Robust Parking Path Planning with Error-Adaptive Sampling under Perception Uncertainty

Seongjin Lee, Wonteak Lim and Myoungho Sunwoo *

Department of Automotive Engineering, Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul 04763, Korea; seongjin90@gmail.com (S.L.); lwt1849@gmail.com (W.L.)
* Correspondence: msunwoo@hanyang.ac.kr; Tel.: +82-2-2220-0453

Received: 13 May 2020; Accepted: 19 June 2020; Published: 23 June 2020

Abstract: In automated parking systems, a path planner generates a path to reach the vacant parking space detected by a perception system. To generate a safe parking path, accurate detection performance is required. However, the perception system always includes perception uncertainty, such as detection errors due to sensor noise and imperfect algorithms. If the parking path planner generates the parking path under uncertainty, problems may arise that cause the vehicle to collide due to the automated parking system. To avoid these problems, it is a challenging problem to generate the parking path from the erroneous parking space. To solve this conundrum, it is important to estimate the perception uncertainty and adapt the detection error in the planning process. This paper proposes a robust parking path planning that combines an error-adaptive sampling of generating possible path candidates with a utility-based method of making an optimal decision under uncertainty. By integrating the sampling-based method and the utility-based method, the proposed algorithm continuously generates an adaptable path considering the detection errors. As a result, the proposed algorithm ensures that the vehicle is safely located in the true position and orientation of the parking space under perception uncertainty.

Keywords: automated parking; planning under uncertainty; replanning; utility theory

1. Introduction

Automotive industries have increasingly provided drivers with a variety of functions for safety and convenience. Automated parking systems that park autonomously without driver intervention are able to save time and reduce the risk of accidents. The process of a parking system starts with the detection of parking spaces using Around View Monitoring (AVM) systems [1–4] or ultrasonic sensors [5–8]. Then, based on the detected parking space, the parking system generates a parking path that leads the vehicle to the parking space safely. The system ends with the vehicle following the generated parking path until arriving at the parking space. Parking space detection and parking path planning are both important technologies. This is because a safe parking path can be planned with the accurate position and orientation of the detected parking space. If there is a detection error in the parking space, the parking path might collide with obstacles.

However, parking path planning in real driving is a challenging problem. In real driving, accurate parking space detection is required, but there are three factors that disturb precise detection: sensor noise, poor environments and an immature perception algorithm. The sensor noise that is always included in sensors is unpredictable. Even if sensor noise is completely removed, the measurement itself may be poor. For example, due to strong light, the image of a parking space from the AVM sensor can be too bright to be useful. In such environments, the detection performance of perception algorithms may not be perfect. For this reason, the perception system always has detection errors. However, if the detection error is considered in the planning process, the parking system can...
ensure arrival at the actual parking space without failure. For example, if the system generates a new parking path when the parking fails, safety can be assured. Previous commercial automated parking systems stop the vehicle if it judges that there is an observation error in the parking space. Otherwise, the system might cause crashes into obstacles due to detection faults.

Over the last few years, some researchers have developed parking path planning algorithms that locate the vehicle in an error-free parking space. These parking path planning algorithms are divided into three types: geometric methods, graph-searching methods, and optimization methods. Geometric methods find a solution by combining geometric primitives, such as straight lines, circles, and spirals [9–11]. These methods are simple but only operated in a predefined situation. Graph-searching methods include Hybrid-A* or RRT [12–14]. These methods construct a graph from the environment, then find an optimal path in the graph. While these methods can generate a parking path in a variety of environments, the main disadvantage of these approaches is that the execution time increases exponentially as the graph becomes more complex in a narrow environment. Optimization methods are divided into two methods; the first group [15–17] utilizes Model Predictive Control (MPC) to solve a global optimization problem, and the second group [18,19] uses the Optimal Control Problem (OCP) to solve a local optimization problem. MPC-based methods are almost able to find a solution wherever the parking space is; however, they can take over one hundred seconds to find a solution [16,17]. On the other hand, OCP-based methods solve the problem with local optimization with the advantage that the execution time is similar to geometric methods’, and find an optimal path regardless of the vehicle’s position. Especially, the method proposed by Zips et. al. [18], has higher probability of finding a solution than Hybrid-A* which is considered as the state-of-the-art in Monte Carlo simulation. However, all algorithms including the OCP-based method are planned in a fully known environment. In other words, these algorithms cannot directly be used in real-driving, since they cannot consider the perception error from sensors. To reflect the perception error into such algorithms, additional methods for perception uncertainty are required.

From the perspective of perception uncertainty, there are probabilistic sampling methods and utility-based sampling methods for environments where perception errors exist. Some researchers [20–24] represent external uncertainty as the probability of the ego position and then sample nodes in the probability boundaries. On the other hand, other researchers [25–28] propose motion planning algorithms that consider environmental uncertainty as the probability of obstacle boundaries, then sample nodes by avoiding the obstacle boundaries. These algorithms sample a node of the searching-tree avoiding the uncertainty that the parking space has an error. In other words, these methods cannot explicitly express the uncertainty of the node. In addition, these methods take a long time to execute in narrow environments since they require an enormous number of sample nodes to find a goal in such environments. On the other hand, utility-based sampling methods are proposed to reflect perception uncertainty directly in the planning process [29–31]. Utility-based sampling methods explicitly express the uncertainty by directly representing the preference of the sample node. The advantage of utility-based sampling methods is that their expression is scalable from sample nodes to sample paths. For example, by formulating a user-defined utility function, both the preference and the uncertainty of each path can be estimated to select the best path. If the OCP-based method, which has the highest probability of finding a parking path, and the utility-based sampling method, which has scalability to paths, are combined, not only can the perception error in the real world be effectively considered but the execution time can also be short.

As explained so far, the OCP-based optimization method has the advantage of generating a parking path in narrow environments within a short execution time, and utility-based sampling methods have advantages of uncertainty expression from the perspective of re-planning. Therefore, if we generate path candidates through OCP-based optimization methods, then consider the uncertainty of parking spaces using utility-based sampling methods, automated parking systems can locate the vehicle in an actual parking space with a small error even if there is an error at the parking space. In this paper, we propose a parking path planning algorithm that recursively selects an optimal
path by considering the utility of each path among the path candidates generated using the OCP-based optimization method. Since the proposed algorithm directly reflects the uncertainty of the parking space in the planning process, the algorithm helps to accurately locate to the actual parking space. In addition, if the vehicle is regarded as not reaching the actual parking space, the proposed algorithm re-generates the optimal path.

The remainder of this paper is arranged as follows. The overview of the proposed method is provided in Section 2. Then, the proposed method is divided into three parts: Parking Space Sampling, Path Candidate Generation, and Optimal Path Selection. These are explained from Sections 3–5. The experimental results are shown in Section 6, and this paper is concluded in Section 7.

2. Overview

The overall process of parking path planning is described in Figure 1a. First, the vehicle, which is located in \( q_e \), tries to start from outside of the parking space. At that time, the parking space is defined as a space consisting of four corner points as shown in Figure 1b. The target state \( q_P = (x_P, y_P, \theta_P) \) indicates position and orientation of the parking space where the center of the rear wheel of the vehicle \( (q_e) \) should be at this location. The distance between the front line and \( q_P \), which is represented \( x_M \), depends on the vehicle length.

The target state \( q_P \) is estimated from the parking space detected by sensors. However, due to the perceptual error, the target state can be incorrectly estimated. The inaccurate target state causes the parking path to lead the vehicle out of the parking space. For this reason, we propose a robust parking path generation algorithm to move the vehicle accurately to the target state.

Figure 2. The overall architecture of re-plannable parking path planning: parking space sampling, path candidate generation, and optimal path selection.
The proposed method is shown in Figure 2. The whole process is divided into three steps: *Parking Space Sampling*, *Path Candidate Generation*, and *Optimal Path Selection*. In the first step, we sample the target states where the actual parking space may be using the perceptional error model. Then, the OCP-based algorithm is employed to generate parking paths for each sampled target state. Finally, we calculate the utility of each path candidate. During this process, the planner decides whether to re-plan by considering the uncertainty of the target state. Each step is described in detail in the following sections.

### 3. Parking Space Sampling

The precise position and orientation of the parking space, which is called a target state, is important for parking path planning. There are free space-based approaches [5–8] and separate line-based approaches [1–4] for recognizing the parking space. Free space-based approaches search for vacant space between obstacles. On the other hand, the separate line-based approaches find the vertical and horizontal lines of the parking slot. Then, these approaches find the corner points of the parking space from each vacant space or separate lines. For the precise detection of the target state, the position of the corner points should be accurate. However, due to sensor noise, the natural vibration of the vehicle, and imperfection of the perception algorithms, the position error of the detected corner points increase. This effect cannot be ignored in the parking path planning. Parking can easily fail if the parking planner uses the parking space generated from the erroneous corner points. To reduce the effect of the erroneous corner points in the planning process, we propose the perceptional error model. By using the perceptional error model, the corner points from the perception system are sampled. Then, the planner generates parking paths from each sampled target state using the sampled corner points. Through this process, the effect of the erroneous corner points can be minimized.

#### 3.1. Perceptional Error Model

The perceptional error starts from sensor noise and increases through the perception algorithm. Since perceptional errors are distributed differently according to the sensor type and algorithm, a different perceptional error model is required depending on the type. However, the objective of this paper is not to evaluate the quality of this model but whether our planner would work with a wide variety of models. We believe that the perception error model used for our experiments is representative and captures the important characteristics of frequently used perception modalities.

Many researchers have studied parking space detection algorithm using various sensors, such as ultrasonic sensors [5–8], AVM systems [1–4], and Lidar [32,33]. The AVM system is widely used for parking space detection since the AVM sensor can detect empty parking spaces, unlike ultrasonic sensors that can only search for empty spaces between two vehicles and has the advantage of being cheaper than Lidars. The AVM system calculates the location of a parking space from the image by applying the geometric relationship between the camera pose and the road surface.

During parking, when vehicle motion (pitch and roll) occurs, the geometric relationship can be affected by the motion, which will cause measurement error. The change of the geometric relationship should be adjusted for reducing the measurement error. However, it is difficult to compensate for the change in the geometric relationship since it is not easy to measure the roll and pitch motion of the vehicle. Due to this phenomenon, the corner points detected by the AVM includes position errors as shown in Figure 3a. The fixed points, which is represented as the red dot, are measured as the blue dots when the vehicle is moving. For simplification, the measured points are modeled as Gaussian distribution at each pixel as shown in Gauss distribution in Figure 3a, and this can be expanded at every pixel like Figure 3b from [34]. In the measurement model of the AVM features, the variances tend to increase with farther distance from the ego vehicle. For example, the farther detected distance from the sensor has higher covariance and the shorter detected distance has lower covariance.
Figure 3. Variance of the Gaussian noise distribution for the AVM system (b) due to the vehicle motion caused by roll and pitch (a).

The perceptional error model is estimated from the sensor noise distribution. Contrary to the AVM noise distribution, the perceptional probability that the true position is located in the pixel decreases as the detected distance increases. This is conducted using the cumulative distribution function (CDF) of the Gaussian as follows:

$$P_{detect}(d) = 1 - \frac{1}{2}(1 + erf\left(\frac{d}{\sigma \sqrt{2}}\right))$$

(1)

where "erf" is the Gauss error function and $d$ is the detected distance from the nearest camera sensor. $\sigma$ represents a tuning factor that controls the rate of the probability reduction of the detected distance. (1) represents an approximation of the true probability of the detected distance. Then, to apply the degradation from the perception algorithm, (1) is multiplied by the detection rate of algorithm $\lambda$.

$$P_{perception}(d) = \lambda \cdot P_{detect}(d)$$

(2)

The perception algorithm can recognize and localize the corner points of the parking space, then estimate the target state from the detected corner points. Therefore, $P_{perception}(d)$ is the perceptional probability for the corner points of the parking space. The distribution of the perceptional probability of our system is shown in Figure 4. The perceptional probability becomes small as the distance from the sensor increases.
3.2. Error-Adaptive Sampling

As mentioned earlier in this section, we sample the corner points to represent the uncertainty of the perception system as shown in Figure 5. The sampled corner points (the yellow dots in Figure 5) are sampled according to the perceptional probability of the corner points. The target state and the probability of the target state are then calculated using the sampled corner points. We construct the whole parking space by using the sampled two corner points as the left and right corners of the parking space, as shown in the gray rectangles in Figure 5. When configuring the parking space from the corner points, the length of the parking space is from the regulations. Then, to provide the target state to path candidate generation, the target state is estimated by considering the length of the vehicle. During this sampling process, we assume that the sampled corner points had become normally distributed. This is achieved using the Gaussian distribution function:

$$
\exp \left( -\frac{1}{2}(X - \hat{X})^T\Sigma^{-1}(X - \hat{X}) \right)
$$

where $X = (x, y)$ denotes the position of the corner points in Cartesian coordinates, $\hat{X}$ refers to the detected corner points from the perception algorithm, and $\Sigma$ indicates the standard deviation depending on the detection probability in Section 3.1. If the sampled corner points are assumed to be normal and binomial, the standard deviation can be formulated by using the normal approximation of the binomial as follows:

$$
\Sigma = N \cdot P \cdot (1 - P)
$$

where $N$ denotes the total number of samples. From (4), a large $N$ makes the standard deviation large, which allows for a wider range of errors to be considered. However, the execution time increases since the number of paths that need to be generated increases. On the other hand, a small $N$ ensures real-time execution, although only a narrow range of errors can be considered. In this paper, $N = 50$ is used to sample the parking spaces.
4. Path Candidate Generation

Since we utilize 50 samples of target states, a path candidate should be generated with a high success rate of path generation in a few milliseconds to apply the real-time application. In the comparison result in [18], the OCP-based method has the highest success rate in Monte Carlo simulation and takes less than 20 ms while Hybrid-A*, considered the state of the art, operates within 150 to 200 ms. That is the reason why the OCP-based method is used to Path Candidate Generation.

To generate a set of path candidates to reach each sampled parking space, the OCP-based method [18] is iteratively operated for each sampled target state. By applying this method, only one path can be optimized to achieve the sampled target state. The OCP-based method is formulated with the kinematic model of the vehicle and a cost function based on a pose error between the ego vehicle and the target state.

4.1. Vehicle Kinematic Model

In the following, the vehicle model, as shown in Figure 6, depicts the mathematical description. Since the velocity of the vehicle is not over 10 kilometers per hour during parking, the tire slip angle is neglected. Therefore the vehicle can be simplified with the bicycle model, which has one front and one rear wheel. The motion of the vehicle is characterized by the coordinates \((x, y)\) of reference point \(P\), which is located at the center of the rear axle, orientation \(\theta\) of the longitudinal axis, velocity \(v\) and steering angle \(\delta\). The differential equation of the vehicle model can be written as

\[
\frac{d}{dt} \begin{pmatrix} x \\ y \\ \theta \\ v \\ \delta \end{pmatrix} = \begin{pmatrix} v \cos \theta \\ v \sin \theta \\ \frac{v \tan \delta}{L} \\ a \\ \dot{\psi} \end{pmatrix}. \tag{5}
\]

Here, \(\mathbf{u}_A = (a, \dot{\psi})^T\) denotes the control input comprised of acceleration \(a\) and the steering angle rate \(\dot{\psi}\), and the parameter \(L\) refers to the wheelbase. For computational efficiency, this kinematic model can be decoupled by Path-Velocity-Decomposition [35]. With this decomposition, the velocity can be written as \(v = D \frac{ds}{dt}\), with the path length \(s\) and the direction of the vehicle \(D \in \{-1, 1\}\) (i.e., forward
or backward). The kinematic state of the vehicle \( \mathbf{q} = (x, y, \theta)^T \) can be obtained from (5), and the decoupled equation reads as follows:

\[
\dot{\mathbf{q}} = \begin{bmatrix}
x' \\
y' \\
\theta'
\end{bmatrix} = \begin{bmatrix}
D \cos \theta \\
D \sin \theta \\
Du_l
\end{bmatrix} = \mathbf{f}_s(\mathbf{q}, u_l, D)
\]  

(6)

\[
\frac{d}{dt} \begin{bmatrix} v \\ \delta \end{bmatrix} = \begin{bmatrix} a \\ \psi \end{bmatrix}
\]  

(7)

where \( (\cdot)' \) denotes the derivative with respect to the path length \( s \), and the new control input \( u_l = \tan \delta / L \). To apply discrete optimization, (6) can be discretized with respect to the path length \( s \) by using second-order Runge-Kutta discretization

\[
k_{1i} = \mu_i \mathbf{f}_s(\mathbf{q}_i, u_{li})
\]

(8)

\[
k_{2i} = \mu_i \mathbf{f}_s(\mathbf{q}_i + \frac{1}{2} k_{1i}, u_{li})
\]

(9)

\[
\mathbf{q}_{i+1} = \mathbf{q}_i + k_{2i} + \mathcal{O}(\mu_i^3)
\]

(10)

with the step length \( \eta \in [\eta_{\text{min}}, \eta_{\text{max}}] \). Note that the step length is constrained by a minimum and maximum value, \( \mu_{\text{min}} \) and \( \mu_{\text{max}} \), respectively. Neglecting the error term \( \mathcal{O}(\mu_i^3) \), the discretized equation reads as follows:

\[
\mathbf{q}_{i+1} = \begin{bmatrix} x_{i+1} \\ y_{i+1} \\ \theta_{i+1}
\end{bmatrix} = \begin{bmatrix}
x_i + D \eta_i \cos(\theta_i + D \frac{\eta_i u_{li}}{2}) \\
y_i + D \eta_i \sin(\theta_i + D \frac{\eta_i u_{li}}{2}) \\
\theta_i + D \eta_i u_{li}
\end{bmatrix} = \mathbf{f}(\mathbf{q}_i, u_i, D)
\]

(11)

The new control input \( \mathbf{u} = (u_i, \eta)^T \) of (11) consists of the steering input \( u_l \in [-0.3, 0.3][\text{rad}] \) and the step length \( \eta \in [0.1, 0.5][\text{m}] \).

Figure 6. Vehicle kinematic model.

4.2. Parking Path Planning

Based on (11), a sequence of static optimization problems is formulated. The whole parking path is not calculated at once but only for one incremental step. Then, the optimization problem solves in a
recursive manner with the calculation of a direction change point. The next subsection will explain how to change the direction. The one-step static optimization problem can be formulated as

$$\text{argmin}_{u_i} \quad l_{O_i}(q_{i+1})$$

subject to

$$q_{i+1} = f(q_i, u_i, D)$$

$$h_p(q_{i+1}) \leq 0$$

$$u_{\text{min}} \leq u_i \leq u_{\text{max}}$$

where the objective function (12) is expressed as,

$$l_{O_i}(q_i) = r_\theta e_{\theta i}^2 + e_i^T R e_i.$$  

Here, $e_i = (x_i - x_{\text{target}}, y_i - y_{\text{target}})^T$ denotes the distance of the vehicle at iteration step $i$ to the target position $(x_{\text{target}}, y_{\text{target}})^T$ and $e_{\theta i}$ refers to the difference between the orientation of the vehicle $\theta_i$ and the orientation of the target orientation $\theta_{\text{target}}$. The weighting term $R$ and $r_\theta$ with respect to each error should be positive definite and a positive value, respectively. There is one equality constraint and two inequality constraints. The first constraint in Equation (13) is equal to (11) for the vehicle kinematic constraint, as mentioned in Section 4.1. Then, the two inequality constraints in (14) and (15) are for the collision and control inputs, respectively. The collision constraint can be formulated with the Minkowski sum [36], and the control input constraint should be within the physical limits of the vehicle, which are the maximum steering angle and the maximum step length in one optimization iteration. Sequential quadratic programming (SQP) is used for solving optimization problem (12) to (15). SQP is one of the most successful methods for the numerical solution of constrained nonlinear optimization problems. SQP is an iterative procedure which models the nonlinear problem (NLP) for a given iterate by a quadratic programming (QP) subproblem, solves that QP subproblem, and then uses the solution to construct a new iterate. This construction is done in such a way that the sequence converges to a local minimum.

In every iteration step, the position and orientation of the vehicle are improved with respect to the cost function (16). In Figure 7, $q_p$ refers to the target state of the parking space from Section 3 and $q_s$ denotes the start position and orientation of the ego vehicle. The state of each iteration step $q_i$ starts from $q_p$ and ends when $q_s$ reaches $q_s$. The optimization problem is only for one optimization step; this problem might be stuck in local minima. To solve the local minima problem, the algorithm is separated into two phases (A and B). During the phase A, the ego vehicle tries to escape from the parking space. The easiest way to escape the parking space is moving straight forward without any steering angle. At this time, the target orientation $\theta_{\text{target}}$ is set to zero for the straight movement. For the same reason, $r_\theta = 1.0$ and $R$ are set to 1.0 and 0, respectively. The phase A continues until the vehicle reaches $x_m$ in Figure 7. After leaving the parking space, the second phase B is a step to locate the ego vehicle to the start pose $q_s$ with a direction change strategy which is explained in the following subsection. The first step of the phase B is changing the target state. Since the objective of the phase B is to align in the start position and orientation, the target state should be set to the start state $q_{\text{target}} = q_s$. The allocation movement in the phase B is different depending on $r_\theta$ and $R$. For example, high $r_\theta$ value makes the vehicle rotate first, while high $R_{(1,1)}$ value firstly reduces the x-directional error and high $R_{(2,2)}$ affects to reduce the y-directional error. To decide the weighting terms, we consider the way human parking. For example, the human driver firstly moves forward until passing the parking space in the y-direction. Then, he rotates the vehicle in the diagonal direction of the parking space. Finally, the vehicle is aligned in the x-direction and orientation of the parking space. From this, we can expect that the order of the important weighting terms is the y-direction ($R_{(2,2)}$), the orientation ($r_\theta$) and the x-direction ($R_{(1,1)}$). However, we find the path from the parking space $q_p$ to the start pose $q_s$ so that the order should be reversed. In this paper, we select $r_\theta = 3.5$ and $R = \text{diag}(25, 1)$, respectively.
4.3. Direction Switching Point

To find a path before arriving at the starting position, the vehicle may exhibit changes in direction. There are two heuristic rules for directional change. The first rule is switching direction when the vehicle cannot move in that direction as shown in Figure 7. This means that the vehicle will collide with obstacles. The second rule is changing the direction when the current result of optimization $l_{O^i}$ is larger than the previous result $l_{O^{i-1}}$, as shown in Figure 7. The larger value of the cost function refers to a more distant position from the starting position. Therefore, the vehicle changes its direction to reduce the distance and orientation error from the starting position.

4.4. Obstacles

To generate a collision-free path, obstacles need to be expressed as constraints. If obstacles can be considered convex polygons, the computation time for optimization will be reduced. Therefore, obstacles are regarded as convex polygons. If the obstacle is non-convex, the obstacle was divided into a set of convex polygons through convex decomposition [36]. All convex polygonal obstacles are described as larger than their original size. During optimization, obstacles are transformed into inequality constraints in the cost function. The inequality constraints check whether the vehicle is within the obstacle boundary.

5. Optimal Path Selection

After generating path candidates, the planner should decide the optimal path that is likely to reach the true target state. This selection process does not work only once, but continues until the vehicle reaches the target state. However, if the path is continuously changed by the optimal selection, this path loses consistency with the previous path. That is why the re-planning strategy is required. In this section, we discuss the optimal selection and re-planning strategy.

5.1. Utility-Based Decision

This section focuses on how to make rational decisions based on the perceptional error model and utility function. This section begins by introducing the foundations of utility theory [37,38] and showing how it forms the basis for rational decision making under uncertainty. Based on utility theory, the degree of desirability with respect to two different statements can be compared and measured.
The real-valued degree of desirability is called utility, and the utility of path \( s \) can be written as \( U(s) \). The utility is composed of a function of various task-specific considerations. The expected utility integrates the notions of utilities, weighted by the probability that the path is successful. This probability is consistent with the probability of the sampled target state in Section 3. The expected utility \( EU \) of path \( s \) is written as

\[
EU(s) = P(s) \cdot U_{\text{ideal}}(s) + (1 - P(s)) \cdot U_{\text{real}}(s)
\]

(17)

where \( U_{\text{ideal}}(\cdot) \) measures how close the path candidate is to the detected target state. Through \( U_{\text{ideal}} \), the ideal path to the actual target state is selected. \( U_{\text{real}}(\cdot) \) estimates how near the path candidate is to the ego vehicle. The high value of \( U_{\text{real}} \) means that the realistic path near the ego vehicle would be selected. \( P(\cdot) \) measures the probability of the path candidate. The probability of the path candidate is equal to the probability of the sampled target state because only one path candidate can be generated from one target state. Based on (17), the probability of the path candidate determines whether the path is ideal or realistic. For example, if there is an ideal path candidate that has a high probability close to the detected target state, the vehicle follows the ideal path candidate. On the other hand, if all path candidates have low probability, the vehicle selects the realistic path candidate near the ego vehicle. The expected utility of path candidates is shown in Figure 8. The expected utility is higher when the path candidate is close to the target state and near the ego vehicle. Each of terms in (17) is covered in detail in the following section.

![Figure 8](image-url). The utility of each path candidate is estimated; the path that is the nearest to the actual target state has the maximum expected utility.

5.1.1. Ideal Utility Function

The desirable path for achieving the goal is the path nearest to the currently detected target state. Even though there is an error in the currently detected target state, we assume that the perceptional system gives the most reliable information from insufficient information. To formulate the degree of nearness, a new parameter, \( d_{\text{ideal}} \), which is the weighted summation of the position difference and
the orientation difference between the end point of the path candidate and the sampled target state is formulated as

\[ d_{\text{ideal}} = e_d^T Q_d e_d \] (18)

where \( e_d = (x_s - x_d, y_s - y_d, \psi_s - \psi_d) \) denotes the position difference and the orientation difference in Cartesian coordinates, then \((x_s, y_s, \psi_s)\) is the position and orientation of the end point of the path candidate. Likewise, \((x_d, y_d, \psi_d)\) indicates the position and orientation of the current detected target state. \( Q_d \) denotes the positive definite matrix for weighting each parameter and is a \(3 \times 3\) matrix equal to \( \text{diag}(w_{x_d}, w_{y_d}, w_{\psi_d}) \). When \( w_{x_d} \) and \( w_{y_d} \) are high, the utility of the path candidate that has the end point near the target state is high. Likewise, when \( w_{\psi_d} \) is high, the utility of the path candidate that arranges the vehicle accurately is high.

To define a normalized utility function, the maximum utility is assigned 1 and the worst utility is assigned 0. By using the Gauss error function \( \text{erf} \), the utility function can be normalized as

\[ U_{\text{ideal}}(s) = \frac{1}{2} \left( 1 - \text{erf} \left( \frac{d_{\text{ideal}}}{\sigma_d \sqrt{2}} \right) \right) \] (19)

The maximum value is calculated at \( d_{\text{ideal}} = 0 \) and the minimum value is calculated at \( d_{\text{ideal}} = \infty \). \( \sigma_d \) denotes the uncertainty of the perception system. This means the decreasing rate from \( d_{\text{ideal}} = 0 \) to \( d_{\text{ideal}} = \infty \). For example, since a small \( \sigma_d \) reduces the utility even if it is slightly different from \( d_{\text{ideal}} = 0 \), the perception system is regarded as certain. Conversely, a high \( \sigma_d \) achieves similar utility around \( d_{\text{ideal}} = 0 \).

5.1.2. Realistic Utility Function

The reality of the path candidate is the degree to which the path candidate can be followed and punishes the collision path candidate. To check the physical collision, the Minkowski sum [36] is used. Since the output of the Minkowski sum is true or false, we use the Minkowski function by multiplying negative values to punish a collision. Then, similar to the ideal utility, the parameter \( d_{\text{real}} \), which represents the closeness of the path candidate, is formulated as:

\[ d_{\text{real}} = e_e^T Q_e e_e \] (20)

where \( e_e = (x_e - x_n, y_e - y_n, \psi_e - \psi_n) \) refers to the position difference and orientation difference between the ego vehicle and the nearest path point. The subscript notations \( e \) and \( n \) represent the ego vehicle and the nearest path point, respectively. \( Q_e \) denotes the positive definite matrix for weighting the error and is equal to \( \text{diag}(w_{x_e}, w_{y_e}, w_{\psi_e}) \). Then, the Gauss error function can also be used to normalize utility. The total realistic utility function is as follows,

\[ U_{\text{real}}(s) = \frac{1}{2} \left( 1 + \text{erf} \left( \frac{d_{\text{real}}}{\sigma_e \sqrt{2}} \right) \right) + \alpha \cdot f_{\text{Minkowski}} \] (21)

where \( f_{\text{Minkowski}} \) refers to the Minkowski function, which checks a collision, and \( \alpha = -10,000 \) denotes the punishment factor for not selecting the collision path candidate.

5.2. Re-Planning Strategy

The overall re-planning diagram is shown in Figure 9. The initial information might be insufficient for locating the vehicle in the actual parking space because of the uncertainty of the perception system. If the uncertainty is high or all paths are considered to collide, the maximum expected utility may be lower than the threshold utility \( U_{\text{threshold}} \). In this case, it is regarded that there is no available path. Then, the planner’s only option is to return the Parking Space Sampling.
Figure 9. Flow diagram of the overall re-planning strategy.

When the maximum expected utility is over $U_{\text{threshold}}$, the planner starts to park. However, due to the noisy measurement, if the planner selects the optimal path at every step, the vehicle loses its consistency. In other words, the vehicle will stagger. To achieve stability, the optimal path is only selected when the vehicle stops.

6. Experimental Results

We evaluated the proposed algorithm in two scenarios as shown in Figure 10. The first scenario was in the parking lot, where other obstacles did not exist. In the second scenario, the ego vehicle tried to park in a parking space where other vehicles were already parked on both sides of it. The evaluation shows how many errors the algorithm can accommodate during parking compared with the planner without re-planning, and the position and orientation error were calculated in the test scenarios.

The tests were conducted using the test vehicle ZOE from Hanyang University (Figure 11). The dimensions of the vehicle were as follows: the total length was 4.084 m, the total width was 1.730 m, and the wheelbase was 2.845 m. Then, we chose a parking lot with the dimensions $2.3 \times 5.0$ m, as specified by the Department of Transportation. The surface of the parking lot was uneven, inclined or declined. Such parking lots might cause unexpected errors for a perception system. Nevertheless, we wanted to show that the proposed system completes parking in these conditions.
Figure 10. The AVM images of two test scenarios: open scenarios and occupied scenarios. (a) An open scenario; there is no other obstacles around the vehicle, (b) An occupied scenario; there are other obstacles around the vehicle.

Figure 11. Test vehicle ZOE equipped with AVM and motion sensors.

The test vehicle was equipped with AVM and wheel speed sensors. For parking space detection, the semantic segmentation-based algorithm from AVM images [39] was implemented. This algorithm finds the intersecting points between the vertical lines and horizontal lines from the semantic segmented lines of the parking spaces. The position of the vehicle was estimated by dead reckoning using a motion sensor [17,40,41]. Then, the vehicle was controlled by control inputs with Model Predictive Control (MPC) in the lateral [42–45] and longitudinal directions [46,47] separately.

The proposed planning algorithm was implemented in an i5 Core embedded PC in the C++ language with a Robot Operating System (ROS) environment [48]. The static optimisation problem is solved using the SQP-algorithm of the numeric software library NLopt [49,50]. The execution periods of parking space sampling, path candidate generation, and utility-guided sampling were 50 milliseconds, respectively. All the algorithms were operated in multiple threaded environments.

6.1. Scenario 1: Open Scenario

The parking process in the open scenario (see Figure 10a) without a re-planning scheme is shown in Figure 12. In this case, the planner generated a parking path (the red line) using the first detected parking space (the blue dashed rectangle). The driving trajectory of the ego vehicle is shown as the black dashed line in Figure 12c. As a result, the vehicle was in the first detected parking space even
though there was a detection error in this space. In this case, there were no obstacles around the vehicle; however, if other vehicles were already parked next to the ego vehicle, this position error might have caused an accident.

On the other hand, Figure 13 shows the overall process of the proposed algorithm. Firstly, the sampled parking spaces were created such as the light gray rectangles in Figure 13 at $t = 0$ s. Then, each utility of the generated path candidates was calculated and described as spectrum lines. Through this process, the optimal path, which was considered a perception error, was selected at $t = 0$ s (the red line). The vehicle started from the optimal path. Then, the detected parking spaces, which are represented as dark gray rectangles, continuously changed during parking. Due to the changed parking space, the vehicle followed another optimal path when the maximum expected utility was calculated again using (17). The position and orientation difference between the first detected parking space and the last detected parking space (the blue dashed rectangle and the gray rectangle in Figure 13c) are over 15 cm and 2.5 deg, respectively. This value can cause a crash if there is another vehicle next to the parking space. Through the proposed algorithm, the vehicle reached the actual parking space with a smaller error than the planner without re-planning. For understanding, the entire process of the re-planning is described in the Supplementary Materials.

(a) $t = 0.0$ s  
(b) $t = 18.0$ s  
(c) $t = 37.0$ s  
(d) $t = 37.0$ s

**Figure 12.** The open scenario without re-planning; there were position and orientation differences between the first (a) and last detected parking spaces (c).

(a) $t = 0.0$ s  
(b) $t = 24.0$ s  
(c) $t = 40.0$ s  
(d) $t = 40.0$ s

**Figure 13.** The open scenario with re-planning; there were also differences between the first and last detected parking spaces; however, the proposed algorithm changed the optimal path according to position and orientation differences.
6.2. Scenario 2: Occupied Scenario

The process of the proposed algorithm in the occupied scenario is shown in Figure 14. At \( t = 0.0 \) s, the vehicle tried to park in the parking space between two other vehicles. The proposed planner first generated the path candidates as shown by the spectrum lines in Figure 14. Unlike the open scenario, the utilities of the path candidates were higher than the open scenario case since the detected parking space was closer than the open scenario case. The closer parking space made the probability of the parking space high. This is why the spectrum lines appeared more yellow. In addition, the path candidate that was considered to be colliding was punished by Equation (21). Then, the proposed algorithm selected the optimal path (the red line) and started to follow it. At \( t = 18.0 \) s, the planner changed the optimal path since the change of the parking space was recognized and the previous optimal path collided with the other vehicle. Similarly, the proposed algorithm constantly changed the path each time it stopped, checking for collisions and changes in the parking space until it reached the parking space. As shown by the blue dashed rectangle of Figure 14c, the parking space was finally changed by over 20 cm and 5 deg in the x and y direction and orientation, respectively. If the vehicle had not changed its path, it would have collided with obstacles. Even though there was a change in the parking space, the proposed algorithm accommodated the perception error and then re-planned the optimal path in the erroneous parking space.

![Figure 14](image)

**Figure 14.** The occupied scenario with re-planning; the proposed algorithm detected the change of the parking space, then, changed its optimal path to locate the vehicle in the actual parking space.

We evaluated each scenario 100 times to check the capability and robustness of the proposed algorithm. The results were verified by the average error of the position and orientation. As shown in Figure 15, the average error of the proposed algorithm was reduced by 27.6, 19.8, and 46.3 percent in each x and y direction and orientation. These reduced values prevented collisions with other vehicles in the occupied cases. In addition, the success rate of the proposed algorithm was increased from 86 to 93 percent compared to the planner without re-planning. At that time, parking was regarded as fail when some of the vehicle crossed the parking slot marker.
7. Conclusions

This paper presents adaptive parking path planning for uncertain environments. The proposed algorithm was comprised of three steps: parking space sampling, path candidate generation, and optimal path selection. To reflect the uncertainty of the perception system, a perceptional error model was proposed in the parking space sampling step. The perceptional error model estimated the probability of the parking space, which was the result of the perception system, to sample possible parking spaces. By applying optimization-based parking path generation, a number of parking path candidates were quickly generated to achieve the sampled goals in path candidate generation. Then, each utility of the path candidates was evaluated from the perspectives of reachability and consistency in the optimal path selection step. While the vehicle was parking, the utilities of each candidate were changed according to the detected parking space and the previous optimal path, then the new optimal path was selected to reach the actual parking space accurately.

The experimental results show the capability to continuously generate the adaptive paths for the perception errors. Through the proposed algorithm, the vehicle could reach an actual parking space even though the input of the planner, which was the currently detected parking space, was continuously changed. In such erroneous perception systems, the proposed algorithm could locate the vehicle in an actual parking space with fewer errors than the parking path planner that does not replan. Moreover, the feasibility of re-planning is shown when the vehicle is expected to collide. The extensive experiments prove that the proposed algorithm is able to provide the safest path among the possible parking path candidates.

Nevertheless, future work will be extended in four ways. On the first hand, the idea is to include various sensor models and detection algorithms which are not considered in this paper. Sensor model can be expanded to Radar and ultrasonic sensors which are widely used to the automated parking system. The second is to cover not only perpendicular parking but also parallel parking, angle parking,
and forward parking. This can be covered in Path Candidate Generation part by changing the appropriate target position and orientation. The third approach may be extended by dealing with dynamic obstacles that can cause the change of the path. Finally, we expand the proposed method to automated valet parking system. To achieve the automated valet parking system, it is necessary to consider dynamic obstacles in areas that are blinded due to vehicles in the parking lot.

**Supplementary Materials:** The following are available at http://www.mdpi.com/1424-8220/20/12/3560/s1, Video S1: Robust 442 Parking Path Planning with Error-Adaptive Sampling under Perception Uncertainty. A supporting video article is 443 available at doi: link.

**Author Contributions:** Conceptualization, S.L. and W.L.; methodology, S.L. and W.L.; software, S.L.; validation, S.L.; formal analysis, S.L.; investigation, S.L. and W.L.; resources, S.L.; data curation, S.L.; writing—original draft preparation, S.L. and W.L.; writing—review and editing, S.L.; visualization, S.L.; supervision, S.L.; project administration, M.S.; funding acquisition, M.S. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was financially supported by the BK21 plus program (22A20130000045) under the Ministry of Education, Republic of Korea, the Industrial Strategy Technology Development Program (No. 10039673, 10079961), the International Collaborative Research and Development Program (N0001992) under the Ministry of Trade, Industry and Energy (MOTIE Korea), and National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (No. 2011-0017495).

**Conflicts of Interest:** The authors declare no conflict of interest.

**References**

1. Lee, S.; Hyeon, D.; Park, G.; Baek, I.J.; Kim, S.W.; Seo, S.W. Directional-DBSCAN: Parking-slot detection using a clustering method in around-view monitoring system. *IEEE Intell. Veh. Symp. Proc.* 2016, 457, 349–354. [CrossRef]
2. Tseng, D.C.; Chao, T.W.; Chang, J.W. Image-based parking guiding using Ackermann steering geometry. *Appl. Mech. Mater.* 2013, 437, 823–826. [CrossRef]
3. Hamada, K.; Hu, Z.; Fan, M.; Chen, H. Surround view based parking lot detection and tracking. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Seoul, Korea, 28 June–1 July 2015; pp. 1106–1111. [CrossRef]
4. Suhr, J.K.; Jung, H.G. Automatic Parking Space Detection and Tracking for Underground and Indoor Environments. *IEEE Trans. Ind. Electron.* 2016, 63, 5687–5698. [CrossRef]
5. Park, W.J.; Kim, B.S.; Seo, D.E.; Kim, D.S.; Lee, K.H. Parking space detection using ultrasonic sensor in parking assistance system. In Proceedings of the IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 4–6 June 2008; pp. 1039–1044. [CrossRef]
6. Kianpisheh, A.; Mustaffa, N.; Limtrairut, P.; Keikhosrokian, P. Smart Parking System (SPS) architecture using ultrasonic detector. *Int. J. Softw. Eng. Appl.* 2012, 6, 55–58.
7. Jeong, S.H.; Choi, C.G.; Oh, J.N.; Yoon, P.J.; Kim, B.S.; Kim, M.; Lee, K.H. Low cost design of parallel parking assist system based on an ultrasonic sensor. *Int. J. Automot. Technol.* 2010, 11, 409–416. [CrossRef]
8. Satonaka, H.; Okuda, M.; Hayasaka, S.; Endo, T.; Tanaka, Y.; Yoshida, T. Development of parking space detection using an ultrasonic sensor. In Proceedings of the 13th World Congress, London, UK, 8–12 October 2006; pp. 1–8.
9. Hsieh, M.F.; Ozguner, U. A parking algorithm for an autonomous vehicle. In Proceedings of the IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 4–6 June 2008; pp. 1155–1160. [CrossRef]
10. Lim, W.; Kim, J.; Jo, K.; Jo, Y.; Sunwoo, M. Turning Standard Line (TSL) Based Path Planning Algorithm for Narrow Parking Lots. *SAE Tech. Pap.* 2015, 1, 298.
11. Jolly, K.G.; Sreerama Kumar, R.; Vijayakumar, R. A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Rob. Autom. Syst.* 2009, 57, 23–33. [CrossRef]
12. Yoon, J.; Crane, C.D. Path planning for unmanned ground vehicle in urban parking area. In Proceedings of the 11th International Conference on Control, Automation and Systems, Gyeonggi-do, Korea, 26–29 October 2011; pp. 887–892.
13. Han, L.; Do, Q.H. Unified path planner for parking an autonomous vehicle based on RRT. In Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 5622–5627. [CrossRef]
14. Dolgov, D.; Thrun, S.; Montemerlo, M.; Diebel, J. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. J. Rob. Res.* 2010, 29, 485–501. [CrossRef]

15. Sakaeta, K.; Oda, T.; Nonaka, K.; Sekiguchi, K. Model predictive parking control with on-line path generations and multiple switching motions. In Proceedings of the IEEE Conference on Control Applications (CCA), Sydney, Australia, 21–23 September 2015; pp. 804–809. [CrossRef]

16. Li, B.; Shao, Z. A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. *Knowl. Based Syst.* 2015, 86, 11–20. [CrossRef]

17. Kobayashi, T.; Majima, S. Real-time optimization control for parking a vehicle automatically. In Proceedings of the IEEE International Vehicle Electronics Conference, Tottori, Japan, 25–28 September 2001; pp. 97–102. [CrossRef]

18. Zips, P.; Böck, M.; Kugi, A. Optimisation based path planning for car parking in narrow environments. *Rob. Autom. Syst.* 2016, 79, 1–11. [CrossRef]

19. Yamamoto, M.; Hayashi, Y.; Mohri, A. Garage parking planning and control of car-like robot using a real-time optimization method. In Proceedings of the IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing, Montreal, QC, Canada, 19–21 July 2005; pp. 248–253. [CrossRef]

20. Bry, A.; Roy, N. Rapidly-exploring random belief trees for motion planning under uncertainty. In Proceedings of the IEEE international conference on robotics and automation, Shanghai, China, 9–13 May 2011; pp. 723–730. [CrossRef]

21. Platt, R.; Tedrake, R.; Kaelbling, L.; Lozano-Pérez, T. Belief space planning assuming maximum likelihood observations. *Rob. Sci. Syst.* 2011, 6, 291–298.

22. Van Den Berg, J.; Abbeel, P.; Goldberg, K. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *Int. J. Rob. Res.* 2011, 30, 895–913. [CrossRef]

23. Patil, S.; Van Den Berg, J.; Alterovitz, R. Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty. In Proceedings of the IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, 14–18 May 2012; pp. 3238–3244. [CrossRef]

24. Chakravorty, S.; Kumar, S. Generalized sampling-based motion planners. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* 2011, 41, 855–866. [CrossRef] [PubMed]

25. Vitus, M.P.; Tomlin, C.J. Belief space planning for linear, Gaussian systems in uncertain environments. *IFAC Proc. Vol.* 2011, 44, 5902–5907. [CrossRef]

26. Missiuro, P.E.; Roy, N. Adapting probabilistic roadmaps to handle uncertain maps. In Proceedings of the IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006; pp. 1261–1267. [CrossRef]

27. Blackmore, L.; Li, H.; Williams, B. A probabilistic approach to optimal robust path planning with obstacles. In Proceedings of the American Control Conference, Minneapolis, MN, USA, 14–16 June 2006. [CrossRef]

28. Blackmore, L.; Ono, M.; Williams, B.C. Chance-constrained optimal path planning with obstacles. *IEEE Trans. Rob.* 2011, 27, 1080–1094. [CrossRef]

29. Burns, B.; Brock, O. Sampling-based motion planning with sensing uncertainty. In Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3313–3318. [CrossRef]

30. Burns, B.; Brock, O. Single-query motion planning with utility-guided random trees. In Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10–14 April 2007; pp. 3307–3312. [CrossRef]

31. Pongpunwattana, A.; Rysdyk, R. Real-time planning for multiple autonomous vehicles in dynamic uncertain environments. *J. Aerosp. Comput. Inf. Commun.* 2004, 1, 580–604. [CrossRef]

32. Im, G.; Kim, M.; Park, J. Parking line based SLAM approach using AVM/LiDAR sensor fusion for rapid and accurate loop closing and parking space detection. *Sensors* 2019, 19, 4811. [CrossRef]

33. Lee, B.; Wei, Y.; Guo, I.Y. Automatic parking of self-driving car based on LiDAR. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* 2017, 42, 241–246. [CrossRef]

34. Jo, K.; Jo, Y.; Suhr, J.K.; Jung, H.G.; Sunwoo, M. Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras. *IEEE Trans. Intell. Transp. Syst.* 2015, 16, 3377–3392. [CrossRef]
35. Kant, K.; Zucker, S.W. Toward Efficient Trajectory Planning: The Path-Velocity Decomposition. *Int. J. Rob. Res.* **1986**, *5*, 72–89. [CrossRef]

36. Lee, I.K.; Kim, M.S.; Elber, G. Polynomial/Rational Approximation of Minkowski Sum Boundary Curves. *Graph. Model. Image Process.* **1998**, *60*, 136–165. [CrossRef]

37. Jensen, N.E. An Introduction to Bernoullian Utility Theory: I. Utility Functions. *Swed. J. Econ.* **1967**, *69*, 163–183. [CrossRef]

38. Kochenderfer, M.J. *Decision Making Under Uncertainty: Theory and Application*, 1st ed.; MIT Press: Cambridge, MA, USA, 2015; pp. 57–75. [CrossRef]

39. Jang, C.; Sunwoo, M. Semantic segmentation-based parking space detection with standalone around view monitoring system. *Mach. Vis. Appl.* **2019**, *30*, 309–319. [CrossRef]

40. Akai, N.; Morales, L.Y.; Yamaguchi, T.; Takeuchi, E.; Yoshihara, Y.; Okuda, H.; Suzuki, T.; Ninomiya, Y. Autonomous driving based on accurate localization using multilayer LiDAR and dead reckoning. In Proceedings of the 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 1–6.

41. Sabet, M.T.; Mohammedi Daniali, H.; Fathi, A.; Alizadeh, E. A Low-Cost Dead Reckoning Navigation System for an AUV Using a Robust AHRS: Design and Experimental Analysis. *IEEE J. Ocean. Eng.* **2017**, *43*, 927–939. [CrossRef]

42. Falcone, P.; Eric Tseng, H.; Borrelli, F.; Asgari, J.; Hrovat, D. MPC-based yaw and lateral stabilisation via active front steering and braking. *Veh. Syst. Dyn.* **2008**, *46*, 611–628. [CrossRef]

43. Gao, Y.; Lin, T.; Borrelli, F.; Tseng, E.; Hrovat, D. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In Proceedings of the Dynamic Systems and Control Conference, Cambridge, MA, USA, 12–15 September 2010; pp. 265–272. [CrossRef]

44. Gray, A.; Gao, Y.; Lin, T.; Hedrick, J.K.; Tseng, H.E.; Borrelli, F. Predictive control for agile semi-autonomous ground vehicles using motion primitives. In Proceedings of the American Control Conference (ACC), Montreal, QC, Canada, 27–29 June 2012; pp. 4239–4244. [CrossRef]

45. Anderson, S.J.; Peters, S.C.; Pilutti, T.E.; Iagnemma, K. Experimental study of an optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios. In *Field and Service Robotics*; Howard, A., Iagnemma, K., Kelly, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 62, pp. 59–68. [CrossRef]

46. Li, S.; Li, K.; Rajamani, R.; Wang, J. Model predictive multi-objective vehicular adaptive cruise control. *IEEE Trans. Control Syst. Technol.* **2010**, *19*, 556–566. [CrossRef]

47. Quigley, M.; Gerkey, B.; Conley, K.; Faust, J.; Foote, T.; Leibs, J.; Berger, E.; Wheeler, R.; Ng, A. ROS: An Open-Source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, 2009. Available online: https://www.willowgarage.com/sites/default/files/icross09-ROS.pdf (accessed on 23 June 2020).

48. Kraft, D. A Software Package for Sequential Quadratic Programming. Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt. 1988. Available online: http://www.opengrey.eu/item/display/10068/147127 (accessed on 23 June 2020).

49. Johnson, S.G. The NLopt Nonlinear-Optimization Package. 2014. Available online: http://github.com/stevengj/nlopt (accessed on 23 June 2020).