HUNGARIAN JOURNAL OF
INDUSTRY AND CHEMISTRY
Vol. 48(1) pp. 25–31 (2020)
hjic.mk.uni-pannon.hu
DOI: 10.33927/hjic-2020-05

HUNGARIAN JOURNAL OF INDUSTRY AND CHEMISTRY
HJIC

# LIDAR-BASED COLLISION-FREE SPACE ESTIMATION APPROACH

MIKLÓS UNGER*[1], ERNŐ HORVÁTH[1], AND CSABA HAJDU[1]

[1]Research Center of Vehicle Industry, Széchenyi István University, Egyetem tér 1, Győr, 9026, HUNGARY

As autonomous technologies flourish within the vehicle industry, an increasing number of academic autonomous competitions are appearing. One of them is the Shell Eco-marathon Autonomous Urban Concept competition (SEM AUC) which seeks to provide hands-on experience for the academic community to design, build and test their own driverless vehicles within a realistic infrastructure. The team at our university participates in this competition and our concept is to rely on simple and robust algorithms. This paper presents a simple collision-free space estimation algorithm for the LiDAR sensor.

**Keywords:** Autonomous vehicle, Voronoi, Delaunay

## 1. Introduction

Nowadays autonomous technology is not only important in the automotive industry but increasingly in the fields of academia and reasearch as well. A sign of this tendency is the increasing number of academic competitions, one of which is the Shell Eco-marathon Autonomous Urban Concept competition (SEM AUC).

### 1.1 Shell Eco-marathon

The Shell Eco-marathon is a competition for engineering students to design and build the most ultra-efficient car. The participating teams come from all over the world. Two main classes of vehicles compete, namely Urban Concepts and prototypes. Both classes can participate in the Autonomous category where five different challenges await the teams as follows:

- Complex track section
- Maneuverability
- Obstacle avoidance
- The unknown challenge
- Autonomous distance

In this paper, a simple solution for the *autonomous distance* challenge is presented.

### 1.2 Our team

The SZEnergy Team is comprised of students from Széchenyi István University, assisted by tutors and researchers from the fields of the development and construction of electric vehicles. Since 2008, the Team has been competing each year in the Shell Eco-marathon, the biggest fuel-efficiency competition in the world. Our Team participates in the electric vehicle category of the Urban Concept class by performing autonomous challenges and regular races using the same car. Since 2019, a new car called SZEmission has been used as shown in Fig. 1.


*Figure 1:* Our car - SZEmission.

## 2. The challenge

Challenges of the AUC will be undertaken on the same standard track as the other competitions at the SEM or on similar but modified track sections. The track is 970 m long and includes inclines since the track partially consists of closed-off roads (Fig. 2). For all five challenges, protective barriers are installed on both sides of the track.

The barriers are 0.5 m high separated by small gaps, though the size of these gaps is unspecified. Contact with these barriers is forbidden, cars are limited to a top speed of 25 km/h and every car is driven separately on the track. The minimum width of the track is 6 meters unless otherwise specified.

*Correspondence: unger.miklos@ga.sze.hu

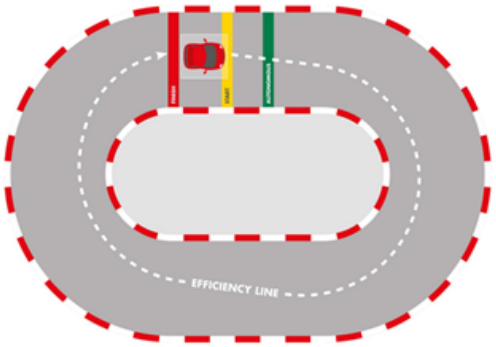


*Figure 2:* Simple layout of the challenge.

Our goal is to create an algorithm that despite the noisy surroundings robustly finds the coordinates of free route between the barriers without touching them based on the output of 3D LIDAR signals. The route must also be unambiguous.

## 3. LiDARs

LiDAR stands for Light Detection and Ranging or Laser Imaging Detection and Ranging and is a sensor commonly used in autonomous vehicles to map the environment. The sensor contains a transmitter and receiver. The transmitter fires laser beams towards the ground which interact with the objects of the environment then are deflected back to the receiver. The distance can be computed using a simple formula, where $d_{\mathrm{o}}$ denotes the distance to the object, $c_{\mathrm{l}}$ represents the speed of light, and $\Delta t$ stands for the time of flight:

$$d_{\mathrm{o}} = \frac{c_l \Delta t}{2}, \tag{1}$$

where the time of flight is measured indirectly by determining the phase shift between the transmitted and received signals, $f_{\mathrm{mod}}$ denotes the frequency modulation, $\varphi_{\mathrm{r}}$ represents the measured difference between the transmitted and received wavelengths [1], and

$$\Delta t = \frac{\varphi_{\mathrm{r}}}{2\pi f_{\mathrm{mod}}}. \tag{2}$$

**LiDAR sensors on autonomous vehicles** Autonomous vehicles use these rotating beam sensors to scan the environment and detect obstacles. Two main types of LiDARs, the 2D and 3D, are used. 2D LiDARs, also referred to as LiDARs with one channel, provide information about the environment in one plane only, hence the objects above and below the LiDARs are not detected. 3D LiDARs use more channels (16, 32, 64 or even 128), thus yield all $x$, $y$ and $z$ coordinates of the surroundings.

Velodyne's Puck (VLP-16) is a LiDAR with 16 channels that give a 360° three-dimensional view.

## 4. Simulation with ROS

### 4.1 ROS in general

Autonomous vehicles can be regarded as four-wheeled robots. In our project, an ROS (Robot Operating System) is used. It is an open-source, flexible framework for writing robot software. The most important components of it are as follows:

**Topics** Topics are named buses, in which data is exchanged using ROS messages. Each topic has a specific name, moreover, one node publishes data to a topic and another node reads the data from the topic by subscribing to it.

**Messages** Every topic consists of a type of message, moreover, topics send and receive data in the form of ROS messages. ROS messages form a data structure used by ROS nodes to exchange data. Different topics send different types of messages, namely 2D LiDARs use LaserScan's, while 3D LiDARs use Point Cloud's.

**Rosbags** Bags are a useful utility for the recording and playback of ROS topics. While working on autonomous vehicles, some situations may arise where it is necessary to work without actual hardware. Using rosbags, sensor data can be recorded and bag files copied to other computers to inspect data by playing it back.

**Gazebo** Gazebo stands for open source robotic simulators tightly integrated with ROS. In this environment, various types of robots, indoor sensors and outdoor elements could be implemented. The values of sensors can be accessed by the ROS through topics.

**Rviz** Rviz is a 3D visualizer in ROS to visualize 2D and 3D values from ROS topics and parameters which helps to visualize data such as robot models, robot 3D transform data (TF), point clouds, laser and image data, as well as a variety of sensor data [2].

### 4.2 Simulation

Simulations are based on models with which we try to copy the physics and geometry of the real world. The more factors that are built into them, the more likely what is expected to happen will actually occur in a real-world test.

Our car acts as a robot, modelled on a CAD program. In this phase of the algorithm, the layout of the car is not important so a Nissan Leaf was used as our race car. In Gazebo, numerous types of sensors are available, the parameters of which can be defined as real-world LiDAR signals as mentioned in Section 3. Unfortunately, the exact layout of the racetrack is unknown, so one was created. Our track consists of straight sections, curves with

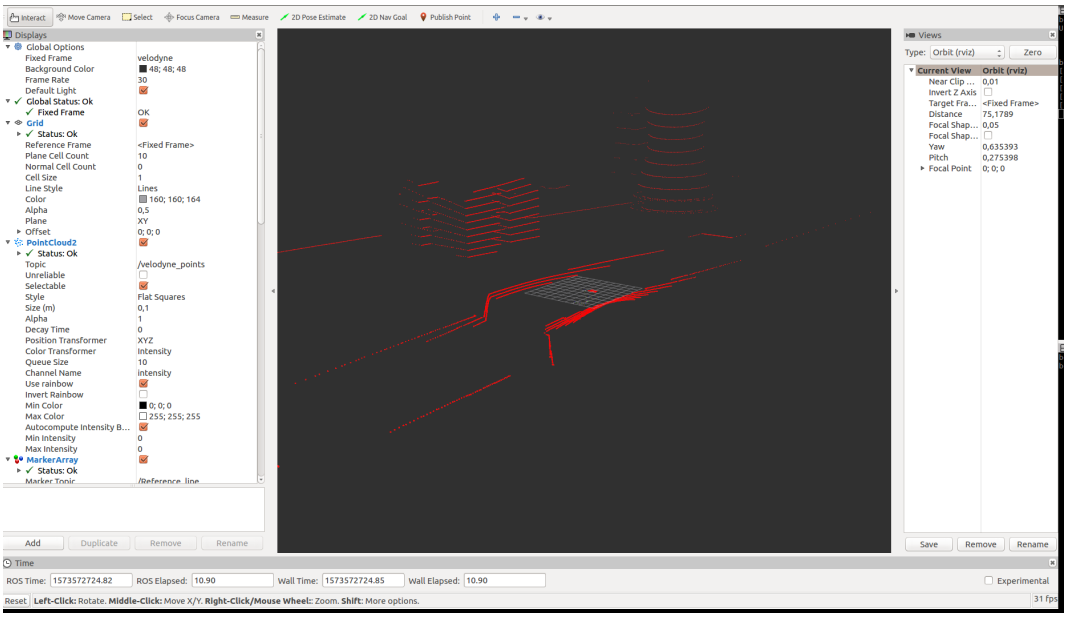

*Figure 3:* Simulation in Rviz. Different topics can be visualized at the same time. It can be seen that the 3D LiDAR point cloud originates from two 2D beams, one above the other.

different radii as well as white and red protective barriers on both sides.

As 3D sensors are used, not only the track but its surroundings are also important. Therefore, buildings, trees in addition to cones between and beyond the two barriers were implemented. When the simulation was completed, it was possible to drive a lap of the track in the simulation and collect the ROS topics data via rosbags. Moreover, with Rviz, the following points could be visualized as presented in Fig. 3.

## 5. Steps of the algorithm

### 5.1 Filtering

Our LiDAR sensor gives information over 360 degrees which is very useful in many cases, however, in this challenge, only information about points in front of the car is needed. If information about displacement of the sensors on the car is available, it can be assumed that the front of the car and all points smaller than the linear equation

$$x = \frac{-b + y}{m} \tag{3}$$

will be deleted, where $b$ denotes the $y$-intercept and $m$ represents the gradient.

### 5.2 Clustering

As from our simulation only raw data (only $x$, $y$ and $z$ coordinates) are obtained, which points belong to each object has to be defined. Clustering is the task of dividing the population or data points into a number of groups such that the data points in a group are more similar to other data points in the same group than to data points in other groups. It is basically a collection of objects arranged according to how similar or dissimilar they are to each other.

For us the most effective method is DBSCAN (Density-Based Spatial Clustering of Applications with Noise), a well-known data clustering algorithm that is commonly used in data mining and machine learning. Based on a set of points, DBSCAN groups together points that are in close proximity to each other based on a distance measurement (usually the Euclidean distance) and a minimum number of points. It also marks as outliers the points that are in low-density regions [3]. The DBSCAN algorithm basically requires two parameters:

**eps** specifies how close points should be to each other to be considered as parts of a cluster. This means that if the distance between two points is smaller or equal to this value (eps), these points are considered to be neighbors.

**minPoints** the minimum number of points to form a dense region. For example, if the minPoints parameter is set as 5, then at least 5 points are needed to form a dense region.

After clustering, all of the points belong to a group as shown in Fig. 4.

### 5.3 Convex hull

Even though the point cloud of a 3D LiDAR stands for more than one thousand points, only a proportion of them is needed, thus points should be filtered. The clustered points have to be packed by each group into a convex hull. The convex hull of a set of points is defined as the

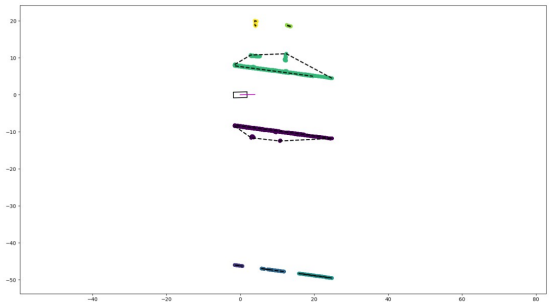

*Figure 4:* The different clusters denoted by different colors.

smallest convex polygon that encloses all of the points in the set [4].

For the purposes of clarification, only the vertices of a convex polygon are used as shown in Fig. 5. Vertices are side points which form the simplified shape of a polygon.

### 5.4 Delaunay triangulation and Voronoi diagram

The Delaunay triangulation for a given set $P$ of discrete points in a plane is a triangulation $\mathrm{DT}(P)$ such that no point in $P$ is inside the circumcircle of any triangle in $\mathrm{DT}(P)$ [5]. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation as presented in Fig. 6.

The circumcenters of Delaunay triangles are the vertices of the Voronoi diagram. In the 2D case, the Voronoi vertices are connected via edges that can be derived from adjacency relations of the Delaunay triangles. If two triangles share an edge in the Delaunay triangulation, their circumcenters are to be connected with an edge in the Voronoi tessellation as shown in Fig. 7.

For a set of points in 2D space, the Voronoi diagram creates cells whose edges are the same distance from two neighboring points. This property grants that all reference points are exactly between two detected items and the edges of Voronoi cells are ideal for the representation of a reference line. Unfortunately, real environments are never ideal and this method would provide more route options in the presence of noise as presented in Fig. 8.

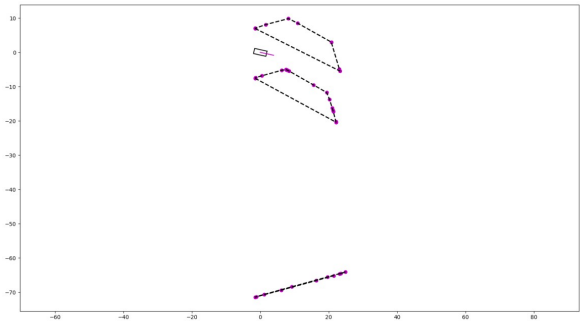

*Figure 5:* To clarify the large number of points, only the vertices of the convex hull were used.

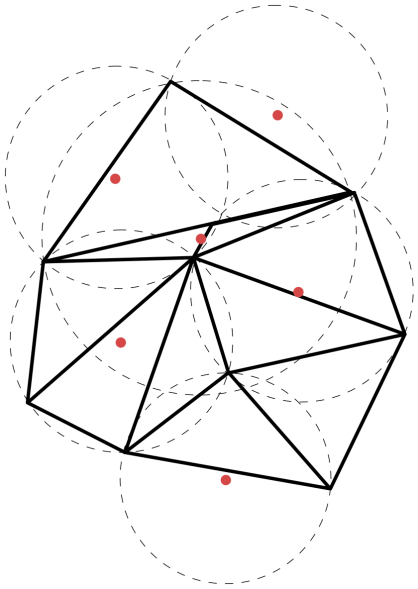

*Figure 6:* The Delaunay triangulation with all the circumcircles and their centers (marked in red).

### 5.5 RANSAC

At this stage of the algorithm, no information is available about which side of the car the detected points are on. As the reference lines should only be between the two middle barriers, this information is important. For this process, an estimator must be implemented.

RANSAC (random sample consensus) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers which do not influence the values of the estimated parameters. The input to the RANSAC algorithm is a set of observed data values, a method of fitting some kind of model to the observations and several confidence parameters. RANSAC achieves its goal by repeating the following steps [6]:

1. Selecting a random subset of the original data, re-

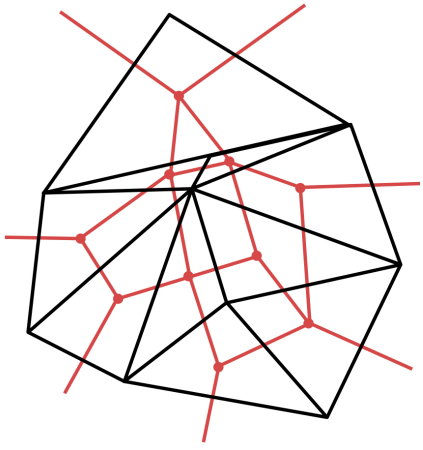

*Figure 7:* A Voronoi diagram is constructed by connecting the centers of the circumstances.

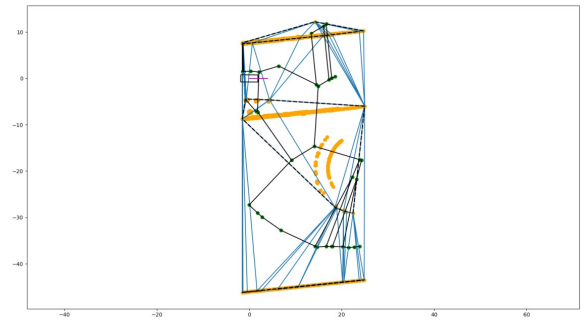

*Figure 8:* A real scenario from the algorithm. The detected points are denoted in orange, the green points represent the centers of the circumstances, and the black lines stand for the Voronoi edges which also represent the reference lines.

ferred to as the hypothetical inliers;

2. Fitting a model to the set of hypothetical inliers;
3. Then testing all other data against the fitted model. Those points that fit the estimated model well, according to a model-specific loss function, are considered to be part of the consensus set.
4. The estimated model is reasonably good if a sufficient number of points are classified as part of the consensus set.
5. Finally, the model may be improved by re-estimating it using all members of the consensus set.

If RANSAC is used for the set of Voronoi midpoints, a line is obtained between the protective barriers on either side of the track (Fig. 9). Now the linear equation for $y$ with its gradient can be written:

$$y = mx + b, \tag{4}$$

where $b$ denotes the $y$-intercept and $m$ represents the gradient.

Points with higher and lower values of $y$ belong to the left- and right-hand sides, respectively.

As RANSAC is an estimator, sometimes it yields wrong results as shown in Fig. 10.

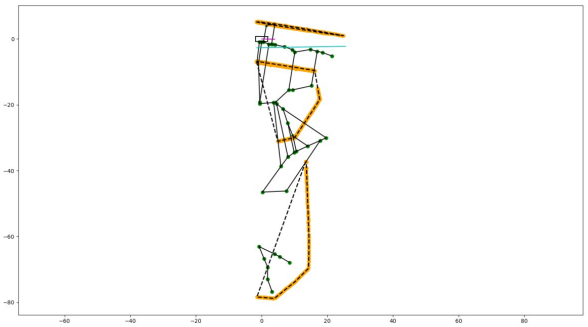

*Figure 9:* The RANSAC line (denoted in blue) splits the space into two parts.

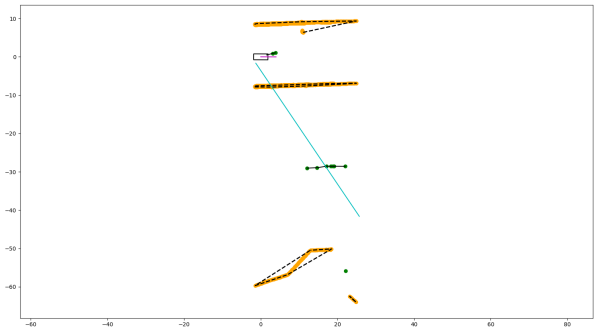

*Figure 10:* The estimator sometimes yields wrong results.

To avoid producing wrong results, some criteria must be declared:

- The starting point of the line should be between the protective barriers.
- The line must be sufficiently long over the full range of the $x$-axis.
- If the RANSAC line intersects any of the lines of the convex hull, it must be rotated until it no longer intersects them.

### 5.6 Delete unnecessary reference lines

One of our main criteria is that the reference line should exclusively be between the two barriers. To ensure that our car moves in such a way, all unnecessary Voronoi midpoints must be eliminated. Now all the points that belong to each side can be collected, however, since they are unsorted, an Andrew's monotone chain convex hull algorithm was used [7]. This algorithm sorted the points into a lexicographical order (first by $x$-coordinates, and should any be equal, by $y$-coordinates) and then constructed upper and lower hulls of the points as seen in Fig. 11.

Some naive approaches concerning how to use the upper and lower sides of the convex hulls exist. Firstly, an attempt was made to copy the shape of the upper hull with a single line using Equation (Eq. 4), but was not possible when used at corners.

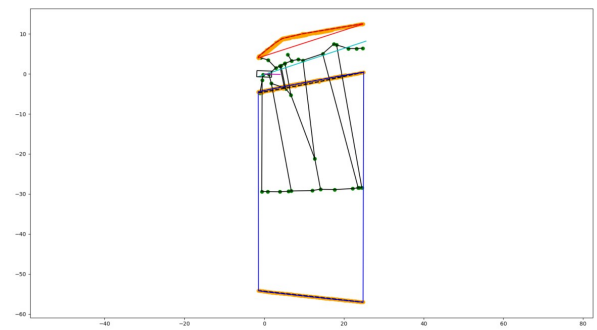

*Figure 11:* The left- and right-hand sides were packed into another convex hull.

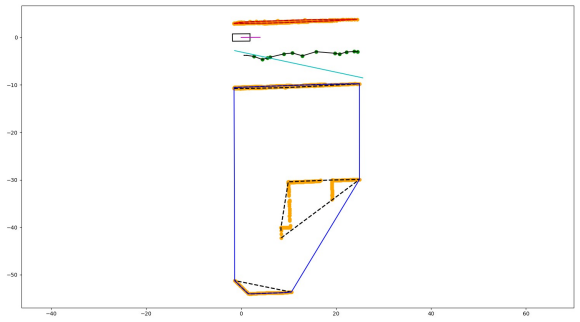

*Figure 12:* The reference line along straight sections affected by noise.

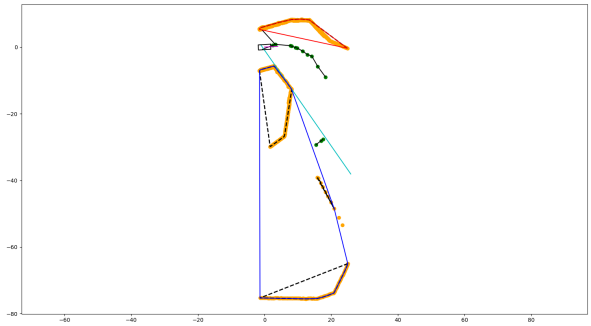

*Figure 13:* The reference line at corners.

Another idea was that since the borderline was not a single straight line, it could be divided into as many sections as the number of vertices found in the upper hull, however, this resulted in a wavy line which in some cases also deleted inlier Voronoi points. Then it was realized that the easiest solution would be to delete all the Voronoi midpoints which lie inside the left and right convex hulls.

## 6. Results and future works

By following the aforementioned steps, the algorithm drew an unambiguous reference line by connecting Voronoi midpoints which lie in the middle of the road of the required territory. Furthermore, this method also works along straight sections and at corners as shown in Figs. 12 and 13.

For safety reasons, vehicle speed is limited to 25 km/h. Our algorithm is designed to operate at this speed range. The algorithm is based on Voronoi cells so its complexity is $O(n)$ [8]. In our test environment the algorithm could estimate the free space with 90 % confidence.

With the aid of Voronoi diagrams, every detected object has an effect on our reference line. Fortunately, the majority of these points are filtered by the steps outlined in Section 5.6, however, outliers between the barriers cannot be dealt with. These points have a detrimental effect on the unambiguity of the reference line and lead to the creation of several nodes resulting in some multiple-choice decisions that need to be made by our car. To make our algorithm robust, these points should be eliminated.

Now a single RANSAC line divides the sides of the road and works well for the layouts of simple corners, however, one simple line cannot satisfy the criteria for more difficult layouts, e.g. chicanes, because the RANSAC line is unable to intersect any lines of the convex hull. To avoid this, the line should be divided into smaller sections that need to follow on from each other and determine if any parts of the linked line intersect the line of the convex hull.

Our path-following algorithm requires that the reference points which follow on from each other are the same distance apart. For this purpose, our reference points must be joined into one line and as many points as the algorithm requires must be generated.

This algorithm has only been examined using test data and has yet to be tested in real environments.

## 7. Conclusion

In 2020, our team will participate in the autonomous distance challenge of the Shell Eco-marathon. The results of this algorithm are promising but a significant amount of research and development has yet to be done until this algorithm can operate and lead our car around a single lap. One of the benefits of the proposed method is that it involves widely used and, over time, highly improved geometric algorithms like Delaunay triangulation and Voronoi diagrams.

## Acknowledgements

The research was carried out as part of the “Autonomous Vehicle Systems Research related to the Autonomous Vehicle Proving Ground of Zalaegerszeg (EFOP-3.6.2-16-2017-00002)” project in the framework of the New Széchenyi Plan. The completion of this project is funded by the European Union and co-financed by the European Social Fund.

## REFERENCES

[1] Verőné Wojtaszek, M.: Fotointerpretáció és távérzékelés 3., A lézer alapú távérzékelés. 2010 Nyugat-magyarországi Egyetem

[2] Lentin, J.: ROS robotic projects. (Packt Publishing Ltd., Birmingham, UK) 2017, ISBN: 978-1-783-55471-3

[3] Kriegel, H.-P.; Kröger, P.; Sander, J.; Zimek, A.: Density-based clustering, *WIREs Data Mining Knowl. Discov.*, 2011, **1** 231–240 DOI: 10.1002/widm.30

[4] Chazelle, B.: An optimal convex hull algorithm in any fixed dimension, *Discrete Comput. Geom.*, 1993, **10** 377–409 DOI: 10.1007/BF02573985

[5] de Berg, M.; Cheong, O.; van Kreveld, M.; Overmars, M.: Computational geometry: Algorithms and applications (Springer-Verlag, Berlin, Germany) 2008, DOI: 10.1007/978-3-540-77974-2

[6] Fischler, M. A.; Bolles, R. C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Commun. ACM*, 1981, **24**(6) 381–395 DOI: 10.1145/358669.358692

[7] Andrew, A. M.: Another efficient algorithm for convex hulls in two dimensions, *Inf. Process. Lett.*, 1979, **9**(5) 216–219 DOI: 10.1016/0020-0190(79)90072-3

[8] Seidel, R.; Adamy, U.; On the exaxt worst case query complexity of planar point location, *J. Algorithms*, 2000, **37** 189–217 DOI: 10.1006/jagm.2000.1101