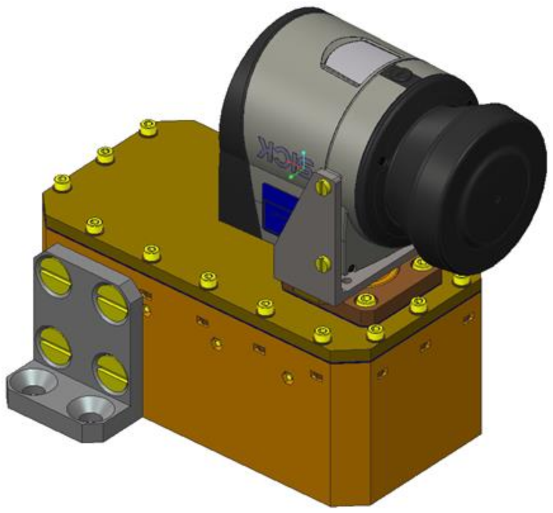

Figure 1: Model of the ED scanning device

INPUT DATA FILTERING

For efficient processing and display of the point cloud without excessive errors it is very important to perform some filtering of the input point cloud. All the following algorithms were implemented using the *Point Cloud Library (PCL)*, which is a very powerful library for point cloud and 3D geometry processing (PCL, 2016b).

Decreasing number of points

The Laser Scanner Sick (LMS) 111 has a working range of 20 m. Because the 3D scanning works with spherical coordinates, density of the measured points lowers with distance from the origin and individual 3D scans must be done much more often than just every 20 m; a good value discovered by testing is between 2 and 5 m. This means that the scans overlap, which is beneficial for increasing details of the resulting point cloud, it also dramatically increases its size and complexity. Multiple points can even occupy almost exactly the same 3D coordinates, which is a waste of data and processing power.

A common way of removing very close points is *voxelization*. The corresponding implementation in the :Point Cloud Library (PCL) is a filter called *Voxel Grid* that works by presenting a 3D box (voxel) grid over the points; then in each box all points are approximated by just one, with location within the box calculated as average of the removed points (PCL, 2016a). This filter in general is used for down sampling, i.e. lowering of density of a point cloud while preserving the overall distribution of points in space. With a small voxel size (approx. 10 mm in our case of mine tunnels), no details are lost and only duplicate points are removed. In testing with scans crated every 2 m and 10 mm voxel size, the point cloud was reduced by 9 % of points.

Removing noise

One of the typical problems of point clouds acquired by 3D scanning is noise caused by measurement errors. Measurement errors can corrupt the point cloud with sparse points lying out of larger clusters of points and not representing real objects. This complicates further processing, especially analyses like estimation of surface normal, and can lead to false values.

PCL contains the filter called *Statistical Outlier Removal* that works by performing statistical analysis on each point’s neighbourhood and trimming those which do not meet certain criteria (PCL, 2016c). The optimal configuration of this filter in testing removed around 0.8 % of points.

Another negative effect of noise in the point cloud data caused by measurement errors is when what should be a smooth surface is actually represented as rough. This can be dealt with by smoothing, for example using the *PCL* algorithm *Moving Least Squares* surface reconstruction method (PCL, 2016d). This resampling algorithm attempts to recreate surfaces by higher order polynomial interpolations between the points. A side effect of this algorithm is the ability to calculate also surface normal vectors, which will later be used for lighting. The important configuration value is the search radius. Optimal value is 100 mm, demonstration of the impact of the value is shown on Figure 2 – too high a value can erase important details.

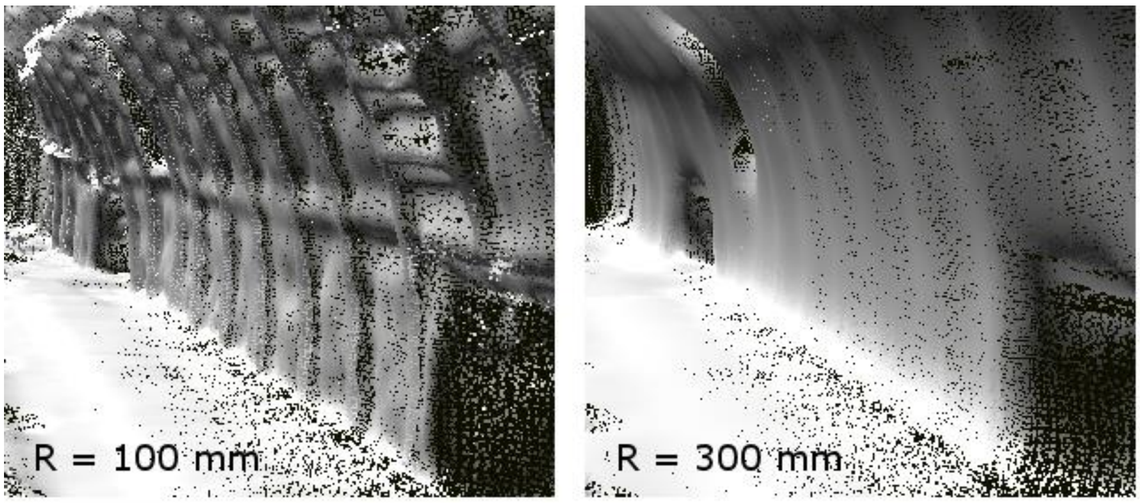

Figure 2: Comparison of two different smoothing settings – radius 100 mm and 300 mm

Graphical engine

The goal of the TeleRescuer project is to provide “virtual teleportation” of a human rescuer into the sealed mine. Thus, the point cloud should be presented to the operator in a clear and illustrative way and allow him to inspect all important parameters in real time while roaming freely through the shafts in a virtual space, or while watching the actual surroundings around the mobile robot for safe driving with remote control. It also should integrate important sensor readings and similar data into the point cloud.

The best solution was to create a custom graphical engine tailored to the needs of this project. The engine is implemented in C++ and uses the Direct3D 9.0c graphical API for hardware graphic acceleration of many complex calculations related to rendering.

Rendering of individual points

Every point in the point cloud stored in a special *PCL* array contains the following parameters:

- Position vector **p** = (p_x, p_y, p_z).
- Normal vector **n** = (n_x, n_y, n_z).
- Colour vector **c** = (r, g, b, a).

The points with these data are expressed as *vertices* and stored in a Direct3D *vertex buffer*, which is then drawn as a Direct3D primitive type *point list*. During the rendering process of Direct3D, every vertex is processed by a *vertex shader* (a programmable stage that can modify parameters of each vertex in many ways, but at least have to transform 3D coordinates to 2D screen coordinates) and by a *pixel shader* (this stage calculates the colour of the resulting pixel).

Performance optimisations

Even hardware-accelerated rendering of the very simple *point list* primitives would not be able to display hundreds of millions of points with an acceptable framerate on a mid-level computer. It is necessary to lower the number of actually processed and drawn points in a way that will not negatively affect the displayed information. The two implemented methods are:

- *View frustum culling* – ignoring points out of the current camera view,
- *Level Of Detail (LOD)* – drawing fewer points in larger distances from the camera.

These algorithms cannot be efficiently calculated for every individual point. The applied solution utilizes the Octree system (Wikipedia, 2016) with one hierarchy level to partition the point cloud into a set of box nodes – Figure 3. View frustum culling and LOD are then applied on the level of nodes rather than points.

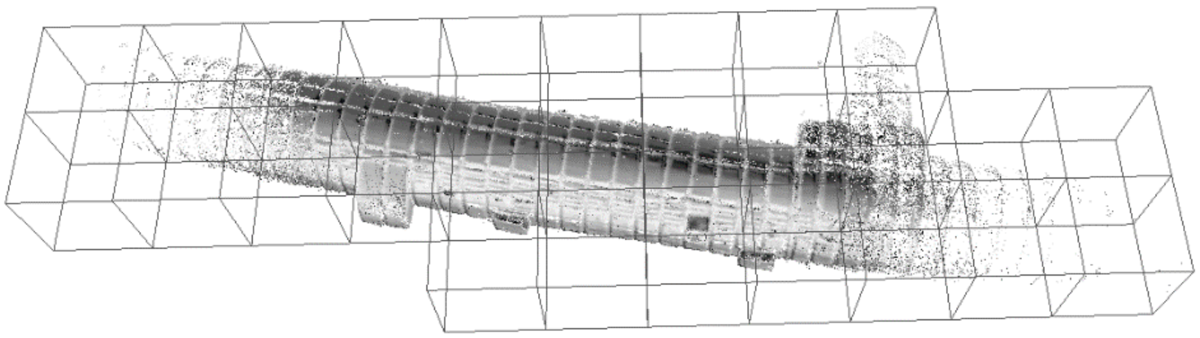

Figure 3: Simplified Octree system with one hierarchy level

Tests demonstrated that view frustum culling removes 70% of points in average, because typical view is from inside the mine shafts looking towards one direction and most points are behind the view. That however still leaves very large number of points, especially when the operator is looking down a very long corridor (in some extreme situations view frustum culling may remove even less than 10 % of points). The LOD algorithm calculates distance of each node from the virtual camera and this value is used to determine how many percent of its points will be drawn. When properly configured, this optimization is not visually detectable at all because far points blend together anyway.

With both the optimizations active, the actual number of rendered points typically never rises above 10 % of existing points, but usually the ratio is even much lower (around 2%).

Projection transformation

For good visual impression it is necessary to draw the point cloud with perspective projection, which is easily achieved in Direct3D by using a 4x4 homogenous perspective projection matrix in the vertex shader stage. By default, every individual point rendered as a Direct3D point lists occupies exactly one pixel on the screen, regardless of its distance from the camera (the projection matrix does not apply here). Distances between points are affected by perspective foreshortening but size of the points is not, which makes large gaps between points close to the camera and the operator loses the impression of being close to a wall – this can be very dangerous for direct control of the robot (Figure 4). The solution of this problem is to use the output parameter PSIZE of vertex shader, which controls the resulting pixel size of the point on screen. This way it is possible to get points drawn bigger than just 1x1 pixels with almost no additional cost. The value of PSIZE can be set individually for every vertex and is calculated in the vertex shader as the reciprocal value of the point’s distance from the camera in 3D space.

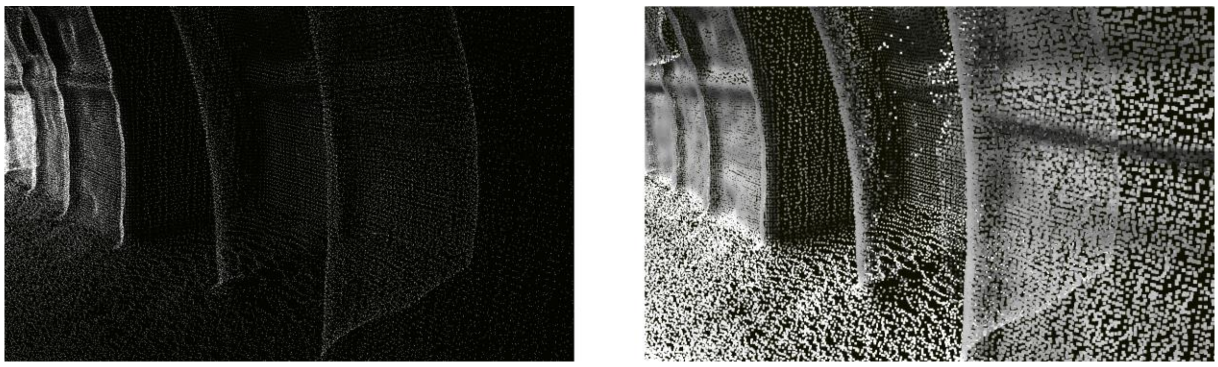

Figure 4: Close view of a wall – rendered with fixed (left) and variable (right) point size

Point colours

The colour used to draw each point on the screen can be used for example to encode additional information from sensors (will be discussed in the following chapters) and another possibility is to use colours for better visual impression, which can help the operator to orientate in the point cloud representation of the real coal mine.

Two main characteristics of a colour – hue and brightness can be used. The resulting pixel colour of each point (Figure 5) in this software is calculated as:

$$\mathbf{c} = (0.7 \cdot \mathbf{c}_h + 0.3 \cdot \mathbf{c}_n) \cdot (a + d + s), \tag{1}$$

where:

- $\mathbf{c}_h$ – colour vector (r_h, g_h, b_h) based on the absolute height of the corresponding point in the 3D space (its p_z coordinate). The colours generated by this algorithm create a linear gradient with key colours (from lowest p_z to highest): green – yellow – purple – blue – azure. The colours were chosen based on natural habit that ground is green (grass) and sky is blue.
- $\mathbf{c}_n$ – colour vector (r_n, g_n, b_n) based on the normal vector of the corresponding point. This colour is a gradient between red, green and blue key colours (red when the normal is aligned with the x axis of the global coordinate system, green for the y axis and blue for the z axis).
- a – ambient lighting scalar factor. Ambient light is constant in the whole scene and brightens up all points for better visibility.
- d – diffuse lighting scalar factor. Diffuse light depends on the angle between the normal vector of the point and the vector of light rays hitting the point.
- s – specular lighting scalar factor. Specular light depends also on the vector from the point to the virtual camera used to watch the 3D scene from and creates bright “reflections” of the light source.

The d and s factors work with geometrical properties of a virtual light source, which is a very simple directional light characterized only by a single vector or light rays (simulation of a light source located infinitely far away).

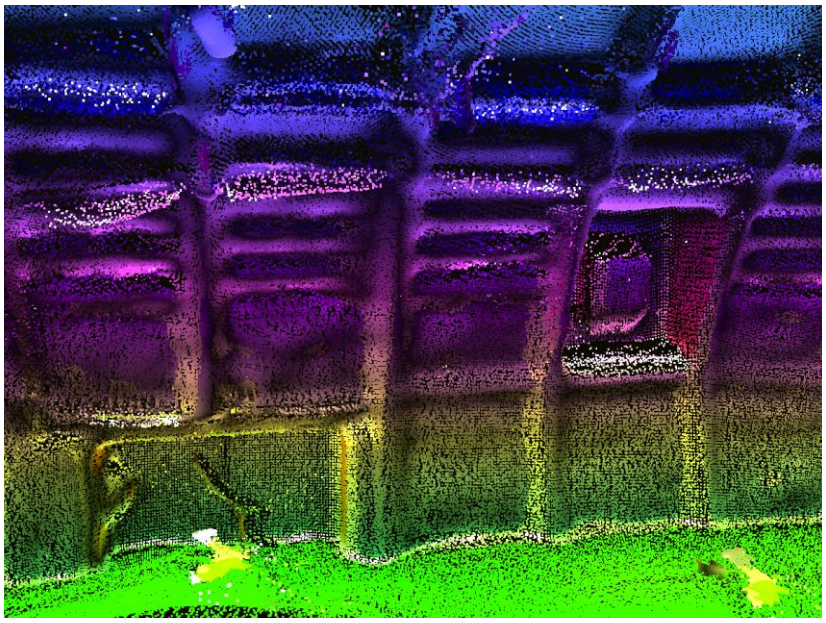

Figure 5: Rendering with point colours calculated by equation (1)

INTEGRATION OF ADDITIONAL INFORMATION TO THE POINT CLOUD

It is useful for the user to be able to get some additional information from the 3D map visualization, especially distance measurements or sensor readings.

Values acquired by sensors on the robot during scanning (such as temperature, wind speed, gas concentration) can be simply encoded by colour gradients and applied to points in the point cloud (Figure 6). When this functionality is activated by the user, the equation (1) is replaced with:

$$\mathbf{c} = \mathbf{c}_s \cdot (a + d + s), \tag{2}$$

where $\mathbf{c}_s$ is colour vector (r_s, g_s, b_s) representing the particular sensor value (using a colour gradient).

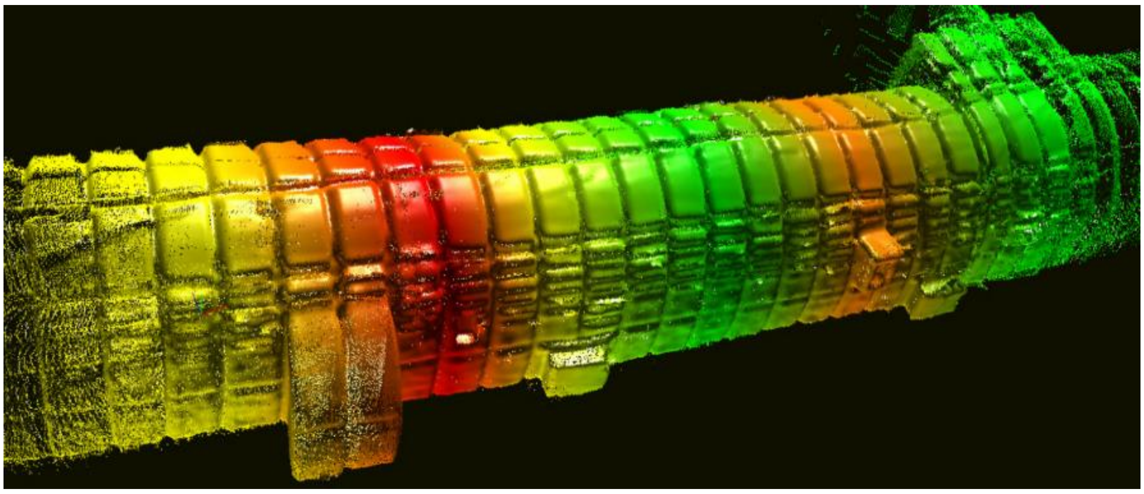

Figure 6: Visualisation of temperatures in a tunnel using a colour gradient

Distance measurements are important to get quick details about for example the height or width of a narrow passage. The simplest distance measuring provided by the application works simply by clicking on any two points directly in the 3D scene – the points are then connected by a line, with distance measurement in meters (or millimeters) printed on the line. To get a better visual clue about the shape of a tunnel, it is possible to display a cross-section in red colour (Figure 7). This can help to discover tunnel branching, a hole or recess in the wall, or an obstacle in the tunnel.

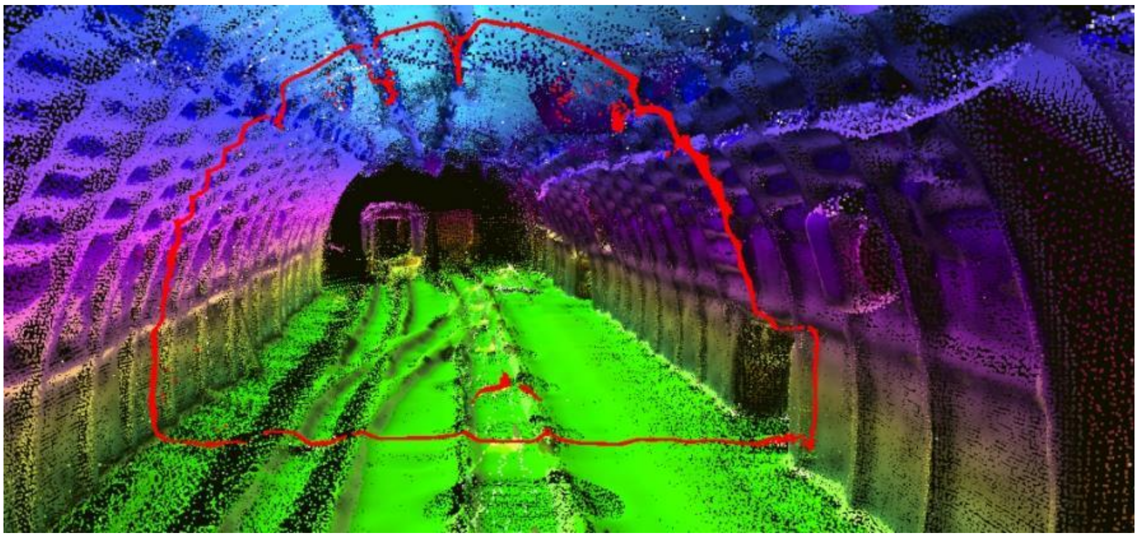
Figure 7: Tunnel cross-section

CONCLUSION

The project is still in progress and the visualization software described in this paper is still in development, but all the mentioned functionality is already implemented and working. The images were prepared with a testing point cloud made from 11 individual 3D scans created in 2 m distances in a real coal mine in Gliwice, Poland. Total number of points after filtering is 2 112 270 and the number of nodes (boxes 2 x 2 x 2 m) is 27 (Figure 7). Rendering runs at more than 250 FPS in 1920x1080 on Intel Core i5-3330 CPU and Nvidia GeForce 750 GTX. This leaves a lot of power reserve for even many-times larger point clouds.

The system is applicable as a general point cloud visualization and renderer, although it was designed primarily for coal mines with their special properties. Unique features of this system are: free walking or orbit camera, advanced rendering engine with illustrative colouring and lighting, sensor data integration, distance measurements and visualization and interactive cross-sectioning.

Further work will involve better integration of additional information, including a visual clue of the robot size for quick decision about its ability to travel through a narrow passage. The rendering engine is also currently being transferred into the Direct3D 11 API. This improvement will allow for example the use of *geometry shaders* (first introduced in Direct3D 10) for additional effects and optimizations.

ACKNOWLEDGMENT

The project has been carried out in a framework of an EU programme of the Re-search fund for Coal and Steel under the grant agreement RFCR-CT-2014-00002.

REFERENCES

Blanco, J L, 2013. Efficiently rendering point clouds of millions of points [online]. Available from: <http://www.mrpt.org/tutorials/programming/gui-windows-and-3d-opengl-graphics/efficiently_ rendering_point_clouds_of_millions_of_points/> [Accessed: 21 November 2016]

Gomathi, V, Sowmeya, S, Avudaiammal, P, 2015. Design of an adaptive coal mine rescue robot using wireless sensor networks. *International Journal of Computer Applications*

Moczulski, W, Cyran, K, Novak, P, Rodriguez, A and Januszka, M, 2014. TeleRescuer – A concept of a system for teleimmersion of a rescuer to areas of coal mines affected by catastrophes. *VI. Międzynarodowa Konferencja Systemy Mechatroniczne Pojazdów i Maszyn Roboczych 2014*

Novák, P, Babjak, J, Kot, T and Olivka, P, 2015a. Exploration mobile robot for coal mines in *Modelling and simulation for autonomous systems, International Workshop, MESAS 2015,* Prague, Czech Republic, April 29-30, 2015, pp:209-215, ISBN 978-3-319-22383-4.

Novák, P, Babjak, J, Kot, T and Moczulski, W, 2015b. Control system of the mobile robot TELERESCUER. *Applied Mechanics and Materials*. Vol. 772, pp:466-470

Olivka, P, Mihola, M, Novák, P, Kot, T and Babjak, J, 2016. The 3D laser range finder design for the navigation and mapping for the coal mine robot in *Proceedings of the 17th International Carpathian Control Conference ICCC 2016*. ISBN 978-1-47-993528-4

PCL, 2016a. Downsampling a PointCloud using a VoxelGrid filter [online]. Available from www: <http://pointclouds.org/documentation/tutorials/ voxel_grid.php> [Accessed: 21 November 2016]

PCL, 2016b. PCL – Point Cloud Library [online]. Available from: <http://pointclouds.org/> [Accessed: 21 November 2016]

PCL, 2016c. Removing outliers using a StatisticalOutlierRemoval filter [online]. Available from: <http://pointclouds.org/documentation/tutorials/ statistical_outlier.php> [Accessed: 21 November 2016]

PCL, 2016d. Smoothing and normal estimation based on polynomial reconstruction [online]. Available from www: <http://pointclouds.org/ documentation/tutorials/resampling.php> Ray, D, Majumder, S, Maity, A, Roy, B and Karmakar, S, 2015. Design and development of a mobile robot for environment monitoring in underground coal mines in *Proceedings of the 2015 Conference on Advances in Robotics*. ISBN 978-1-4503-3356-6

Rusu, R B, Willow, G and Park, M, 2011. 3D is here: Point Cloud Library (PCL) in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* pp:1-4. ISBN 978-1-61284-386-5.

Universität Karlsruhe, 2011. Point Cloud Representation [online]. Available from: <http://geom.ivd.kit.edu/ downloads/pubs/pub-linsen_ 2001.pdf> [Accessed: 21 November 2016]

Wikipedia, 2016. Octree [online]. Available from <https://en.wikipedia.org/wiki/Octree> [Accessed: 21 November 2016]