Theses booklet

Research and optimization of perception and trajectory planning algorithms for autonomous mobile robots and road vehicles

Autonóm mobil robotok, valamint közúti járművek környezetészlelését és trajektóriatervezését megvalósító algoritmusok kutatása és optimalizálása

Ernő Horváth – Doctoral Theses booklet

Modeling and Development of Infrastructural Systems Doctoral School of Multidisciplinary Engineering Sciences

CONTENT

1. Introduction

It is a fascinating fact that we humans can solve such complex tasks every day without much attention, whereas we can hardly imitate these behaviours with the most up-to-date sensors, actuators and algorithms. We are showing some serious motoric skills in hectic traffic, during preparation of a simple meal or during sport activity. On the other hand, non-complex algorithms easily overperforms us in complicated logic games or even intelligence tests. Similar paradoxical phenomenon was observed by Hans Moravec [1] in the '80s. According to Moravec's paradox, the imitation of a low-level sensory motor skills of up to a few years old can require enormous computing resources, while a simpler chess program can even easily overcome an adult. Evaluating the difficulty of a given task is not always clear or sometimes conditions are involved. In my research I always treated this fact humbled, and my aim is not to change this in the future. Therefore, for example, a mobile robot navigation in a factory or a self-driving vehicle in traffic is facing some serious tasks. In designing and operating such systems, sensing and perception are one of the most important tools. Therefore, my doctoral thesis also targets the problem of robotic perception especially new design perspectives and methodologies. Consequently, aimed to familiarize myself with the methodological approaches of the field and make proposals for their improvement. The whole problem set is further refined by prof. Roland Siegwart, a recognized scientist in the field of artificial intelligence at University of Zürich. He has revealed some considerations [2] in 2018, according to which if it is not counted with the structured gradation of a given task, some of us may go too far when we put the achievements of NN on a pedestal and create false expectations. Nowadays NN systems can be applied only to structurable problems, but of course the research will always focus on stepping over or expanding these limitations. I used NN technology intensively in my own research, but always considering an important conclusion of the article mentioned above. Robotics will always use NN technology, but they also must outspread it.

Currently there is an extensive research tendency regarding autonomous (also known as self-driving) vehicles were the perception and path planning part of my doctoral work is also relevant. At the time of writing besides widespread academic or industry-related research there are several level 4 autonomy start-ups (Voyage, Waymo, Uber, Aptiv, Navya) who provide autonomous taxi services in some cases even without safety drivers, but always with limitations. Also, in driving assistance (level 2) technologies are remarkably advanced, the most well-known example is Tesla. Drago Anguelov [3] who is a Principal Scientist at Google-Waymo views the situation as taming the long tail, because for the most common cases there are existing solutions, but solving extremities is a similar magnitude problem. So, the long tail of event means that it is a hard problem to build algorithms, train your neural networks for the common cases of autonomy, but it is the same or even challenging level of difficulty to solve the rest. In autonomous vehicles NN technology can be used in different levels such as in perception and in decision making.

NN perception is quite common because large amount of - visual or LIDAR - data can be processed this way. Autonomous decision-making tasks such as path planning can also be solved with or without NN. During the years of my PhD studies I was lucky enough to work together with lots of talented students together on autonomous vehicles so some of my theoretical results also applied in practice. I would highlight the Szenergy team at our university who are competing in a word-wide autonomous competition with the help of many lecturers including myself.

In order to bear out my research, I chose the means of comparing the methods acknowledged by the scientific community and the results of my own. Besides that, I also used different evaluation methods and I shared publicly the related source codes. During my PhD studies I was able to publish my results even from small conferences to recognized international or impact factor journals. During my PhD work I had the possibility develop with a numerous programming language, such as MATLAB, Python, C++, C# or LabVIEW. Even more libraries, technologies, APIs, SDKs, tools and middleware helped my work; including but not limited to: TensorFlow, Neural Network Toolbox, ROS, rviz, rqt_plot, V-REP, Gazebo, OpenCV, NumPy, Numba, PyQt, NI Real-Time Module, NI FPGA Module, CUDA and cuDNN.

My doctoral thesis has the following structure. After this introductory section, in the second section the related work and recent results is discussed. This section gives the information required to have an insight to the basis of my work. Here the historical background and the current state of the art is detailed about neural networks, path planning and map representations. My three theses are based on this knowledge, which can be also considered as an extension of this knowledge, and it is detailed in the third section. The rest of the document contains the conclusions, the outlook and the related publication list.

This section overviews the historical background and the current results the related fields to my PhD work. I intend to adumbrate a complete guide to the most important scientific fields. These fields involve but not limited to robotic and self-driving associated tasks such map representations, trajectory planning and execution and neural networks. Also, I briefly describe the vehicles, robots, sensors and hardware used during the validation of the proposed algorithms and methods.

As a rule of thumb, I intend to illustrate every concept with *own measurements and graphics*. The section gives a general intuition about his field but focuses on the closely related architectures of the doctoral work. In addition, the limits and the drawback of the techniques is discussed alongside with some possible solutions.

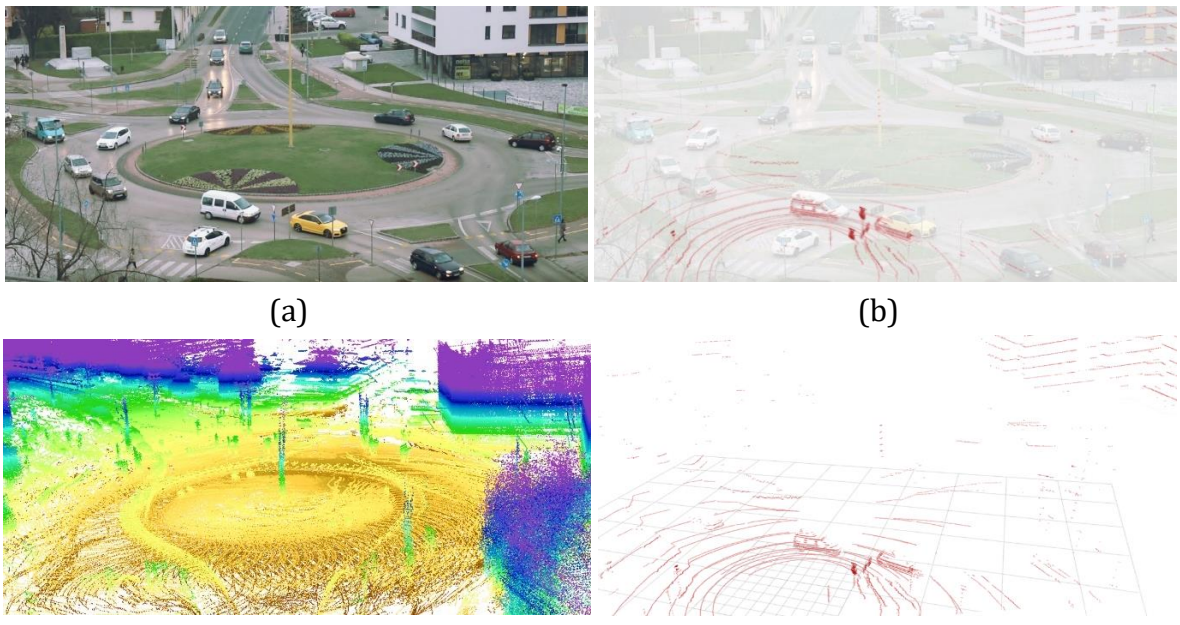
1.1. Robots, vehicles and hardware used

During my PhD work I was lucky enough to collect experiences from variety of robots and vehicles. I always had possibility to try out my new algorithms on mobile robots such as Khepera III, Neobotix MP500 and TurtleBot 3. These robots were MATLAB and ROS

compatible, if not by default, after modification. I also had access to the research centre's vehicles, the Szenergy vehicles, and the Nissan Leaf.

Figure 1 - Robots and vehicles: Turtlebot, Szenergy and Nissan Leaf

These robots and vehicles used different sensors, hardware and architectures. This was often a challenge, but as a result of working with these technologies I learned how to get along with a variety of tools. To arbitrary highlight the most distant technologies I used FPGA and real-time OS for the most time critic control and data acquisition tasks, on the other hand I used different script languages for visualization and data processing.
From perception point of view I used LIDAR, radar, ultrasonic and (depth) camera sensors.



(a) (b)

(c) (d)

Figure 2 - An example scenario: the Nissan equipped with 2x16 channel LIDARs (a) RGB image (b) point cloud and RGB (c) concatenated point cloud (d) single measurement – at the Széchenyi University Campus, Győr

In the discussed theses I mostly rely on LIDAR and camera and GPS data and from these sources my algorithms realize trajectory planning, trajectory execution and perception for self-driving vehicles and autonomous robots.

2. Contribution

This section is devoted to detail my scientific contribution. In the following three theses are proposed. The first one is a method for path construction. The second is a trajectory following method. The third one is a signature-based sensory model. The last one is a neural network phenomenon and a recommendation based on that.

2.1. Thesis 1.

I proposed a method to solve the Coverage Path Planning problem (CPP) called Iterative Structured Orientation Coverage (ISOC) and showed that it can lead to a shorter trajectory compared to the commonly used Boustrophedon Cellular Decomposition Coverage (BCDC). In the context of this work, I elaborated three different solutions where the first two relies on the rotation of parallel lines, whereas the third solution uses the concept of inertia. The suggested approaches are validated by simulation and experimental results.
References: [H19], [H14]

Coverage path planning (CPP) aims to cover a certain area with the shortest movements possible. This problem appears in the various domains such as agricultural applications, painting robots or cleaning robots. CPP makes use of two classes [4]: it is complete if it guarantees complete coverage or heuristic in other cases. There are also two main CPP strategies: offline if there is an a priori known map, otherwise online if the robot needs to discover the environment. The most extensively used approaches to CPP are Random Path Planning (RPP) [5], Exact Cellular Decomposition (ECD) [6], Boustrophedon Cellular DeComposition (BCDC) [6], Backtracking Spiral Algorithm (BSA) [5], Internal Spiral Search (ISS) [7], U-turn A* Path Planning (UAPP) [7] and Neural Network (NN)-based CPP [8]. The proposed approach (ISOC) uses the concept of main lines to cover the map (domain). The main lines are parallel to each other, their distances are related on the sensory range, because the coverage is connected to the range sensor's perception. Based on the main lines, the main segments can be created concluded by the structure of the map. The final trajectory is composed with the help of the auxiliary segments. These auxiliary segments intend to connect the main segments and thus make one single continuous trajectory. According to this approach, the movement mostly consists of the long main segments, which is one of the reasons why the Iterative Structured Orientation Coverage can lead to a shorter trajectory compared to the widely used Boustrophedon Cellular Decomposition Coverage [6]. Figure 3 (a) shows the creation of main lines and

Figure 3(b) shows how to construct the main segments and connect it with the auxiliary segments to compose the trajectory.

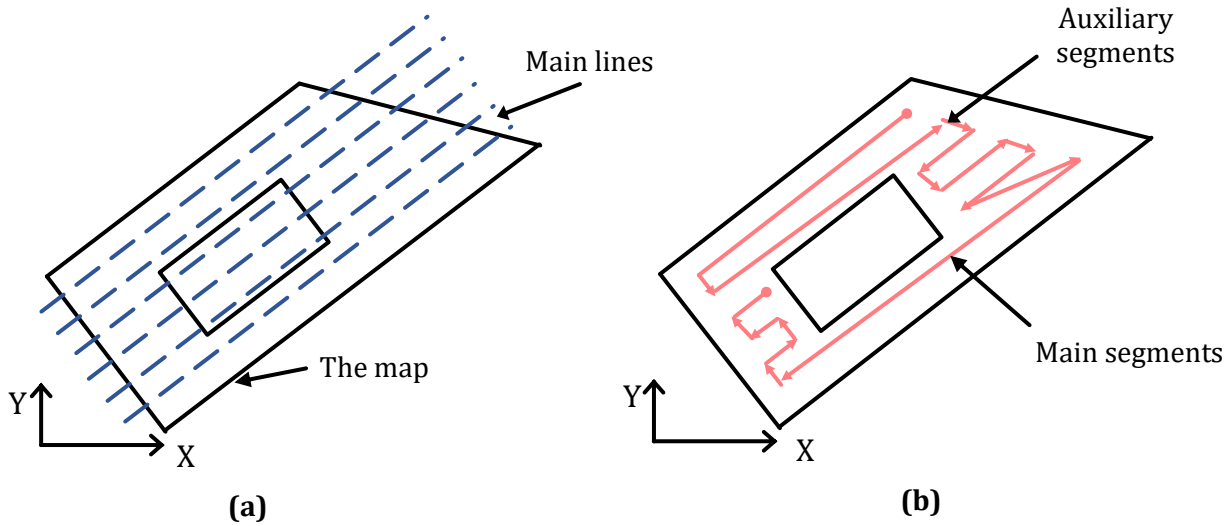


Figure 3 - Illustration of the main lines (a) and the main and auxiliary segments (b)

According to my best knowledge there was no realization of the Boustrophedon cellular decomposition for grid maps, even the realization is straightforward from line map to grid map. In order to appropriately compare the iterative structured orientation coverage (ISOC) with boustrophedon cellular decomposition coverage (BCDC) the same input set (maps) and the same programming environment (MATLAB) needed to be used. Figure 4 shows how important is to choose the right orientation. On Figure 4 (a) shorter but more main segments are determined in contrast on Figure 4 (c) where longer but less main lines are present. The possible connection is visualized on (b) and (d) which shows an example where the less main lines results the shortest final trajectory.

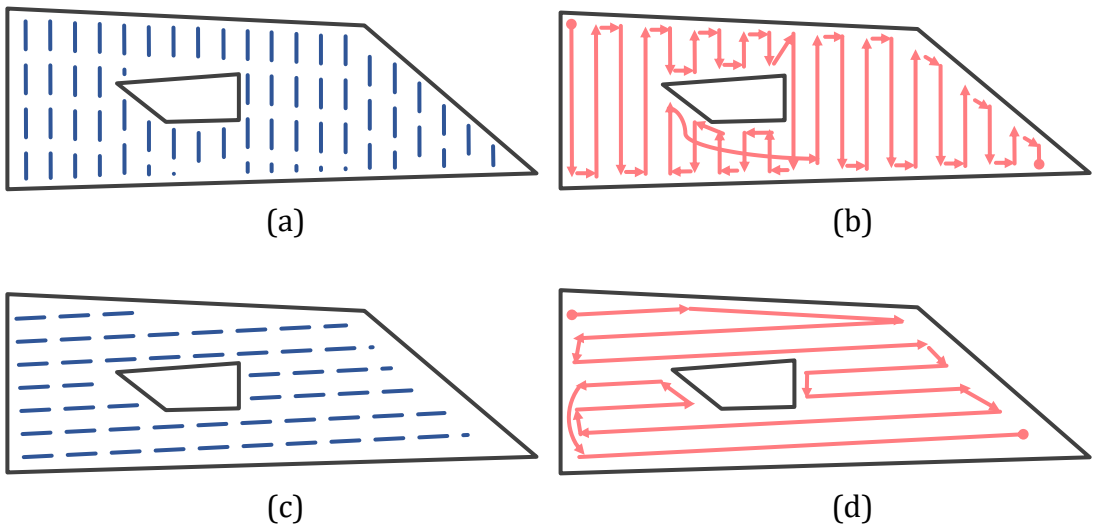


Figure 4 - Visualization of ISOC with two orientations, main lines are presented on (a) and (b), the whole path is on (b) and (d)

The initial data of the CPP approach is the map. The map is generated from a picture and modeled by the map matrix $M = [M_{\text{ij}}]_{i=1...n,j=1...m} \in \Re^{n\times m}$, with the elements:

$$M_{\text{ij}} = \begin{cases} 0, & \textit{if} \text{ cell } (i,j) \text{ is free } (white) \\ ]0,1[& \text{likelihood of occupancy} \\ 1, & \textit{if} \text{ cell } (i,j) \text{ is occupied } (black) \end{cases} \quad (1)$$

where n is the number of horizontal pixels and m is the number of vertical pixels.
Using the main lines concept, the problem of finding a minimum length path, which covers the whole map, reduces to the following 3 steps:

1. Find the appropriate main line, i.e., the orientation of the beam of parallel lines.
2. Construct main segments.
3. Link these segments with auxiliary segments such that the final (continuous) path has a minimum length.

These steps will be detailed in the following and later as a unified algorithm which is a part of the ISOC. For step 1 three different solution will be presented noted with solution "A", "B" and "C". The equation of the beam of parallel lines expressed in the discrete domain is:

$$\begin{cases} L_{q_1k_1} \equiv i = \left\lfloor j\frac{q_1}{(m+1)} \right\rfloor + k_1 + 1, & \text{if } \alpha \in \left[-\frac{\pi}{4}, \frac{\pi}{4}\right], \\ L_{q_2k_2}j = \left\lfloor i\frac{q_2}{(n+1)} \right\rfloor + k_2 + 1, & \text{if } \alpha \in [-\frac{\pi}{2}, -\frac{\pi}{4}) \cup (\frac{\pi}{4}, \frac{\pi}{2}] \end{cases} \quad (2)$$

where α is the line slope in the continuous domain, $\alpha = \tan^{-1}\left(\frac{q_1}{m+1}\right)$ or $\alpha = \tan^{-1}\left(\frac{q_2}{n+1}\right)$, $q_1 = 1...m$ or $q_2 = 1...n$ have a direct effect on the slope in the discrete domain, with the unified notation $q \in \{q_1, q_2\}$, $\lfloor x \rfloor$ indicates generally the integer part of $x \in \Re$, $k_1 = \beta_1\delta_1$ or $k_2 = \beta_2\delta_2$ is the intercept with the unified notation $k \in \{k_1, k_2\}$ for both horizontal and vertical axes, $\beta_1 = 0...(n-1)/\delta_1$, $\beta_2 = 0...(m-1)/\delta_2$ the integer steps δ_1 and δ_2 are computed in terms of

$$\delta_1 = \left\lfloor b\frac{\sqrt{q_1^2+(m+1)^2}}{m+1} \right\rfloor, \ \delta_2 = \left\lfloor b\frac{\sqrt{q_2^2+(n+1)^2}}{n+1} \right\rfloor, \quad (3)$$

where b is the distance between the lines (the robot width), $L_{q_1k_1}$ and $L_{q_2k_2}$ are the lines that belong to the beam with the unified notation $L_{\text{qk}} \in \{L_{q_1k_1}, L_{q_2k_2}\}$ for both axes. Each line can be associated with a matrix $\Lambda^{\text{qk}} = [\Lambda_{\text{ij}}^{\text{qk}}]_{i=1...n,j=1...m} \in \Re^{n\times m}$ with the elements $\Lambda_{\text{ij}}^{\text{qk}}$

$$\Lambda_{\text{ij}}^{\text{qk}} = \begin{cases} 1, & \text{if } (i,j) = (L_{q_1k_1}, j) \wedge (i,j) = (i, L_{q_2k_2}), \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Three solutions for the computation of the main lines are proposed in this thesis. The first two solutions generate a beam of lines and define the main segments as the intersection between the map and the beam of lines, compute the segments length and compute (or approximate) the path length in terms of the sum of these lengths. An iterative process is conduced to compute the slope of the beam of lines, which generates a set of path lengths.

The result, represented the main segments, is related to the minimum length path as the solution to the optimization problem

$$q^* = \underset{q \in D_q}{\operatorname{argmin}} \sum_{k \in D_k} \Gamma\left(\Lambda^{\mathrm{qk}}, M\right), \tag{5}$$

where q^* gives the optimum slope in the discrete domain, D_q is the discrete domain of slope, D_k is the intercept domain, the general notation $\Gamma(\Lambda^{\mathrm{qk}}, M)$ is the general notation for the path length:

$$\Gamma\left(\Lambda^{\mathrm{qk}}, M\right) = \lambda \sum_{i=1}^{n} \sum_{j=1}^{m} p_{\mathrm{ij}}, \;\; p_{\mathrm{ij}} = \begin{cases} 0, & \text{if } M_{\mathrm{ij}} = 1, \\ 1, & \text{if } \Lambda_{\mathrm{ij}}^{\mathrm{qk}} = 1, \end{cases} \tag{6}$$

where the general notation $\lambda \in \{\lambda_1, \lambda_2\}$ is used for the distance between the points calculated as

$$\begin{aligned} \lambda_1 &= \sqrt{\frac{(m+1)^2 + q_1^2}{m+1}}, \\ \lambda_2 &= \sqrt{\frac{(n+1)^2 + q_2^2}{n+1}}. \end{aligned} \tag{7}$$

2.1.1 Solution "A" for construction of the main lines

Solution "A" is based on the parallel lines rotation, this solution consists of the following steps:

1.1 The lines expressed in (2) which depend on q and k, are generated.

1.2 The matrices Λ^{qk} with the elements $\Lambda_{\mathrm{ij}}^{\mathrm{qk}}$ expressed in equation (4) are generated.

1.3 The path length $\Gamma(\Lambda^{\mathrm{qk}}, M)$ is computed according to (6).

1.4 The objective function in (5) is computed in terms of the sum $\sum_{k \in D_k} \Gamma(\Lambda^{\mathrm{qk}}, M)$ for $q = \text{const}$ and variable $k, \;\; k \in D_k$.

1.5 The optimization problem defined in (5) is solved considering that the objective function in the right-hand term of (4) depends on the variable $q, \;\; q \in D_q$, and the solution to this optimization problem, i.e., the variable that gives the minimum path length, is q^*.

The first two solutions differ by the slope domain and by the map definition. The first solution preserves the initial map and defines a continuous domain of slope $D = [0, \pi]$ in order to include all possible orientations of the beam of parallel lines.

2.1.2 Solution "B" for construction of the main

Solution "B" is also based on parallel lines rotation. Tis solution defines a new map using a composition of the initial map and uses a smaller domain of slope, i.e. $D = [0, \pi/4]$. Solution "B" is based on the generation of a new map by the union of four maps that are rotated. These four maps correspond to the four quadrants I, II, III and IV, obtained as follows. The map in the quadrant I is $M_I = [M_{I,\mathrm{ij}}]_{i=1\ldots n, j=1\ldots m} \in \Re^{n \times m}$, with the map matrix elements

$$M_{I,\mathrm{ij}} = M_{\mathrm{ij}}, i = 1\ldots n, j = 1\ldots m. \tag{8}$$

The map in the quadrant II is $M_{\mathrm{II}} = [M_{\mathrm{II,ij}}]_{i=1\ldots n, j=1\ldots m} \in \Re^{n\times m}$, with the map matrix elements

$$M_{II,\mathrm{ij}} = M_{\mathrm{im}-j+1}, i = 1\ldots n, j = 1\ldots m. \quad (9)$$

The map in the quadrant III is M_{III}, obtained in terms of the composition

$$M_{\mathrm{III}} = [P|M^T] \in \Re^{m\times m},\ P = [P_{\mathrm{ij}}]_{i=1\ldots m, j=1\ldots m-n},\ P_{\mathrm{ij}} = 1, \quad (10)$$

where the subscript T indicates matrix transposition.

The map in the quadrant IV is $M_{\mathrm{IV}} = [M_{\mathrm{IV,ij}}]_{i=1\ldots n, j=1\ldots m} \in \Re^{n\times m}$, with the map matrix elements

$$M_{I,\mathrm{ij}} = M_{\mathrm{in}-j+1}, i = 1\ldots n, j = 1\ldots m. \quad (11)$$

Solution "B" consists of the following steps:

2.1. The map matrix in the four quadrants is computed using (8) to (11).

2.2.-2.6. These are the steps 1.1 to 1.5 in the first solution.

2.1.3 Solution "C" for construction of the main lines

Solution "C" approximates the main lines with the map axis, which is inspired from the properties specific to mechanical inertia. The map axis slope is obtained in terms of

$$\alpha = \frac{1}{2}\tan^{-1}\left(\frac{2I_{\mathrm{xy}}}{I_y - I_x}\right), \quad (12)$$

where the following center of gravity-type relationships are employed:

$$I_{\mathrm{xy}} = \sum_{i=1}^{n}\sum_{j=1}^{m} i_c j_c M_{\mathrm{ij}},\ I_x = \sum_{i=1}^{n}\sum_{j=1}^{m} j_c^2 M_{\mathrm{ij}},\ I_y = \sum_{i=1}^{n}\sum_{j=1}^{m} i_c^2 M_{\mathrm{ij}},$$

$$x = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{jM_{ij}}}{\sum_{i=1}^{n}\sum_{j=1}^{m} M_{\mathrm{ij}}},\ y = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} \mathrm{iM_{ij}}}{\sum_{i=1}^{n}\sum_{j=1}^{m} M_{\mathrm{ij}}}, \quad (13)$$

$$i_c = i - y,\ j_c = j - x,\ i = 1\ldots n,\ j = 1\ldots m$$

and M_{ij} are the elements of the map matrix M defined in (1). The beam of parallel lines is next computed using (2). Solution "C" consists of the following steps:

3.1. The map axis slope is computed using (12) and (13).

3.2. - 3.6. These are the steps 1.1 to 1.5 in the first solution.

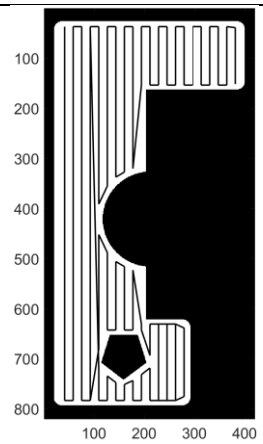
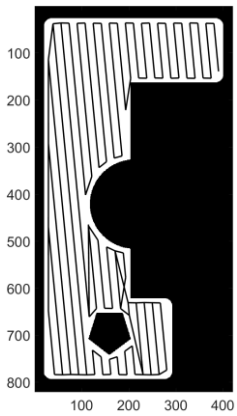
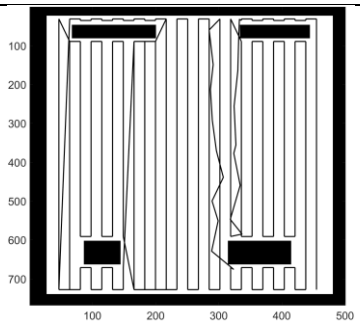
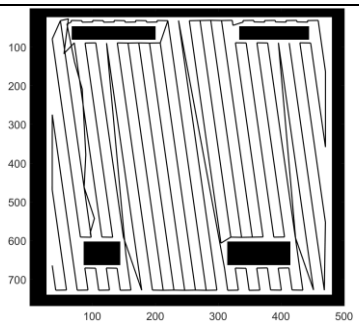
2.1.4 Construction of the auxiliary segments

In the previous section three solution is presented (A, B, C) and all of the solution provides an organised set of main lines. The lines order has been defined in the process of definition of the main lines using the intercept of each line expressed in (2). The definition of the main segments splits the lines in several segments producing a list of segments ordered (in their turn) using a left to right convention. This means that each segment has two kinds of neighborhoods, i.e., the segments that belong to the lists of neighbor main lines and the segments that do not belong to the same list. The second type of segments is eluded because of the obstacles between these segments.

2.1.5 Conclusion and evaluation

This section validates the ISOC approaches presented in the previous section by simulations and experiments conducted on mobile robots. The validation by simulation includes a comparison between the proposed approaches, and of the proposed approaches with another well-known approach discussed in Section 1, namely the BCDC approach. The validation by experiments is focused on a Khepera mobile robot.

Table 1 - Comparison between the proposed approaches

Results using solution "A" and "B"	Results using solution "C"
 $\alpha = 90^o, \Gamma = 14443$	 $\alpha = 84.52^o, \Gamma = 13684$
 $\alpha = 270^o, \Gamma = 22575$	 $\alpha = 269.54^o, \Gamma = 23796$

Three types of maps have been used to validate the ISOC algorithm presented in the previous section: a *simulated map* obtained from V-REP, a *real-life measurement* map, and an *artificially generated* map. The maps developed in V-REP and imported to MATLAB are called simulation maps. The real-world measurement maps are available datasets, which we have been downloaded from [9]. These maps represent the third floor common area of the MIT Stata Center (Dreyfoos Center) [9]. The artificially generated maps have been obtained by either hand drawing or randomly using MATLAB. The proposed approaches are next compared to the BCDC approach, and the results are presented in Table 2. The comparison shows that the ISOC algorithm is always slower than the BCDC algorithm, but most of the time it generates a shorter path. This confirms the results that have been

foreseen at the very beginning of the creation of the ISOC approaches and algorithm: ISOC deals with complex maps better, meaning that it generates shorter paths, but this is reflected in its increased computational complexity. The comparison was done on a computer with i7 3.7 Ghz processor and 16 GB RAM. Form point of computational complexity the three proposed solutions has the following characteristics. Solution "A" is the most time consuming, solution "B" is less time consuming but it uses more memory. Solution "A" and "B" always generates the same results. Solution "C" is usually the fastest but it doesn't guarantee the best orientation.

Map size (pixel)	Number of holes	Occupied ratio (%)	BCDC path length (mm)	ISOC path length (mm)	BCDC time (s)	ISOC time (s)	Image
435600	1	73.04	9984.41	9625.15	15.65	32.17	In the PhD theses
453696	3	55.77	21392.96	18449.71	10.79	45.65	
409600	5	60.06	27214.68	22278.05	14.82	96.97	
90000	2	39.23	2879.35	2880.04	10.54	49.48	

A Khepera III differential drive robot has been used in the experiments. The map has been previously known, and after the path commutation, an open loop control was applied for the robot. A map was evolved in the second phase and highlighted with yellow marker in order to visualize the real-world experiments as shown in Figure 5 (a). The robot path was measured with a camera applied above the robot path. This camera took photos in approximately equivalent periods of time. Figure 5 (b) shows a merge of several images that were taken during the experiments, and suggestively illustrates that the robot covers the path. The entire commented source code is available online.

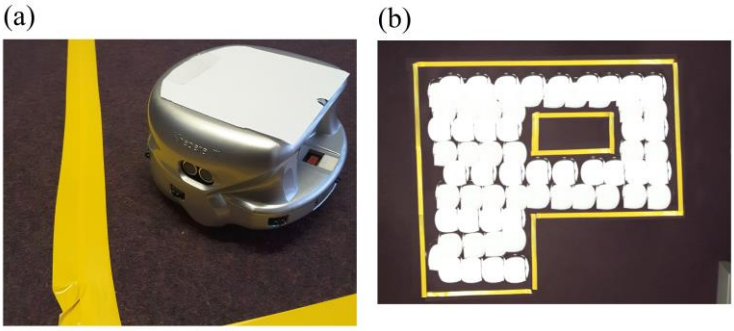


Figure 5 - Khepera robot (a) and merged images illustrating the robot path (b)

2.2. Thesis 2.

I created a kinematic trajectory following approach, called the multiple goal pursuit. One of the benefits of this algorithm is the more precise tunability in curved areas; and the more human-like reasoning involved. The human-like thinking comes from tracking and following more goal points on the road simultaneously. I investigated more versions of this algorithm; I also created the windowed version which divides the trajectory into more working sets and fits the curves to those sets.

References: [H26], [H27], [H28]

The motivation of the research behind this thesis was to construct a simple, but flexible kinematic trajectory following approach. In addition, I wanted to implement a *more*

human-like thinking, meaning to track and follow more goal points on the road simultaneously. As a driver we naturally take multiple considerations into account, without even considering it consciously. E.g. not only lane keeping is an important task but the cyclist in our lane also influences our planning. The proposed approach is encouraged by the pure-pursuit algorithm. Pure-pursuit algorithm is a popular trajectory tracking algorithm, widely used in mobile robotics and vehicular control for numerous reasons. The operation is simple and straightforward as it depends only on the kinematic model of the target mechanical system. The algorithm can be tuned by choosing look-ahead distance of points of the reference trajectory. In this thesis, I propose a low-computational complexity therefore fast trajectory following approach which was inspired by the pure-pursuit algorithm.

2.2.1 Description of multiple goal pursuit algorithm

The multiple goal pursuit is based on a simple to follow more than one selected goal point at once and to select the best fitting (see Figure 6). The realization is based on the finite number of discrete possible curves. These curves are implicitly determined by the wheel angle, which is constrained by the kinematic properties of the vehicle. In my realization first instead of using an optimizer I used a finite number of discrete possible curves and defined the best fit from this set of possibilities. Later I also implemented a simple optimizer for the task.

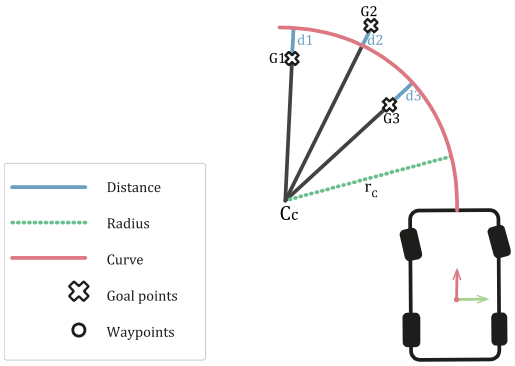


Figure 6 - High-level overview of multiple goal points-based pursuit algorithm

The algorithm extends Step 4 of the original pure-pursuit algorithm (described in the dissertation). Assume that the reference trajectory is given, with a set of T geometric points, and the P position and θ orientation of the vehicle. Define $G \subseteq T$ as the set of selected goal points, with a preset length $N = |G|$. Goal points are selected starting from the closest reference point to the vehicles current position (index denoted as j), shifted with an additional preset $o \in \mathbb{Z}$ offset:

$$G = \{G_1, G_2, \dots G_N \mid G_k, (j + o) \leq k < (j + o + N)\} \quad (14)$$

Assume an angle α_i from a domain of possible angles $[\alpha_{min}, \alpha_{max}]$ (presumably the wheel angle limits). A sequence of curves can be calculated each with radius $\rho_i = -2/\alpha$, and a centre point C_i. A G_k goal point from set G to any C_i determines a line segment: the normalized difference of the length of this segment and ρ_i radius is the $d \in \mathbb{R}^+$ distance from the curve. The metric for selecting a good angle is the sum of this difference for each

goal point. Summarizing the calculation of d_{sum} (equation 15) distance for the goal point set G:

$$d_{sum_i} = \sum_{k=0}^{N} \left| \left\| \overrightarrow{C_i G_k} \right\| - \rho_i \right| \quad (15)$$

Finally, the curve with the minimal d_{sum} is chosen. Figure 6 shows the steps of calculating the best angle given multiple selected goal points. This modification naturally introduces some issues. The most obvious one that at a given time there is no guaranteed curve that reaches all the selected multiple-goal points. Another example for issues is at the reaching the end of the reference trajectory, where there are physically no multiple goal to select. In the following I will go into the details why these issues emerge and how the algorithm overcomes these matters. One of the issues is that there is no definitive solution for the curve so a solution for that should be constructed. I created two different version for this particular problem. The first solution is a kind of greedy approach. In this case a discrete set of possible curves is created (similarly to Figure 7) and a simple loop iterates trough this set. The algorithm in this case is a simple minimum search, the minimal Euclidian distance between the given goal points and the curve determines the solution. On the other hand, this solution is rather a trade-off between the resolution of the curve set and the speed of the algorithm. The more precise result emerges the higher the resolution is, but of course this requires more computation and more time. For this reason, an optimization algorithm is created too. This algorithm iteratively approximates the best solution and will be described in the next section.

2.2.2 Description of windowed multiple goal algorithm

In this section a variant of the original multiple goal pure pursuit will be described, i.e. which is based on *windowing*. This algorithm is rather not a classic trajectory follower in terms of slightly redesigning, replanning the original trajectory. With this method a similar to the original trajectory is created, but the slight modification makes it more suitable to the vehicle or robot to follow. The design strategy consists of finding the optimum trajectory which approximates a desired trajectory defined by a set of chosen points. This means that the mentioned set is an initial data and a minimum problem must be defined.

From the multitude of possibilities, we will define the minimum problem starting with the following hypothesis:

1. During the approximation the steering angle γ is constant;
2. Changing the steering angle γ is an instantaneous process.

If the previous hypotheses are correlated, we can conclude that the initial set of points is approximated by a trajectory composed from circle arcs. The transition from one arc to another is instantaneous. For a particular circle arc a subset (named here *the working set*) is selected from the initial set.

The following decisions are subjects of debates:

- The number of points included in the working set;
- The number of common points which belong to different working sets;

- The car position when the next working set is defined.

Figure 7 illustrates the mentioned observations.

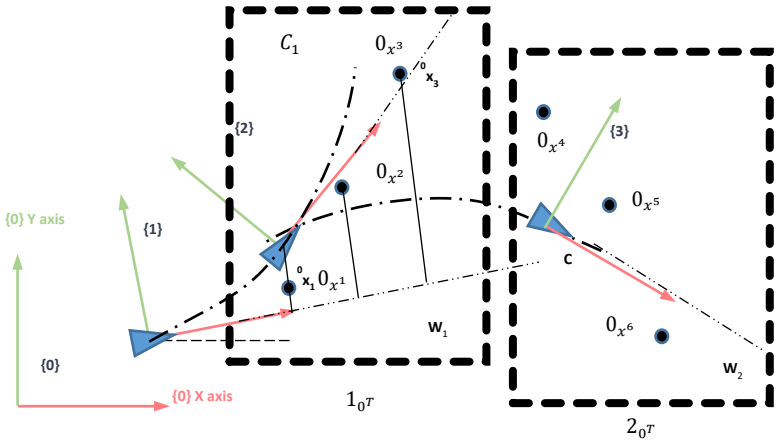


Figure 7 - The windowed goal pursuit strategy

In Figure 7. a possible strategy is illustrated. The figures contain two referential systems. The first 0 is the global (fix) referential system the second 1, 2, 3 is the mobile referential system attached to the vehicle. The blue triangle is the abstraction of the vehicle. Two windows are illustrated $w_{1,2}$. Each of them contains a *working set* of three points.

The trajectory is a composition of two circle arcs C_1 and C_2. The first (C_1) is defined in 1 using the window w_1 i.e. the points ${}^0x_1$, ${}^0x_2$, ${}^0x_3$. The second (C_2) is defined in 2 and use the window w_2 i.e. the points ${}^0x_4$, ${}^0x_5$, ${}^0x_6$. Each of the mentioned point is defined in the 0 referential frame.

Using m windows and n points for each window, the mathematical definition of the problem is the following:

$$R_j = \arg(min \sum_{i=1}^{m} e_i) \tag{16}$$

where: R_j is the radius of the arc C_j; e_i is the distance from point i to the arc C_j; j is the current window number $j = 1, \dots, m$; i is the current point number which belong to the widow j. The distance from point $\boldsymbol{x}_i$ to C_j it is also a problem worth to discuss. Figure 8. illustrates two possibilities: a.) considers only one coordinate of the current point. By contrary in Figure 8 b.) the shortest distance is considered.

For the first choice:

$$e_i = y_i - C_j(x_i) \tag{17}$$

where:

$$\begin{bmatrix} {}^jx_i \\ {}^jy_i \\ 1 \end{bmatrix} = {}^j_0T\,{}^0\mathbf{x_i} = \begin{bmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} {}^0\mathbf{x_i}; \tag{18}$$

θ is the orientation of the j referential system; x_j, y_j are the coordinate of the origin for j referential system. If we assume this definition, we will recognize that this is a quadratic regression problem which has the analytical solution.

$$\frac{1}{R} = \kappa = (\mathbf{X^t X})^{-1}(\mathbf{X^t Y}) \tag{19}$$

where:

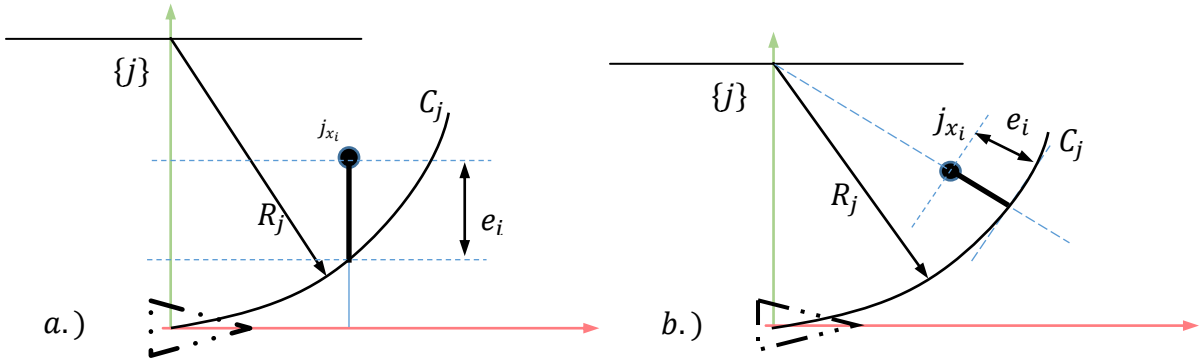


Figure 8 - Possible ways of computing the distances

$$\mathbf{X} = \begin{bmatrix} x_1^2 + y_1^2 \\ \dots \\ x_n^2 + y_n^2 \end{bmatrix} \qquad \mathbf{Y} = \begin{bmatrix} 2y_1 \\ \dots \\ 2y_n \end{bmatrix} \tag{20) (21}$$

For the second choice: $e_i = \left| R - \left\| \begin{bmatrix} x_i \\ y_i - R \end{bmatrix} \right\| \right| = \left| R - \sqrt{x_i^2 + (y_i - R)^2} \right|$ (22)

which has a numerical solution.

The algorithm uses the vehicle parameters, pose, the trajectory points and the input parameters as an input. The vehicle parameters are: b wheelbase and γ steering angle, and the pose. The points: ${}^{o}\boldsymbol{x_k}$. The algorithm parameters are: n working set, d skyline and κ curve limit. This limit decides whether the locomotion is based on linear trajectory or on circular trajectory window. In every iteration a new window selection is calculated.

The simulation uses the previous algorithm for a kinematic model which is in accordance with the starting hypothesis i.e. the steering is instantaneous. In the first simulations, the following kinematic bicycle model is used:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \frac{v}{L}\tan(\gamma) \\ \gamma = ct \end{cases} \tag{23}$$

To analyse the results, we define the lateral approximation error. The lateral deviation error of the current position $e(k)$ is the distance between this position and the current segment of the desired trajectory. The total deviation error e_T is the sum of the error during the all simulation from 1 to N. The relative deviation error e_r is the ratio between the total error and the total number of simulated points.

$$e(k) = distance\left(\boldsymbol{x}(k), \overline{{}^{0}\mathbf{x}_u\,{}^{0}\mathbf{x}_{u+1}}\right) = \left|\overline{\mathbf{x}(\mathrm{k})\mathbf{x}_{u+1}} - \mathbf{x}_u\widehat{\mathbf{x}_{u+1}}\left(\overline{\mathbf{x}(\mathrm{k})\mathbf{x}_{u+1}} \cdot \overline{\mathbf{x}_u\mathbf{x}_{u+1}}\right)\right| \tag{24}$$

$$e_T = \sum_{k=1}^{N} e(k) \tag{25}$$

$$e_r = \frac{e_T}{N} \tag{26}$$

These errors depend on number of the points include in the working set (n) named the skyline and on the position of the decision point (d). For the desired trajectory we have

made a study relative to this dependency and we obtained the results relative error value for each case (n, d).

2.2.3 Metrics

In order to evaluate these algorithms two metrics were elaborated: the sum of the lateral deviation, and maximum lateral deviation. To validate and investigate the performance of the proposed algorithms, error metrics must be defined. Two basic metrics were used, the average lateral deviation and the maximum lateral deviation. It can be assumed that the reference trajectory is composed of $n > 2$ reference points and is defined at any t time step. In this case, the lateral deviation is the perpendicular distance between a line l and the position P of the vehicle. Line l is determined by two L_1, L_2 points of the reference trajectory, a direction vector $\overrightarrow{L_U} = \overrightarrow{L_1 L_2}$. Using the general formula to calculate the perpendicular distance between point P and line l, the lateral deviation error is calculated as:

$$e_{lat}(P,(l)) = \frac{\|\overrightarrow{L_A P} \times \overrightarrow{L_u}\|}{\|\overrightarrow{L_u}\|} = \frac{\|\overrightarrow{L_2 L_1} \times \overrightarrow{PL_1}\|}{\|\overrightarrow{L_2 L_1}\|} \quad (27)$$

The algorithm can be validated with the average and the maximal lateral distance over a total of N_m measurement points (28). I used the arithmetic mean of measured lateral distances (assuming P_i is the vehicle position at a given time):

$$e_{avg} = \frac{\sum_{lat}^{Nm} D_{lat}^{i}(P_i,\{L1,L2\})}{|N_m|} \quad (28)$$

Over this set of measurements, calculating the maximum lateral deviation (29) is also straightforward:

$$e_{max} = max\{ D_{lat}^{i}(P,\{L1,L2\})\} \quad (29)$$

2.2.4 Test-environment used at real-world measurements

This section is devoted describing the automotive case study in which the implementation was used. The main task of this case study was to instrument an electric vehicle.with autonomous functionalities. The prototype of this system performed satisfactorily on the opening of the ZalaZone vehicle industry proving ground. The operational domain of this vehicle is limited to the proving ground and the university campus and is required to reach a relatively slow speed of 25 km/h. This vehicle is front-wheel driven, with a wheelbase of 2.7 meters and a track width of 1.77 meters. To ensure instantaneous localization of the vehicle we used a high accuracy RTK capable D-GPS sensor (KVH GEO-FOG 3D Dual), which was also used to capture reference trajectories.

2.2.5 Conclusion and evaluation

This thesis described a novel approach to improve the widely used pure-pursuit algorithm. The proposed algorithms are usable at low vehicle speed with a reference trajectory generated by planner components or with an explicit reference trajectory (e.g. GNSS-based waypoints). All resources and descriptions needed to reproduce the algorithm are available at our organization's public repository. In PhD theses the the

validation results and comparison with other versions of pure pursuit is summarized, here it is not detailed due size restrictions. Two tracks were selected: a segment of the ZalaZone proving ground, and a driving school test track near our university's campus. We used three different methods to validate the implementation. The first two methods are simulation-based testing of the algorithm: first is a simple simulation with simple graphical implementation. This simulation is capable of 2D visual feedback, alongside monitoring lateral deviation from the preset trajectory. This Python-based implementation is shared at our organization's GitHub repository. In the next step, we implemented this algorithm as a Robot Operating System (ROS) node. We embedded this node into a simulated version of our testing car. The simulation was based on OSRF Gazebo, which allowed us the versatile software integration and to test with dynamic properties involved. This implementation is currently available at private repositories and might be released to the public in the future. The implementation is ROS-based, we could test this software both in simulation on our testing vehicle, without significant modification in the code. Simulation-based execution proved no significant deviation from the simulated tests. A working implementation of the presented algorithm is shared at our organization's public GitHub repository.

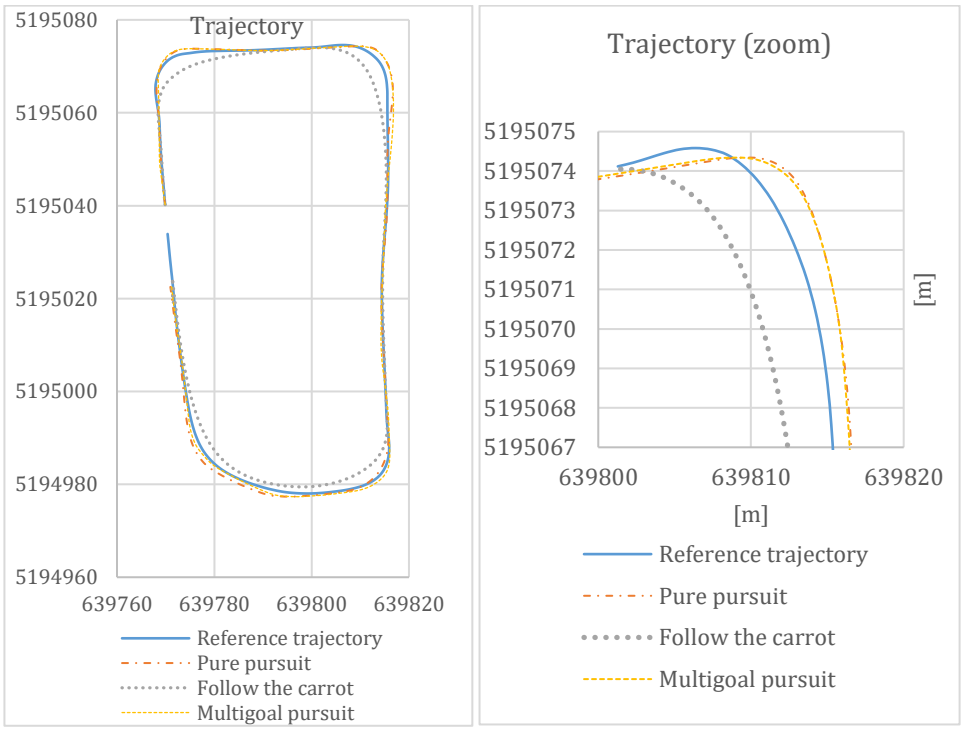


Figure 9 - Typical behaviors of the examined approaches

As another experiment we instrumented an electric vehicle (Nissan Leaf) with autonomous functionalities. The operational domain of this vehicle is limited to the ZalaZone vehicle industry proving ground and the university campus and is required to reach a relatively slow speed of 25 km/h. This vehicle is front wheel driven, with a wheelbase of 2.7 meters and a track width of 1.77 meters. To ensure instantaneous

localization of the vehicle we used a high accuracy RTK capable GPSs. One of them was the KVH GEO-FOG 3D the other was Swift Navigation Duro Inertial RTK. These sensors were also used to capture reference trajectories. After the algorithm showed satisfying behaviour in the simulation environment, we decided to test it in a real-world scenario. During the experiment we used the Nvidia Jetson TX2 in the Nissan Leaf. The Nvidia Jetson TX2 is a 7.5-watt embedded controller where we used Ubuntu 18.04 (Bionic) along with ROS Melodic Morenia. The embedded controller has 8 GB of memory and 59.7 GB/s of memory bandwidth, NVIDIA Denver2 and quad-core ARM Cortex-A57 CPU, and an integrated 256-core Pascal GPU. During the experiments we only used the the KVH GEO-FOG 3D sensor as the source of localization we provided the wheel angle reference signal alongside with the speed reference via CAN and we measured the wheel angle and speed back in the same protocol. We obtained that the multigoal pursuit is able to drive the vehicle with the mentioned embedded controller at slow speeds, so about 25 km/h. The algorithm is able to perform smooth and continuous movement along a predefined trajectory, even manoeuvring around corners and edges. The main advantage of the proposed algorithm is the human-like reasoning involved during the trajectory following.

2.3. Thesis 3.

I showed that probabilistic occupancy grid map construction is possible to be done via the "crisp signatures" sensory model. The novelty of this approach consists of involving the concept of signatures in map construction and the elimination of Bayesian sensory, so the probabilistic characteristics of the environment model could be involved in the mapping phase. This approach is considered as another mathematical sight of sensory model realization and works well especially with time-of-flight-based sensors such as a LIDAR. The proposed approaches are validated by simulation and experimental results, the main benefits of the proposed approach consist of the signature-structured handling of complex sensor data and the algorithmic separation of the probabilistic characteristics.
References: [H1], [H3], [H12], [H14], [H23], [H24], [H25]
Nowadays the low-level tasks of self-driving - or other in words autonomous - vehicles and robots are intensively researched. These low-level tasks include for example the path-planning and obstacle avoidance task. Both path planning and obstacle avoidance usually relies on map building. In the following the occupancy grid based probabilistic map building will be examined. This means that the localization algorithm builds the map in probabilistic manner. In 1998 Thrun, Burgard and Fox proved, that the probabilistic approach can be used as a method involved in the SLAM (Simultaneous Localization And Mapping) problem [10]. Various sensory models are calculating the probability in different ways. In the occupancy grid map every grid cell is represented as probability, most often a number between 0 and 1. Several methods such as Bayesian, Dempster-Shafer, Fuzzy, Borderstein or TBF sensor representation [11] can be mentioned. The proposed solution originality consists of two things. The first is defining a probabilistic function for each element of the grid, the second is involving the concept of signatures in

the process. The proposed sensory model is called as the *crisp* model. Here the discrete 2D measurement was taken and the 3rd dimension comes from occupied feature of the given coordinate. On the figures the numbers close to or equal to 1 means that the given coordinate (grid cell) is occupied. Accordingly, numbers close to or equal to 0 means that the coordinate is empty. All other areas are initialized with 0.5 because there is no prior knowledge about this grid cells. On the visualization of the crisp sensory model is visible that there are no values other than 0, 0.5 and 1. According to Sebastian Thrun [12] LIDAR range finders do not always require the probabilistic – for example Bayesian – sensory model for a single measurement.

2.3.1 Application of signatures

The classical, Bayesian sensory model does not involve signatures. The signatures are flexible mathematical constructions which can contain multiple measurements of a single-channel LIDAR (2D LIDAR) or even multiple measurements of a multiple-channel LIDAR (3D LIDAR). Considered $A_S(x)$ as signature, which corresponds to a single-channel LIDAR measurement.

$$A_S(x) = \begin{bmatrix} \begin{bmatrix} a_1 \\ \begin{bmatrix} a_{1,1} \\ a_{1,2} \\ a_{1,3} \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} a_2 \\ \begin{bmatrix} a_{2,1} \\ a_{2,2} \\ a_{2,3} \end{bmatrix} \end{bmatrix} \\ \cdots \\ \begin{bmatrix} a_n \\ \begin{bmatrix} a_{n,1} \\ a_{n,2} \\ a_{n,3} \end{bmatrix} \end{bmatrix} \\ p \end{bmatrix} \tag{30}$$

Where at given m measurement a_m tells if the cell is occupied, $a_{m,1}$ is the ID, $a_{m,2}$ is the distance $a_{m,3}$ angle, and p is an optional parameter for the pose of the sensor. This gives us the possibility to easily represent the LIDAR measurement. The other benefits of this approach is that it can be easily extended to a multiple-channel LIDAR too and the available signature operations can be applied too. For example, if a resolution is changed or another range sensor is applied, the extension and contraction operation is available. The generalization of the range sensor can be viewed as an $A_S(x)$ signature described in equation (30).

$$A_M(x) = \begin{bmatrix} A_{S_1} \\ A_{S_2} \\ \dots \\ A_{S_c} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} a_{1,1} \\ \begin{bmatrix} a_{1,1,1} \\ a_{1,1,2} \\ a_{1,1,3} \end{bmatrix} \end{bmatrix} \begin{bmatrix} a_{2,1} \\ \begin{bmatrix} a_{2,1,1} \\ a_{2,1,2} \\ a_{2,1,3} \end{bmatrix} \end{bmatrix} & \dots & \begin{bmatrix} a_{c,1} \\ \begin{bmatrix} a_{c,1,1} \\ a_{c,1,2} \\ a_{c,1,3} \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} a_{1,2} \\ \begin{bmatrix} a_{1,2,1} \\ a_{1,2,2} \\ a_{1,2,3} \end{bmatrix} \end{bmatrix} \begin{bmatrix} a_{2,2} \\ \begin{bmatrix} a_{2,2,1} \\ a_{2,2,2} \\ a_{2,2,3} \end{bmatrix} \end{bmatrix} & \dots & \begin{bmatrix} a_{c,2} \\ \begin{bmatrix} a_{c,2,1} \\ a_{c,2,2} \\ a_{c,2,3} \end{bmatrix} \end{bmatrix} \\ \dots \quad \dots & & \dots \\ \begin{bmatrix} a_{1,n} \\ \begin{bmatrix} a_{1,n,1} \\ a_{1,n,2} \\ a_{1,n,3} \end{bmatrix} \end{bmatrix} \begin{bmatrix} a_{2,n} \\ \begin{bmatrix} a_{2,n,1} \\ a_{2,n,2} \\ a_{2,n,3} \end{bmatrix} \end{bmatrix} & \dots & \begin{bmatrix} a_{c,n} \\ \begin{bmatrix} a_{c,n,1} \\ a_{c,n,2} \\ a_{c,n,3} \end{bmatrix} \end{bmatrix} \end{bmatrix}^T \tag{31}$$

The signature consists of multiple A_S signatures: $A_M(x) = [A_{S_1} \quad A_{S_2} \quad \dots \quad A_{S_c}]^T$, where C is the number of channels. In the thesis only this approach is examined, but note that one of the benefits of this approach is its flexibility to extend and write mathematical operations which can adapt to this flexibility.

2.3.2 Mapping

Map construction is a process that intends to spatially represent the objects in a particular space. At the limit, not in the objects themselves is interesting, but particularly in the obstacles, that is, those locations that cannot be accessed by the robot. These locations are called occupied. It is already understood that the locations mentioned are defined by a position vector, in an absolute reference system. The obstacles discovery is done by a sensory system mounted on a mobile robot or vehicle that moves in the absolute reference system. This means - that finally the measured position of obstacle must be transformed from the sensor referential frame in the absolute referential frame. This is not a trivial problem because of the following. The pose (position and orientation) of the robot is modelled in a stochastic way. The sensor measurements are also influenced by noise, so once again probabilities are to be considered. The obstacle position is assimilated with a grid cell occupancy. A particular location: can be subject of different measurements whose results are necessary to be correlated. The problem which is intended to solve is to construct an original mathematical model of this process. The originality of the approach consists on computational simplification. This simplification can be done because of the following: The original sensory model was developed with respect to the early, noisy sensors (e.g. sonars). For example, a LIDAR produces more reliable measurement, even the measurement noise is still involved. The measurements are more frequent, a grid cell is evaluated many times. The probabilistic computation can be involved into the map construction in update phase.

From [12] the location belongs to the Bayesian localization class, and can be described with the following equations which encompass a two phases: prediction and correction of the actual estimate. This can be formalized as follows.

$\{bel(x_{t-1}), u_t, z_t, m\}$
for _all_ x_t _do

$$\begin{cases} \overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m) \cdot bel(x_{t-1}) \\ bel(x_t) = \eta\, p(z_t \mid x_t, m) \cdot \overline{bel}(x_t) \end{cases} \tag{32}$$

endfor
$\{bel(x_t)\}$
where: x_t, u_t, m, z_t are the system state, the command signal, the map and the measurements. The prediction takes account of the model and the correction of the measurements. Firstly, an estimated prediction is established $\overline{bel}(x_t)$. In the second part the estimated prediction is corrected using the measurement model $p(z_t \mid x_t, m)$;
For the localization step the following model is proposed:

$\{bel(\boldsymbol{x}), u_t, \boldsymbol{z}_t, m\}$
for _all_ $\boldsymbol{x}$ _do

$$\begin{cases} \bar{\sigma}_{m,t} = \sigma_{m,t-1}\sigma_m \\ \overline{bel}(\boldsymbol{x}) = \bar{\sigma}_{m,t}\delta(\boldsymbol{x}, \boldsymbol{x}_t) \\ m(\boldsymbol{x}) = \sigma_s \delta(\boldsymbol{x}, \boldsymbol{z}_t) \\ \sigma_{m,t} = (\bar{\sigma}_m + \sigma_s - \bar{\sigma}_m\sigma_s) \\ bel(\boldsymbol{x}) = \sigma_{m,t}\delta\left(\boldsymbol{x}, \frac{\bar{\sigma}_{m,t}\boldsymbol{x}_t + \sigma_s \boldsymbol{z}_t}{\bar{\sigma}_{m,t} + \sigma_s}\right) \end{cases} \tag{33}$$

endfor
$\{bel(\boldsymbol{x})\}$
where:
$\boldsymbol{x}$ is a state variable, the mobile robot pose;

$$\delta(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 \;\; if \sum_{i=1}^{n} |x_i - y_i| = 0 \\ 0 \;\; if \sum_{i=1}^{n} |x_i - y_i| \neq 0 \end{cases}$$

$x_t = f(x_{t-1}, u)$ is the mobile robot model;
$\sigma_{m,t}$ is the degree of trust of the robot pose at moment t;
$\bar{\sigma}_{m,t}$ is the estimated degree of trust of the robot pose at moment t
σ_m is the degree of trust of the model;
σ_{st} is the degree of trust of the sensor measuring;
For the corrected degree of trust $\sigma_{m,t} = (\bar{\sigma}_m + \sigma_s - \bar{\sigma}_m\sigma_s)$ the condition is imposed as $\begin{cases} 1 \geq \sigma_{m,t} \geq \bar{\sigma}_m \\ 1 \geq \sigma_{m,t} \geq \sigma_s \end{cases}$.
For the obstacle measurements (identification), the following model is proposed:

$$\begin{cases} \sigma\left({}^1\boldsymbol{x}_{i,t}\right) = \sigma^0 \delta({}^1\boldsymbol{x}_{i,t}, {}^1\boldsymbol{z}_{t,i}) \\ \sigma_t^0 = \sigma_{m,t}\sigma_s \end{cases} \tag{34}$$

where: ${}^1\boldsymbol{z}_{i,t} = \begin{bmatrix} l_i \cos(\alpha_i) \\ l_i \sin(\alpha_i) \end{bmatrix}$; ${}^1\boldsymbol{x}_{i,t} = \lambda\, {}^1\boldsymbol{z}_{i,t}$

${}^1\boldsymbol{x}_{i,t}$, ${}^1\boldsymbol{z}_{i,t}$ are vectors which belongs to the i^{th} measurement and are defined in the robot referential system, $\lambda \in [0,1]$; i refers to the number of the measurement.
This step is represented with the robot pose degree of trust is $\sigma_m$and the occupancy degree of trust is σ_{oi}. The measurements results must to be transformed from the robot referential system into the absolute referential system:

$$\begin{cases} \boldsymbol{x}_{i,t} = \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) & r_{tx} \\ \sin(\theta_t) & \cos(\theta_t) & r_{ty} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_{t,i} \\ 1 \end{bmatrix} \\ \bar{\sigma}(\boldsymbol{x}_{i,t}) = \sigma({}^1\boldsymbol{x}_{i,t}) \end{cases} \tag{35}$$

where: r_{tx}, r_{ty} and θ_t are the position and the orientation of the mobile robot.
The map update is can be solved e.g. by Bayesian the formula:

$$p(h|o) = \frac{p(h)p(o|h)}{\sum_{i=2}^{2} p(h_i)p(o|h_i)} \tag{36}$$

where: h is the hypothesis that the location x is occupied; o is here the observation (the values of z); in fact, there are 2 hypotheses: x is occupied and x is free.
For the map update, the following model is proposed:

$$\begin{cases} \sigma(\boldsymbol{x}_{i,t}) = \frac{1}{2}\left(\tanh\left(\frac{s}{2} - \frac{1}{2}\right) + 1\right) \\ s = \begin{cases} \sigma(\boldsymbol{x}_{i,t-1}) + 1 & if \quad \sigma(\boldsymbol{x}_{i,t}) \neq 0 \\ \sigma(\boldsymbol{x}_{i,t-1}) - 1 & if \quad \sigma(\boldsymbol{x}_{i,t}) = 0 \end{cases} \end{cases} \tag{37}$$

2.3.3 The crisp sensory model with signatures

Probabilistic map building is necessary because of the localization inaccuracies. These characteristics will be introduced in the map building phase. An element - a cell - of the map can be associated either with an obstacle or an empty area i.e. two hypothesis can be defined for a cell: h_o *the cell is occupied* or h_e *the cell is free*. The mathematical representation of these hypotheses is probabilistic: $p(h_o)$ respectively $p(h_e)$. The correlation between these probabilities is modelled by equation (38).

$$p(h_o) + p(h_e) = 1 \tag{38}$$

The proposed algorithm stores only the occupied grids. As a result we obtained a i by j matrix or a 2D array $L_w(i,j)$, which will be named the wall layer. Because of (1) is no need to store the empty grids. As an initial step the wall layer has a uniform probability of $p(h_o) = p(h_e) = 0.5$. After each measurement elements of $L_w(i,j)$ are recalculated according to equation (39).

$$\begin{aligned} p(h_o|o) &\propto p(h_o) \cdot p(o|h_o) \\ p(h_e|o) &\propto p(h_e) \cdot p(o|h_e) \end{aligned} \tag{39}$$

Using Bayes' theorem the probabilities are normalized (40) .

$$p(h_o|o) = \frac{p(h_o) \cdot p(o|h_o)}{p(h_o) \cdot p(o|h_o) + p(h_e) \cdot p(o|h_e)} \tag{40}$$

$$p(h_e|o) = \frac{p(h_e) \cdot p(o|h_e)}{p(h_o) \cdot p(o|h_o) + p(h_e) \cdot p(o|h_e)}$$

I have used a convention accordingly the highest probability is red and the lowest probability is blue. On only two independent measurements are visualized for the sake of perspicuity, even the algorithm uses an iterative process with several measurements where each measurement is used for map improvement. In order to prove my concept, an experiment was carried out, where we can demonstrate that the approach is working. We have chosen a mapping with known poses [13] scenario. In the experiment we have chosen a Gazebo as the simulator, and we created a single-channel LIDAR with 270° field of view. The sensor has a 1° resolution, so a measurement consists of a signature where the 270 measurements and the pose are represented.

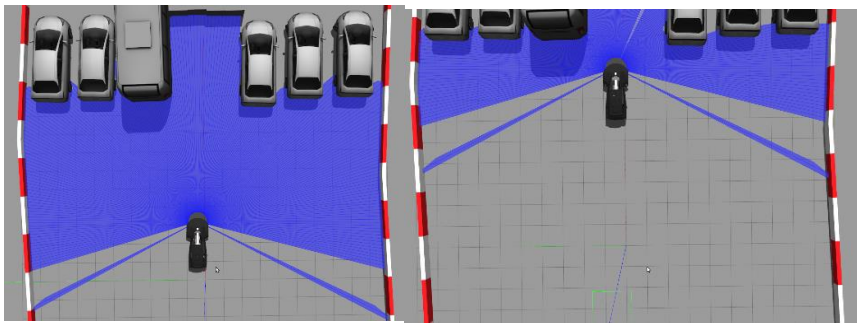


Figure 10 - The scenario to prove the concept

In the discussed scenario a simple parking manoeuvre is carried out, but the focus was not on the manoeuvre itself, rather on the map building capabilities with signatures. The grid map is chosen to be 120x120 cells, this is a reasonable resolution given the overall scenario size and the objects. The pose and the range measurements are accessible from Gazebo, we wrote a simple script in order to visualize the mapping.

2.3.4 Conclusions

The proposed crisp signatures sensory model involves signatures as a mathematical tool to structure the sensory data better and also lets the probabilistic characteristics of the environment model involved in the mapping phase. This approach is considered as another mathematical sight of sensory model realization. A showed the sensory model works well especially with time-of-flight-based sensors such as a LIDAR. The main benefits of the proposed approach consist of the algorithmic separation of the probabilistic characteristics, the signature-structured handling of complex sensor data and thus the capability of extension and transfer.

2.4. Thesis 4.

I showed that in narrow subdomain of perception for experimental vehicles a VGG-VD neural network trained on low annotation categories with transfer learning case RMSprop converges slightly faster than Adam optimizer; despite the known fact, that that Adam

generally overperforms RMSprop. To prove this phenomenon the results, dataset and necessary conditions have been shared.
References: [H16], [H20], [H21]

As nowadays necessities are arising in the fields of sound and image recognition, self-driving decision making, data processing and understanding, the more attention the neural network (NN) and deep learning (DL) solution gets. From the point of view of my scientific work the subdomain of perception and for mobile robots and road vehicles are particularly important. Instead of examining general neural network disciplines this thesis focuses on a narrow subdomain of this field. In the following the few-annotation category, segmentation-based, perception problems are targeted. Perception in the domain of vehicles or in robotics as general is similar to the conventional computer vision domain tasks [14]. It must be emphatized that this is a similarity not identity, the techical and techological approaches are smimilar but the objective is different [15]. Here, the fundamental problems are called sensing and perception. Sensing is defined as a data acquisition and transfer from a sensor (e.g. camera) whereas perception is more like a multiform process. Basically the understanding or the analysis of the data is done there. The research targets the perception where the features are extracted from the sensors data after the acquisition. Compared to the classical robotics problems there are less tasks involved (e.g. there is no grasping or assistant systems) but the objects which are involved are much more diverse. In the classical computer vision (CV) domain, which are often present in robotics conventional algorithms extract information from the raw sensor data based on artificially designed features (pl. SIFT [16], SURF [17], ORB [18], pattern recognition [19], template matching [19]). In contrast of this approach the NN and DL solutions [20], [21] use fundamentally different principle regarding these and other [22] problems. Nowadays there is clearly a trend which supports the solution of complex approximation problems with growing computational powered hardware (GPU, CPU, TPU) and multi layered networks such as convolutional neural networks. The thesis focuses on neural network-based solutions where the problem characteristics defines few annotation categories. The dataset consists of general traffic, race and model vehicle images.

Figure 11 - Typical images from the Shell Eco-marathon competition

Because of viewpoint variation, scale variation, intra-class variation segmentation is considered as a challenging task [23]. The segmentation problem is intensively-researched, and nowadays neural network architectures are solving it with the most success [24] [25]. Many of them rely on a special instance of convolutional neural networks (CNN), the fully convolutional neural networks (FCN). To solve a segmentation problem properly you need to make a lot of choices. These choices involve defining the NN architecture, including the number of layers the relation between them, applying a good activation function, choosing a proper hardware, and execute whole training process with a suitable optimization technique. The next sections tend to help in these choices.

2.4.1 Gradient-based optimization techniques

All modern approach of neural network optimization takes advantage of the fact that all operations used in the training are differentiable. Differentiable in this context means that it "can be derived" and the derivative of a tensor operation is called the gradient (∇). The gradient is computable as the loss with respect to the network's coefficients. To decrease the loss the optimization needs to move in the opposite direction from the gradient. Let's consider a continuous function; here a small change in the input can only result in a small change in the output. That can be surely assumed from the definition of continuity. Because of fact that the function is smooth – or in other words it doesn't have any rapid curves - when small enough epsilon (ε) is changed it's possible to approximate the change in the output linear function of slope regarding epsilon. The slope is called the derivative of the function in the chosen point. The generalization of the concept of derivatives to multidimensional function's inputs are functions that take tensors as inputs.

The state-of-the-art optimizers nowadays are: Adagrad, Adam, Proximal Gradient Descent and RMSprop. [26] Each optimizer belongs to the gradient descent optimizer family. The gradient is denoted (with the ∇ Nabla symbol) respect to the function $f(\theta)$ and v vector at each point θ is the directional derivative of f along v:

$$D_v f(\theta) = \nabla f(\theta) \cdot v \tag{41}$$

Each method is a variant of the vanilla gradient descent, where θ are the parameters, η is the learning rate, $J(\theta)$ objective function to the entire training set:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta_t) \tag{42}$$

The first optimizer examined for my case is Adagrad which involves stochastic optimization techniques and online learning which employ proximal functions to control the gradient steps of the algorithm [26] [27]. Adagrad applies different learning rate for every parameter θ_i at every time step t, where $g_{t,i}$ is the gradient of the objective function [26]:

$$\theta_{t+1,i} = \theta_{t,i} - \eta \cdot g_{t,i} \tag{43}$$

Proximal Gradient Descent is another gradient descent method. The algorithm alternates between two phases: [28] the first is performing unconstrained gradient descent and the second is solving an instantaneous optimization. This problem trades off minimization of a regularization term while keeping close proximity to the result of the first phase.

The last two optimizers are discussed together because they have similar approach. The advantage of Adam is that it does not require a stationary objective [29]. Adam works with sparse gradients, and it naturally performs a form of step size annealing. According to the literature [29] Adam slightly outperform RMSprop in general problems but in this case it's the opposite. It is true that Adam and RMSprop are similar in the means of performance but according to our experiments RMSprop is the one who performs slightly better, see figure 6. RMSprop is proposed by Tieleman and Hinton [30] note that this is not a classic publication rather a lecture. In case of RMSprop the equation also involves the he running average $E[g^2]$ [26]:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]t+\varepsilon}} \cdot g_t \tag{44}$$

From the mentioned optimizers RMSprop and Adam were the bests in our case. The heavy fluctuation in figure 6 is due the logarithmic scale, but this enlarges the differences between the loss curves. Choosing a proper learning rate can have fortunate consequences to the training progress.

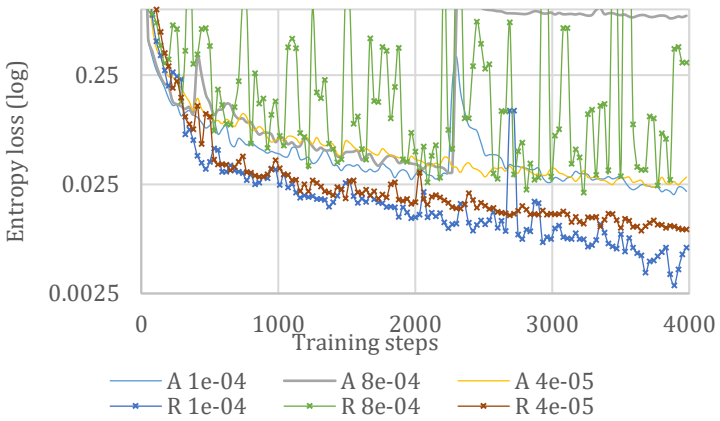


Figure 12 - Adam and RMSprop optimization algorithms (A – Adam, R – RMSprop) each with 3 different learning rates (10^{-4}, 8×10^{-4}, 4×10^{-5})

High learning rate can have two kinds of effects regarding the entropy loss. The first is slow convergence or not even reaching the desired minimum, the second is fluctuating loss. However according to our experiences this undesirable fluctuation may lead the loss to a different maybe even better local minimum. This behaviour is of course not easily reproducible; it has random characteristics. In extreme cases very high learning rate can even cause divergence.

2.4.2 Neural-network architectures

FCN in this case means that unlike a simple CNN, FCN only contains convolutional layers. FCN architectures works on feature extraction principle. Every layer is responsible for extracting different features such as colors, contours, edges, see figure 2. One of the well-known networks are the BVLC FCN [31] and their variables (FCN32-s, FCN16-s, FCN8-s), SegNet [32].

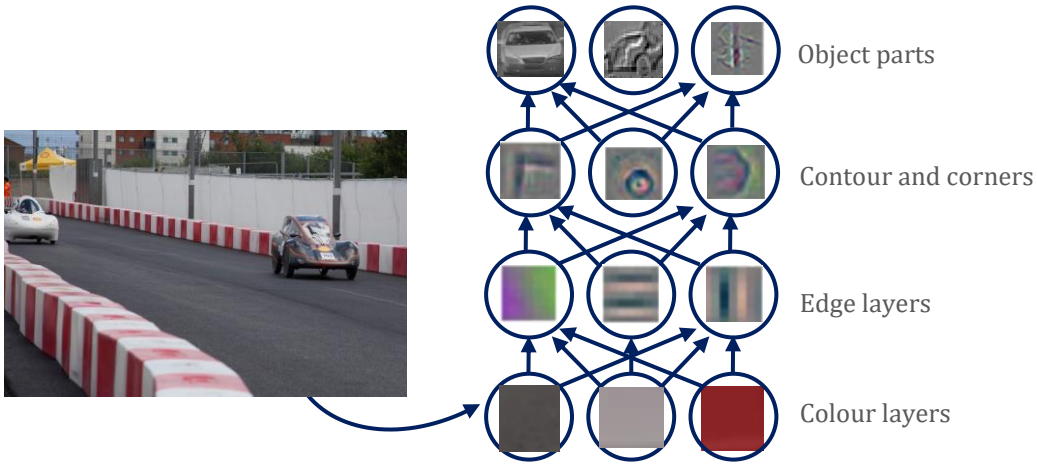


Figure 13 - The visualization of the neural network's logical layers typical image segmentation

Also there are possibilities to use CNN's which were made for classification, but extend them and use as FCN, and transfer their learned representations by fine-tuning. The example for this approach is AlexNet-FCN [31], DeepLab-FCN-CRF [33], the VGG-FCN [31]. Due their simple, but very deep network architecture and high performance the variations of the VGG-FCN networks were chosen to be examined further.

Due to the numerous parameters involved (e.g. VGG-VD19 has 156,351,488 and VGG-VD16 has 138,357,480 trainable parameters) training a convolutional network can be a highly time-consuming process. The training and application of a NN consists typically of various matrix operations such as matrix multiplication or addition, which are parallelizable operations. Since of this phenomenon GPUs (graphical processing units) can overperfom CPUs (central processing units) because GPUs can handle lot of parallel calculations using thousands of cores. This approach when performing computations instead the traditional CPU on GPU is called GPGPU (General Purpose Computing on Graphics Processing Units). If a GPU is needed to be chosen for NN training and application the following qualities of the GPU needs considered. First the processing power, which means how fast a GPU can work with the NN parameters. This can be approximated as number of CUDA cores multiplied by the clock speed of each core. The second is the memory bandwidth, which is the speed of the memory. It is measured in gigabytes per second (GB/s). The third is the memory size, which is in gigabytes (GB) and determines how big batch size you can choose. If you have small memory try to train only a small batch size in order to fit the training model into memory but this will slow down the learning. As a rule of thumb for a typical convolutional neural network 8 GB is the minimum recommended. In my PhD theses there are various CPUs and GPUs are compared according how much training step can achieve with 224x224 images. In this booklet this data is not present due size limitations. Possible benchmarks also could contain accuracy, memory consumption, parallel scaling, but the most important is the performance which can be measured in training step / hour. In practice you either train a

network from the beginning, from scratch or initialize your weights as random or - more commonly - you begin with a pretrained network and train that for your special purpose. This method is called transfer learning.
We have chosen transfer learning because all the state-of-the-art networks provides their pretrained weights. The main deep learning frameworks such as Caffe, MXNet, TensorFlow, PyTorch, Keras, Theano, CNTK and so on provides pretrained models. There are lot of openly available datasets which were trained for hundred or thousand iterations on benchmark datasets such as on PASCAL VOC or on CIFAR or on BSDS. At this experiment we used TensorFlow as or machine learning framework. Of course, I considered other frameworks too which may more beneficial from certain point of view.

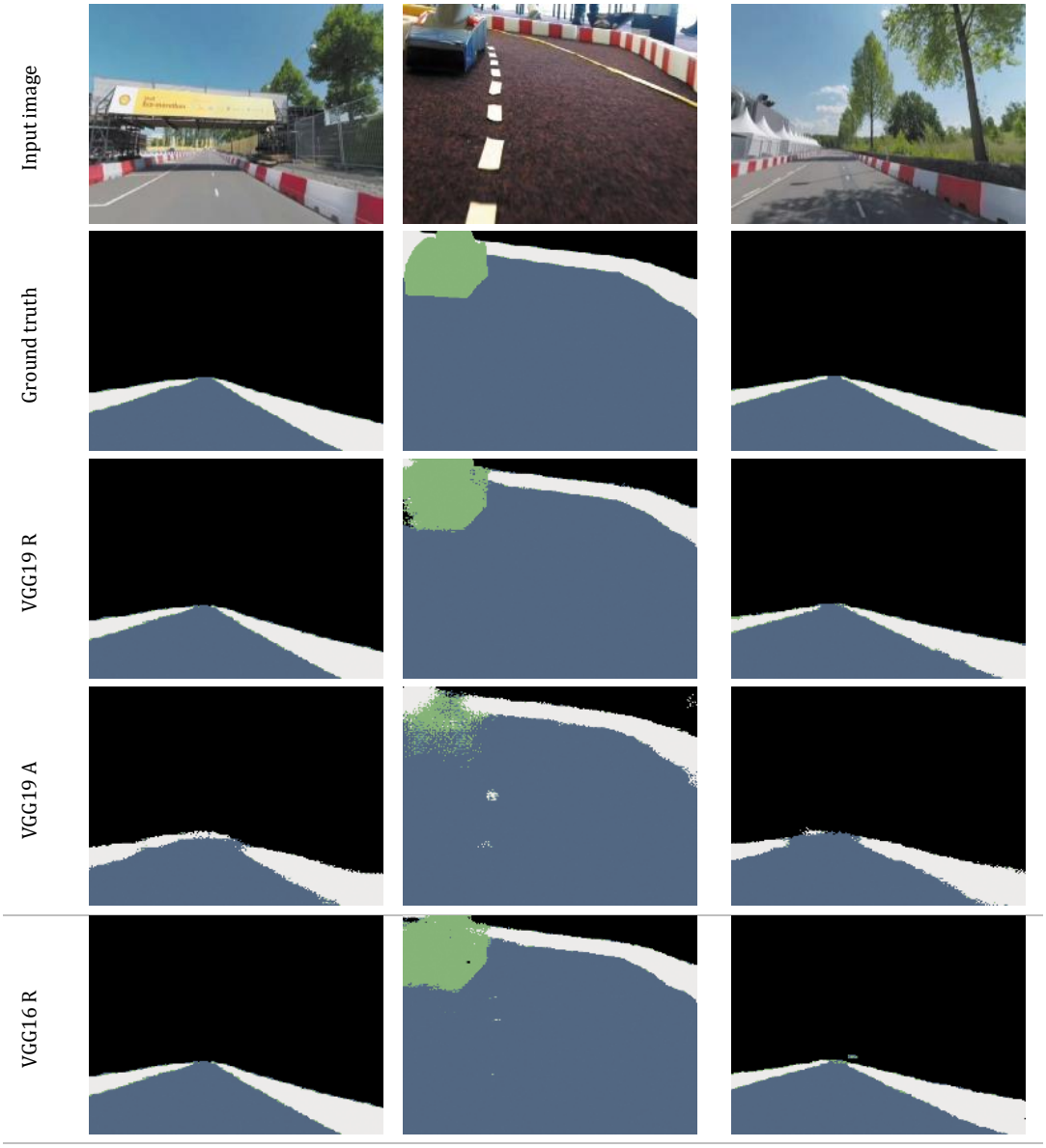


Figure 14 - Training results on different architectures (VGG-16-FCN, VGG-19-FCN) with different optimization techniques (A - Adam and R - RMSprop)

According to the observations choosing a right optimization method can shorten the training process. On figure 8 there are some results visible on different architectures with different optimization techniques. The training was done with the same architecture and with the same training step. It is visible that the 16 layered VGG-FCN with RMSprop has better results as the 19 layered version with Adam. Usually the 19 layered version has better results, but if not the best optimization technique was chosen the even the 16 layered can be better, at least at the same training steps.

2.4.3 Conclusions

The goal of our work was to test different solutions in the narrow subdomain of perception for experimental vehicles. After that by collecting and analysing results we phrased recommendations for this subdomain. We recommend the following workflow. It is advised to use ReLU or leaky ReLU as activation function for FCNs. As a matter of optimization technique RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances, but in this case RMSprop is a slightly better choice despite the fact that Adam generally overperforms RMSprop.

3. Own publications

[H1] C Pozna, E Horváth, J Kovács: *Developing rapid prototype-capable applications for industrial mobile robot platforms* IEEE, 18th International Conference on Intelligent Engineering (INES) Tihany, Magyarország, 2014

[H2] E Horváth, J Hollósi: *Kutatási célú gyorsprototípus alkalmazások fejlesztése Neobotix MP500 mobilrobot rendszerre* EMT XIII. OGÉT, Nemzetközi Gépészeti Találkozó Şumuleu Ciuc, Csíksomlyó, Románia, 2015

[H3] C Pozna, E Horváth, J Kovács: *An abstraction of the Lidar measurements* 10th Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI) Timişoara, Temesvár, Románia, 2015

[H4] E Horváth, P Kőrös, I Lakatos, P Dely: *Development of individual information technology systems of experimental vehicles* 10th Jubilee IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI) Timişoara, Temesvár, Románia, 2015

[H8] C Pozna, E Horváth, J Hollósi: *The inverse kinematics problem, a heuristical approach* IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI) Herl'any, Szlovákia, 2016

[H9] E Horváth, C Pozna, C Hajdu, J Hollósi: *A use case of the simulation-based approach to mobile robot algorithm development* IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI) Herl'any, Slovakia, 2016

[H10] E Horváth, P Kőrös, I Lakatos: *A Case Study of Software Developments in Electric Experimental Vehicles* Scientific Bulletin of the "Politehnica" University of Timisoara, Transactions on Automatic Control and Computer Science, ISSN 1224-600X Timişoara, Temesvár, Románia, 2015

[H11] C Pozna, E Horváth: *Predictive Distributed Formation Control for Swarm Robots Using Mobile Agents* Scientific Bulletin of the "Politehnica" University of Timisoara, Transactions on Automatic Control and Computer Science, ISSN 1224-600X Timişoara, Temesvár, Románia, 2016

[H12] E Horváth, C Pozna, H Somogyi: *Járműveken és mobil robotokon alkalmazható "foglaltsági rács" alapú térképépítési eljárás* EMT XIV. OGÉT, Nemzetközi Gépészeti Találkozó Deva, Déva, Romania, 2016

[H13] H Somogyi, E Horváth: *Autonóm járművek navigációs rendszerének minimális költségű szenzorokkal való megvalósíthatóságának elemzése* EMT XIV. OGÉT, Nemzetközi Gépészeti Találkozó Deva, Déva, Romania, 2016

[H14] E Horváth, C Pozna: *Probabilistic Occupancy Grid Map Building for Neobotix MP500 Robot* IEEE WPNC'16 IEEE 13th Workshop on Positioning, Navigation and Communications Bréma, Bremen, Germany, 2016

[H15] E Horváth, P Kőrös: *Alacsony energiafelhasználású villamos hajtású prototípus jármű identifikációs mérései és fejlesztési kérdései* EMT XXV. OGÉT, Nemzetközi Gépészeti Találkozó , 243-246. Kolozsvár, Románia, 2017

[H16] E Horváth, C Pozna, Á Ballagi: *Road Recognition using Fully Convolutional Neural Networks* 3rd International Conference for Doctoral Students - IPC, Braşov, Romania, 2017

[H17] E Horváth, P Kőrös, I Lakatos, F Szauter: *Two Operating States-Based Low Energy Consumption Vehicle Control* 13th ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications Cleveland, Ohio, USA, 2017

[H18] E Horváth, P Kőrös, I Lakatos: *Két üzemen alapuló energiahatékony járművezérlés megvalósítása kísérleti járművön* Innováció és fenntartható felszíni közlekedés konferencia (IFFK) Budapest, Magyarország, 2017

[H19] E Horváth, C Pozna, R E Precup: Robot coverage path planning based on iterative structured orientation Acta Polytechnica Hungarica, Volume 15, Issue 2, DOI: 10.12700/APH.15.1.2018.2.12, Impact factor: 0.745, 2018

[H20] J Hollósi, E Horváth, C Pozna: *Two-stage Racetrack Segmentation Method using Color Feature Filtering and Superpixel-based Convolutional Neural Network* 12th IEEE International Symposium on Applied Computational Intelligence and Informatics Temesvár / Timişoara, Románia, 2018

[H21] E Horváth, C Pozna, Á Ballagi: *Evaluation of Neural Network-based Sensing and Perception in Experimental Vehicle*s 22nd IEEE International Conference on Intelligent Engineering Systems (INES2018) Las Palmas de Gran Canaria, Spain, 2018

[H23] E Horváth, Cs Hajdu, C Pozna, Á Ballagi: *Range Sensor-based Occupancy Grid Mapping with Signatures* 20th International Carpathian Control Conference ICCC Poland , 2019

[H24] E Horváth, C Pozna, Á Ballagi: *A Mobile Robot and Vehicle Occupancy Map Construction Model* - 23nd IEEE International Conference on Intelligent Engineering Systems (INES2019)- INES Hungary, 2019

[H25] E Horváth, Cs Hajdu, C Pozna, Á Ballagi: Range Sensor-based Occupancy Grid Mapping with Signatures 2019 20th IEEE International Carpathian Control Conference (ICCC), Kraków-Wieliczka, Poland, 2019

[H26] E Horváth, C Pozna, Á Ballagi: *Novel Pure-Pursuit Trajectory Following Approaches and their Practical Applications* 10th IEEE International Conference on Infocommunications, Naples, Italy, 2019

[H27] E Horváth, Cs Hajdu, C Pozna: *Enhancement of pure-pursuit path-tracking algorithm with multi-goal selection* IEEE International Conference on Gridding and Polytope Based Modeling and Control, Győr, Hungary, 2019

[H28] E Horváth, C Pozna, P Kőrös, Cs Hajdu, Á Ballagi: *Theoretical background and application of multiple goal pursuit trajectory follower*, Hungarian Journal of Industry and Chemistry, 2020

4. Bibliography

[1] H. Moravec, “Mind children: The future of robot and human intelligence,” *Harvard University Press,* 1988.

[2] R. Siegwart, “No need to worry!,” 2018.

[3] D. Anguelov, “Taming the Long Tail of Autonomous Driving Challenges,” Massachusetts, New England, USA, 2019.

[4] E. Galceran and M. Carreras, “A Survey on Coverage Path Planning for Robotics,” *Journal Robotics and Autonomous Systems,* vol. 61, no. 12, pp. 1258-1276, 2013.

[5] M. Waanders, “Coverage path planning for mobile cleaning robots,” in *15th Twente Student Conference on IT*, The Netherlands, 2011.

[6] H. Choset, “Coverage for robotics – A survey of recent results,” *Annals of Mathematics and Artificial Intelligence,* vol. 31, no. 1, pp. 113-126, 2001.

[7] C. Zengyu, L. Shuxia, Z. Ran and Z. Qikun, “Research on Complete Coverage Path Planning Algorithms based on A* Algorithms,” *The Open Cybernetics & Systemics Journal,* vol. 8, pp. 418-426, 2014.

[8] S. X. Yang and C. Luo, “A Neural Network Approach to Complete Coverage Path Planning,” *IEEE Transactions on Cybernetics,* vol. 34, no. 1, pp. 718 - 724, 2004.

[9] P. E. Missiuro, Map represents 3rd floor common area of the MIT Stata Center, http://dspace.mit.edu/handle/1721.1/62269: 2010.

[10] S. Thrun, W. Burgard and D. Fox., "A probabilistic approach to concurrent mapping and localization for mobile robots," *Autonomous Robots,* vol. 5, no. 3-4, pp. 253-271., 1998.

[11] E. Ivanjko, I. Petrovic and M. Brezak, “Experimental Comparison of Sonar Based Occupancy Grid,” *Automatika: Journal for Control, Measurement, Electronics, Computing & Communications,* vol. 50, no. 1-2., pp. 65-79, 2009.

[12] S. Thrun, “Learning Occupancy Grid Maps with Forward Sensor Models,” *Autonomous Robots,* vol. 15, pp. 111-127, 2003.

[13] S. Thrun, W. Burgard and D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), The MIT Press, 2005.

[14] L. Tai and M. Liu, "Deep-learning in Mobile Robotics - from Perception to Control Systems: A Survey on Why and Why not," *CoRR, Computing Research Repository,* 2016.

[15] Ł. Kaiser, A. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones and J. Uszkoreit, “One Model To Learn Them All,” *CoRR, Computing Research Repository,* 2017.

[16] D. G. Lowe, “Object recognition from local scale-invariant features,” *The proceedings of the seventh IEEE international conference on Computer vision,* vol. 2, pp. 1150-1157, 1999.

[17] H. Bay, T. Tuytelaars and L. V. Gool, “Surf: Speeded up robust features,” *European conference on computer vision,* pp. 404-417, 2006.

[18] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” *IEEE international conference on Computer Vision (ICCV),* pp. 2564-2571, 2011.

[19] C. C. Hau, Handbook Of Pattern Recognition And Computer Vision, Singapore: World Scientific, 2015.

[20] A. Karpathy, *Convolutional Neural Networks for Visual Recognition,* 2017.

[21] O. Russakovsky, H. S. Jia Deng, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV),* vol. 115, no. 3, pp. 211-252, 2015.

[22] Y. Bengio, “Learning deep architectures for AI,” *Foundations and trends in Machine Learning,* vol. 2, no. 15, pp. 1-27, 2009.

[23] A. Karpathy and e. al., Writers, *Convolutional Neural Networks for Visual Recognition notes accompany the Stanford class CS231.* [Performance]. 2017.

[24] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, MIT Press, 2016.

[25] C. Szegedy, A. Toshev and D. Erhan, “Deep neural networks for object detection,” *Advances in Neural Information Processing Systems,* pp. 2553-2561, 2013.

[26] S. Ruder, “Ruder, Sebastian. "An overview of gradient descent optimization algorithms,” *arXiv preprint,* 2016.

[27] J. Duchi, E. Hazan and Y. S. ". 1. (2011):, “Adaptive subgradient methods for online learning and stochastic optimization.,” *Journal of Machine Learning Research,* vol. 12, no. Jul, pp. 2121-2159., 2011.

[28] J. Duchi and Y. Singer, “Efficient online and batch learning using forward backward splitting,” *Journal of Machine Learning Research,* vol. 10, no. Dec, pp. 2899-2934., 2009.

[29] D. P. Kingma and J. L. Ba, “Adam: a Method for Stochastic Optimization,” *International Conference on Learning Representations,* p. 1–13, 2015.

[30] T. Tieleman, G. Hinton, N. Srivastava and K. Swersky, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *Coursera: Neural networks for machine learning,* pp. 26-31, 2012.

[31] J. Long, E. Shelhamer and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, UC Berkeley, 2015.

[32] V. Badrinarayanan, A. Kendall and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 2017.

[33] H. Noh, S. Hong and B. Han, “Learning Deconvolution Network for Semantic Segmentation,” *Computing Research Repository ,* vol. 1505, no. 04366, pp. 11520-11528, 2015.

[34] H. Choset and P. Pignon, “Coverage Path Planning: The Boustrophedon Decomposition,” in *International Conference on Field and Service Robotics*, Canberra, 1997.

[35] D. L. Applegate, R. E. Bixby, V. Chvátal and W. J. Cook, The Traveling Salesman Problem:, Princeton, USA: Princeton Univerity Press, 2006.

[36] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler and A. Y. Ng, “ROS: an open-source Robot Operating System cite,” *ICRA workshop on open source software,* vol. 3, no. 2, 2009.