Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Improving spatial trajectory planning by using an enhanced road representation

Edoardo Casapietra, Thomas H Weisswange, Franz Kummert, Christian Goerick

2017

Preprint:

This is an accepted article published in 4th International Symposium on Future Active Safety Technology: Toward zero traffic accidents (FAST-zero'17). The final authenticated version is available online at: https://doi.org/[DOI not available]

FAST-zero '17

IMPROVING SPATIAL TRAJECTORY PLANNING BY USING AN ENHANCED ROAD REPRESENTATION

Edoardo Casapietra [1)2)] Thomas H. Weisswange [2)] Christian Goerick [2)] Franz Kummert [1)]

1) University of Bielefeld, Cor-Lab
Universitätsstraße 25, D-33615 Bielefeld, Germany (E-mail: eddy1988@jsar.or.jp)
2) Honda Research Institute Europe GmbH
Carl-Legien-Straße 30, 63073 Offenbach am Main, Germany

ABSTRACT: Modern Advanced Driver Assistance Systems have to make plans on lateral and longitudinal motion of the vehicle. Classical systems handle only either short term dynamics (e.g. evasion), which requires knowledge about obstacles versus free space in the direct vicinity, or long term dynamics, as in navigational route planning, which requires knowledge about the road network. To advance the functionality towards automated driving systems, the gap between these two levels has to be closed by providing medium-level trajectories to determine the behavior of the vehicle for the next seconds. Such trajectory planning requires a spatial and semantic road representation of sufficient detail to ensure smooth and safe maneuvers. The required range of such a representation is hard to get only using current sensor-based direct road detection methods. State-of-the-art systems circumvent this problem by relying on highly precise road maps, which are expensive to get and present limitations in terms of short-term changes like constructions sites. In this paper we present an overview of our approach which provides a spatial road representation combining information from direct detection methods with additional indirect estimation means, which allows us to increases the range of the road representation. We introduce a system that uses real world data and includes a trajectory planner on top of this representation to be able to evaluate the effectiveness of the approach specifically with respect to a meaningful application. The comparison with a baseline method that only uses accumulated direct road detection information demonstrate the improvements in the spatio-temporal stability of the resulting trajectories.

KEY WORDS: Environment Representation, Road Detection, Trajectory Planning, ADAS, Autonomous Driving.

1. Introduction

The detection of road layout and semantics is an important issue in modern Advanced Driving Assistance Systems (ADAS) and autonomous driving systems. In particular, trajectory planning algorithms need a road representation to operate on: this representation has to be spatial, as the system needs to know exactly on which areas it is safe to drive, so that they can safely plan fine-grained maneuvers. Since typical trajectories are computed for timespans in the order of seconds[(3)], the spatial detection range needed for the road representation to achieve a stable and smooth trajectory is in the tenths to hundreds of meters. This range is very hard to achieve by using direct road detection[(18)], especially in inner-city, due to occlusions caused by various obstacles (e.g. buildings and high traffic) as well as hardware limitations. State-of-the-art systems cope with this problem by employing high-detail spatial road maps to complement direct detection[(21) (23) (27)]. However, maps are expensive to make and not available on every road. Another resource that can be used is commercially available navigational maps, but their topological nature is not ideal for planning concrete trajectories. Finally, ego-localization and map alignment is always an issue in the usage of any map: the accuracy of GPS can vary significantly in different environments, and errors of a few meters can be common with commercially available systems, especially in inner-city scenarios. A possible solution is to plan coarse trajectories that are meant to be corrected in the next seconds, when the area they traverse gets within direct detection range. This policy is not optimal, as it can cause sudden correction maneuvers, or it can force low speeds to increase the temporal horizon.

We presented a method to increase road detection range by complementing direct detection with a behavior-based indirect approach[(10)], showing that it can achieve better performances for road classification than direct detection alone can, in particular at long range. The method employs a spatial road representation derived from both direct and indirect road detection, where a probability of being road is computed for every patch of a regular square grid. Direct detection on monocular images is provided by "Road Terrain Detection System"[(14)] (RTDS), a feature-based detection system that provides road terrain confidence. Indirect detection is based on the interpretation of the other vehicles' behavior. Since our main assumption is that vehicles move always on road area, we estimate their past and future movements to infer the road layout where we cannot see it directly. The road confidences provided by RTDS and behavior interpretation are blended together by means of a function that gives different weights to the two sources, according to the position of the patch in the field of view and possible occlusions that would prevent the camera to see the patch.. We also showed that this representation can be expanded with the inclusion of additional semantics, such as lanes and driving directions[(9)].

In this paper we want to show that the enhancement provided by indirect detection is relevant for the purpose of trajectory planning. In order to do so, we present a system that couples our

Copyright © 2017 Society of Automotive Engineers of Japan, Inc. All rights reserved

representation with a basic trajectory planner to compute the ego-car trajectory on real-world data. The trajectory planner receives the current state of the ego-car (position, direction and speed) and the location of a predefined target area (from a navigational map). It computes a trajectory by creating a tree of possible future positions of the ego-car, each with a cost depending on the road probability of all the patches traversed to get there: the higher the probability, the lower the cost. Afterwards, it performs a variant of A* search ("Fringe Search"[6]) to find the path that leads to the target area with minimum total cost. We evaluate the trajectories computed with our representation comparing them to a baseline representation created by using only direct detection.

2. Related Work

Most state-of-the-art autonomous driving systems need road maps to plan their trajectories. For example, the participants to the 2011 DARPA Urban Challenge were given road maps of the full site by the organisers[3][21][23]. Similarly, BERTHA used rich maps that had been created offline which specified left and right boundaries of the roads[27]. The maps where in global coordinates, so a precise video-based self-localization was required. The trajectory was computed as the solution of a cost function minimization problem, where constraints were imposed by road boundaries and detected obstacles.

In any case, autonomous cars need an internal spatial representation of the environment to plan their trajectories on. The representation used by BERTHA exploited the notion of "lanelets" (atomic, interconnected road segments) to represent both topological and geometrical aspect of lanes[5]. Whereas this kind of representation allows to compute trajectories in the form of mathematical functions, other kinds of representations, especially if grid-based, are better suited to compute trajectories by tree search. A very popular type of grid-based representation is the occupancy grid, a spatial representation that describes the surroundings of the ego-vehicle in terms of free or occupied space. This representation can be computed by using different sensors, like lasers[8][24], radars[8], or stereo cameras[4], and can also include prior map knowledge[16] or theoretical models, e.g. stochastic[4] or particle-based[12]. Another possible spatial representation for drivable road that has been implemented is the probabilistic drivability map[17][22].

Focusing on the detection of road area, ADAS usually employ road detection systems. The most common and versatile sensor used for road detection is the camera. Road detection algorhitms can be divided into two categories, "marked" and "unmarked". Marked road detection infers road area by detecting lane markings and applying prior knowledge about road geometry[26]. It is most commonly used in simple and highly structured environments like highways. Unmarked road detection systems instead can infer road by observing low-level visual features, such as color, texture and elevation[11]. The road detection system that we use in this work is the SPRAY-based road detection system (RTDS)[14]. This method consists in two stages. The first is purely visual and extracts three base classifiers for the scene, while the second extracts spatial features (SPRAY) from the three base classifiers to classify road terrain as output. Researchers have also explored the possibility to realize hybrid approaches by combining low-level features with contextual cues (e.g. vanishing points and lane markings) and road priors obtained from navigational maps[2].

All these methods are considered direct detection because they rely on sensors that directly interact with the road. However, in the latest years researchers have started developing alternatives, indirect cues, to infer road area, due to the challenges that ADAS face in complex environments. The most promising indirect cue that has been investigated is the behavior of other entities in the environment. Initially their behavior was the target to be inferred, and the road was the cue. Behavioral assumptions have been exploited to improve the tracking of vehicles[1], and to predict their motion at intersections[19]. Information about context has been used to predict the behavior and classify the related danger of pedestrians[7] or vehicles[25]. The approach we present in this paper does the opposite: it exploits the behavior of vehicles to infer the context. In this direction, the behavior of pedestrians has been used to infer navigational maps for robots[20]. The behavior of vehicles has been used, together with lane markings detection and context models, to infer road geometry at long range in highways[13]: this work used a Bayesian inference algorithm to compute the parameters of a clothoid-based road model. Similarly, the behavior of vehicles has also been used together with other visual cues in order to infer the topological and geometrical layout of road intersections[15], out of a predefined set of possible topologies, using a proabilistic generative model. However, both of these latter systems used only the current behavior of vehicles, while the approach we present in the following section exploits also the predicted behavior.

3. System Overview

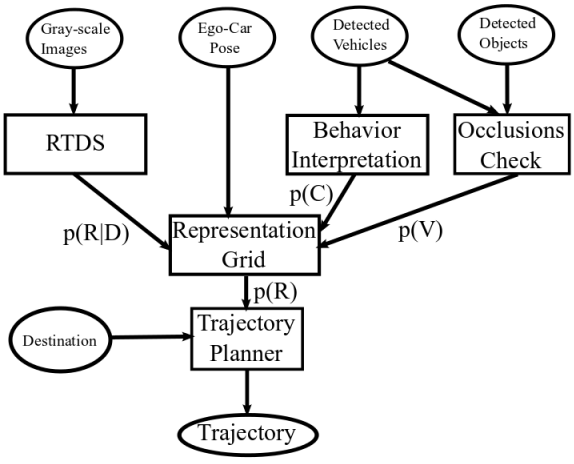

Fig. 1 Block diagram of the system architecture.

Our system can be divided into three stages. The first one is made by three independent blocks, that collect the inputs and compute the terms for the road probability formula. The second stage applies the formula and manages the representation grid, while the third stage is a trajectory planner that uses the representation created by the second stage. We already presented part of the concepts used by the first two stages[10], but some of formulation has been changed, so we will describe them in details in the following sections.

Copyright © 2017 Society of Automotive Engineers of Japan, Inc. All rights reserved

Our system has five different inputs: the first is a gray-scale image stream. The images are used as inputs for the RTDS module, which outputs a top-down view road confidence map. The second input is a list of detected vehicles and their position and velocity. The detected vehicles are used by the "Behavior Interpretation" module, which computes the past and future positions of each vehicle (see section 3.2), as well as by the occlusion check module. The latter also uses an additional input, the detected static objects, to compute the visibility of each patch (see section 3.4). The fourth input is the pose of the ego-car in world coordinates. The representation module uses it as a reference and computes the road probability for each patch (see section 3.1) and provides a road probability map to the trajectory planner (see section 3.5), which uses it to compute the trajectory to reach the destination (defined by user input). Figure 1 shows the block diagram of the system.

3.1. Road Formula

Our approach is based on the fusion between a camera-based direct detection system and a behavior-based indirect detection system. While the direct detection part provides reliable information for unoccluded roads, the behavior-based contribution helps in crowded scenarios, where direct detection is hampered. The main assumption of the behavior-based approach is simple: vehicles move on roads. When we observe a vehicle, we can assume that the area it is moving on must be road. Similarly, if we can estimate where a vehicle is headed, we can also infer the presence of a road there. The same concept can be extended to the past, as estimating where a vehicle comes from provides another cue for road area.
In order to create a probabilistic formulation of this concept, we define four events:
Event *R*: The event that the considered patch is part of the road. In the notation we use, $p(R_i)$ is the probability that patch i is road.
Event *C*: The event that the considered patch has been, is or will be traversed by at least one moving road vehicle. The time window in which *C* is considered is not necessarily relevant for our approach, but obviously $p(C_i)$ can be estimated only in a reasonably limited time window. This estimation is a key aspect of the approach, and will be discussed in detail in the next chapter. For simplicity, we assume that if a vehicle traverses a patch, then that patch is part of the road. In this case, we can state that $p(R_i|C_i) = 1, \forall i$.
Event *V*: The event that the considered patch is visible to the direct detection system. We assume that *V* is independent from *R*, so that $p(R_i \cap V_i) = p(R_i)p(V_i)$.
Event *D*: The event that the considered patch is correctly evaluated by the direct detection system. Since this can happen only if the patch is visible, we can consider $p(V_i|D_i) = 1, \forall i$. Conversely, $p(D_i|V_i)$ expresses the reliability of the direct detection system. Finally, $p(R_i|D_i)$ is the road confidence value provided by said system, and it is one of the inputs of our approach. For simplicity, we assume that *D* is independent from *C*, so that $p(D_i \cap C_i) = p(D_i)p(C_i)$.
Our goal is to obtain a formulation of $p(R_i)$ in terms of our inputs. In order to do so, we apply the law of total probability with respect to the event D, which leads to the following result:

$$p(R_i) = p(R_i \mid \overline{D_i}) + \left[p(R_i \mid D_i) - p(R_i \mid \overline{D_i})\right] p(D_i)$$

$$p(R_i \mid \overline{D_i}) = p(C_i) + p(R_i \mid \overline{C_i} \cap \overline{D_i})\, p(\overline{C_i})$$

$$p(D_i) = P(D_i \mid V_i)\, p(V_i)$$

Each term of the formula will be discussed in details in the following sections.

3.2. Position Estimation

Only one vehicle is necessary for event *C*, and in case of multiple vehicles j the event C_i is the union of the events $C_{i,j}$, representing the event that vehicle j traverses patch i. Thus, we can write:

$$C_i = \bigcup_j C_{i,j}$$

For simplicity, we assume that $C_{i,j}$ are all independent from each other. This is a strong assumption, since it is clear that there can be situations where the trajectories of two vehicles depend on each other (they want to avoid collisions), but since our event *C* ignores the time component of trajectories, this interaction is much less significant.
Since we treat them as independent events, their probabilities sum in the following fashion:

if $n = 2$

$$p_n(C_i) = p(C_{i,1}) + p(C_{i,2}) - p(C_{i,1})\, p(C_{i,2})$$

if $n > 2$

$$p_n(C_i) = p_{n-1}(C_i) + p(C_{i,n}) - p_{n-1}(C_i)\, p(C_{i,n})$$

We model the probability for a single vehicle with a 2D function, that provides a probability value given the relative location of the patch i in the reference system of vehicle j (*X* is the pitch axis and *Z* is the roll axis), and the speed of the vehicle itself. Even though we use our own function, any function that provides a probability for event $C_{i,j}$, as we defined it, can be used in this framework. Since we want to estimate both past and future positions of the vehicle, we use a symmetrical function with respect to both axis in the reference system of the vehicle. This function produces high values immediately in front of the car, and fades away with the distance to both axis. Every vehicle has a rectangular shape, with a certain direction and speed. If the detection was perfectly accurate our *p(C)* function should just have the shape of the vehicle itself, with a 100% probability on the patches covered. However, every kind of detection yields some form of error, and we chose to keep this probabilistic approach also under the car, so that we can model possible position inaccuracies.
The function we chose has the shape of a difference of two sigmoids. The standard deviation $\sigma_{i,j}$ varies along the *Z*-axis, but it is constant within the boundaries of the rectangle modeling the car. Beyond them, $\sigma_{i,j}$ increases progressively, in order to take into account the possible maneuvers of the vehicle. This increases depending on vehicle speed. The higher the speed, the slower the increase, since the vehicle will be less likely to turn at high speed.

$$p(C_{i,j}) = \left[erf\left(\frac{X_{i,j} + \eta_j}{\sqrt{2}\sigma_{i,j}} \right) - erf\left(\frac{X_{i,j} - \eta_j}{\sqrt{2}\sigma_{i,j}} \right) \right]$$

$$\sigma_{i,j} = \omega_j \qquad \text{if } \left| Z_{i,j} \right| \le \frac{L_j}{2}$$

Copyright © 2017 Society of Automotive Engineers of Japan, Inc. All rights reserved

$$\sigma_{i,j} = \omega_j + A \frac{Z_{i,j}^2}{\left\| \overrightarrow{v_j} \right\|^2} \qquad \text{if } \left| Z_{i,j} \right| > \frac{L_j}{2}$$

Here L_j is the length of the vehicle j, η_j is a parameter related to the width of it, ω_j is related to the uncertainty of its position, and A is a tuning factor. A contour plot of this function is shown in figure 2.

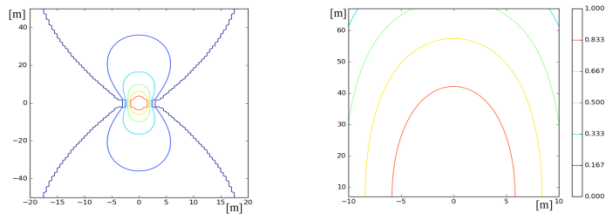

Fig. 2 [Left] Contour plot of $p(C_{i,j})$ for a vehicle positioned in [0, 0] and directed upward. [Right] Contour plot of $p(V_i)$

As the detected position and velocity of vehicles might change with every measurement, it is necessary to define a way to update the estimated $p(C_{i,j})$. For this purpose we distinguish two main categories: temporary and permanent estimations. Temporary estimations needs to be refreshed at every frame, as they depend on the current state of the observed vehicles. We consider the estimated future positions of vehicles as temporary. At every time step, the previous estimation is dropped and replaced with a new one.

Conversely, permanent estimations are those that depend on the state of the vehicle only at the instant they were computed, and thus they do not change with further measurements. Estimated past and present positions are permanent: they are computed only once, and will never change, because they do not depend on the successive behavior of their vehicle. In order to handle this difference, each patch of the representation grid holds two lists: the first list includes all the vehicles that have a temporary estimated probability to traverse the patch in the future. The second list includes all the vehicle that have a permanent probability to have traversed the patch.

When a vehicle disappears from the scene, its future $p(C_{i,j})$ does not disappear with it, but instead is moved to the permanent list. In fact, the disappearance of a vehicle is potentially a useful event: it means that it went to a road we cannot see, and therefore we want to save the last estimation we had of the area it was heading to. It will not change anymore, if we assume the vehicle has disappeared for good. If it does reappear (i.e. the disappearance was due to a misdetection), the system will be able to retrieve its $p(C_{i,j})$ and prevent inconsistencies. After some time, all the old permanent $p(C_{i,j})$ are merged into a single value, in order to avoid an unreasonable memory burden for the system.

3.3. Avoidance

The term $p(R_i|\bar{C}_i \cap \bar{D}_i)$ in the formula for $p(R_i)$ is a very important element of the road probability formula. It is the probability that patch i is road, in the case that none of the vehicles in the scene traverses it and the direct detection does not provide reliable information. This term could be interpreted simply as the prior probability of a patch being road, as other vehicles and direct detection do not provide information about it. It could be set to *0.5* if we do not know anything, or it could be set to the average road/not-road ratio in the environment we are in, if we have data about it.

For simplicity, we consider R_i to be independent from $\bar{D}_i$. Intuitively, the fact that the direct detection system does not classify correctly patch i does not provide any information about the road probability there. Thus, from here onward we will consider $(R_i|\bar{C}_i \cap \bar{D}_i) = p(R_i|\bar{C}_i)$.

The term $p(R_i|\bar{C}_i)$ does not have to be the same value for every patch i. In fact, there are many cases where it is evident it really should not. For example, consider a common roundabout. Vehicles approaching the roundabout would appear to be directed to the center of it, and this would lead the system to believe there is road at the center. However, all the vehicle will steer before entering that area. In this situation, the fact that the vehicles did not traverse the area they were initially predicted to drive on is a hint we can consider. The vehicles were headed there, but they actively avoided it. This behavior can suggest that said area is not road. Conversely, the fact that they did avoid an area they were never supposed to go to does not give as much information.

We can model this reasoning by defining $p(R_i|\bar{C}_i)$ as dependent on a previous value of $p(C_i)$, which we call $p^*(C_i)$. The higher $p^*(C_i)$ is, the lower $p(R_i|\bar{C}_i)$. In other terms, the likelier that a patch is being traversed, the likelier that it is not road in case it ends up being avoided. It has to be noted that this refers to the forward $p(C_i)$ only, as our reasoning involves only future positions.

This effect has to be mantained after the vehicle avoided the patch itself. In fact, after the maneuver, $p^*(C_i)$ would decrease at some point (since it is refreshed at every cycle, being a temporary estimation) and thus $p(R_i|\bar{C}_i)$ would increase again. Our solution is to make $p^*(C_i)$ equal to the maximum $p(C_i)$ recorded for that patch over time. This ensures that the effect is not lost. In our formula we use parameter k as a tuning factor.

$$p(R_i \mid \overline{C_i}) = P^0 \left[p^*(\overline{C_i}) \right]^k = P^0 \left[1 - \left[p^*(C_i) \right] \right]^k$$

$$p^*(C_i) = \max[\ p(C_i)]$$

The possible values of $p(R_i|\bar{C}_i)$ range from the prior probability P^0 (if no vehicles had a chance to be directed towards the patch) to an ideal lower limit of 0 (if at some point we had $p(C_i) = 1$).

3.4. Visibility and Direct Detection

In our road probability formula, $p(V_i)$ represents the chance that patch i is visible to the direct detection system. The system provides a road confidence output in a rectangular area in front of the vehicle, and by our definition every patch outside this area has $p(V_i) = 0$. Inside the area, the system checks for occlusions, caused by other vehicles or static obstacles detected by the sensors. If a patch lies behind an occlusion, its $p(V_i)$ is set to 0, while if it is not occluded we set $p(V_i) = 1$. The road probability formula uses the maximum value recorded, so that if a patch was visible at some point, the information is not lost when the patch gets occluded.

The reliability of direct detection, $p(D_i|V_i)$, is computed for every patch that is considered visible by the system. This term expresses the probability that the patch has been classified correctly by the direct detection. This term is not constant, because it is to be expected that the reliability of any direct detection algorithm decreases with distance[18], as well as with view angle. For this reason, we model $p(D_i|V_i)$ with a spatial function, that has higher values on the center of the rectangular field of view and in the vicinity of the ego-vehicle, while it has lower values on the sides and on the far edge of the FOV. The formula that assigns the visibility is:

Copyright © 2017 Society of Automotive Engineers of Japan, Inc. All rights reserved

$$p(D_i \mid V_i) = \cos\left(E\,\frac{X_{i,e}}{b}\right)\cos\left(F\,\frac{Z_{i,e}}{2l}\right)$$

Where the rectangle has a width of *2b* and a length of *2l*, $[X_{i,e}, Z_{i,e}]$ is the position of patch *i* in the reference system of the ego-vehicle, and *E*, *F* are two constant parameters. A contour plot of this function can be seen in figure 2. It is important to note that the domain of the visibility function does not start at $Z_i = 0$, as the camera field of view does not include the area immediately in front of the car. The road probability formula uses the average $p(D_i|V_i)$ over all frames where the patch was visible. The output of direct detection, $p(R_i|D_i)$, is averaged over time by using the respective values of $p(D_i|V_i)$ as weights.

3.5. Trajectory Planner

In order to show the potential of our system in terms of utility for trajectory planning, we set up a standard trajectory planner for grid representations that will use our representation to compute trajectories. The planner uses Fringe Search[6], a variant of A*, as pathfinding algorithm. This algorithm uses the same concept used by A* of estimating the remaining cost to reach the target from each node in order to select which nodes to expand first, but unlike A* it iterates over two lists, "now" and "later", which store respectively the current and next iteration. The planner goes through the nodes stored in "now", expands them and inserts their children in one of the two lists depending on their expected cost. If it is lower than a threshold, they get inserted in "now", while if it is higher they go into "later". Once the iteration over "now" ends because all the remaining nodes have an estimated cost higher than a threshold, the algorithm increases the threshold, moves all the nodes from "later" to "now", and moves to the next iteration. The main advantage of Fringe Search is that these lists do not need to be sorted, unlike what happens for A*, where sorting nodes can be very time-consuming.

The planner's goal is to compute the best path between the initial position A to any point in the target area. The target area has the shape of a 2D Gaussian function, where every patch has a score that will be taken into account by the planner. The higher the score, the lower is the overall cost of a path ending in that point. The center of the area is taken from an offline topological map that provides the world coordinates of the next intersection. The Gaussian shape of the target area is meant to model the uncertainty of the GPS localization.

The planner computes the optimal trajectory by performing a tree search, over nodes representing a certain spatial area. The tree is exemplified in Figure 3. At every node the vehicle reaches a location and can either turn left, right or go straight. Turning is modeled as increasing the yaw rate, so that the vehicle can turn more if it started turning on the previous node. The distance between a node and its parent depends on the speed of the ego-car (every node models the distance travelled by the ego-car during a fixed amount of time), which in turn is modeled as dependent on its yaw rate. While turning the ego-car slows down, as it happens in real-world maneuvers. Every node has five children: one that maintains the yawrate, two that increase it to the right (by 0.25 and 0.5 rad/s) and two that increase it to the left (by the same amounts).

The algorhitm assigns a cost to each node, that includes a road-dependent term and a yaw-rate-dependent one. The road cost depends on the road probability of all the *N* patches that the ego-car would traverse if it was directed there from the parent node.
The formula we chose is:

$$C_R(n) = \frac{1}{N}\sum_{i\in n}^{i=N}\left(\frac{1}{p(R_i)} - 1\right)$$

$$C_Y(n) = \left|\gamma(n)\right|$$

$$C(n) = JC_R(n) + KC_Y(n) + C(n') + Y(n)$$

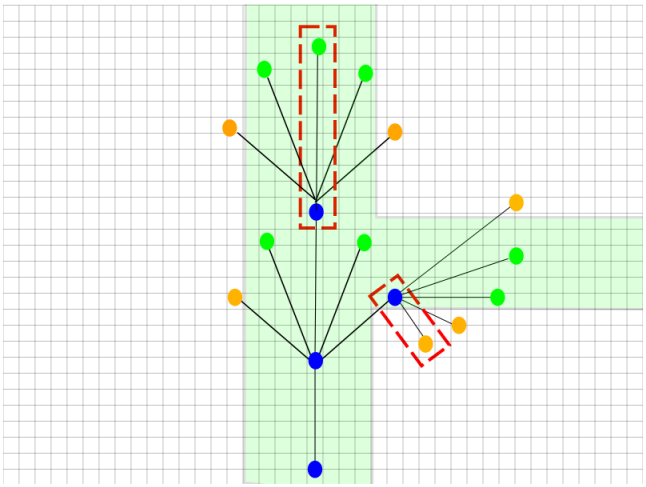

Fig. 3 Exemplary tree shape. The cost of each node is based on all the patches within the respective red area (the width of the rectangle is the same as the ego-car). In blue are the nodes that have been expanded. In green, the nodes in "now". In orange, the nodes in "later". The green area represents road.

Here $C_R(n)$ and $C_Y(n)$ are, respectively, the cost associated to road probability and yaw rate, $C(n')$ is the cost of its parent (without the estimated part), while *J* and *K* are tuning parameters. The estimated remaining cost $Y(n)$ to reach the target is approximated as the number of nodes (including *n* itself) required to arrive there at maximum speed. This is in fact the absolute best case scenario of the cost formula, so the actual cost cannot be less than that. In order to speed up the algorhitm, every node that has a cost higher than a predefined limit is flagged, and any of its children that also have a cost higher than the threshold are closed and marked as dead ends. The same happens if the distance to the center of target area increases for three consecutive nodes. When a path reaches a valid target, the iteration is completed and if there are multiple paths that end on a valid target the algorhitm sorts them based on their cost and the value of the target itself.

4. Experiments

Experiments have been run on real-world data, recorded by a mobile platform in the town of Offenbach am Main, Germany. The data consist of a video stream taken by a monocular camera, CAN data and traffic objects detected by a 360-degree IBEO Lidar setup. The Lidars provide position and velocity of detected vehicles in the scene, as well as the outline of any detected static object, which we use for our occlusion check, as explained in section 3.4.
Using a similar setup, we previously[9] compared our representation to hand-annotated ground truth, showing that it can classify road area reliably. Additionally, we showed significant improvements in terms of f1-score compared to a baseline representation based on direct detection (RTDS) only, especially at long range. Since comparing our representation to the simple RTDS output would not be fair, because it only uses a single

Copyright © 2017 Society of Automotive Engineers of Japan, Inc. All rights reserved

frame and does not keep knowledge of the past as our representation does, the baseline representation was created by using our system in a configuration that ignores vehicles. The result is a representation of RTDS output filtered over time based on the same concepts explained in section 3.4. We will use the same baseline in the following experiments. However, in this paper we evaluate the potential of our representation for trajectory planning purposes. We test if the increased range granted by the behavior-based enhancement produces a representation that is more precise and more stable over time with respect to our baseline, and consequently allows to plan more stable trajectories.

The following experiments all use short streams, where the two representations are computed and updated at every frame of the stream. We chose streams that were relevant in terms of showing differences between our representation and the baseline. As such, the streams have to show scenes where there are other cars on the road, as without them our representation is not different from the baseline by construction, where there are occluded roads that cannot be seen by direct detection, and where it is possible to plan a meaningful maneuver. Considering these terms, intersection in inner city, with buildings occluding the destination road, are the best choice and all the chosen streams present that kind of scenario. They all have vehicles moving from or towards that road, which is defined by the user as a target location which will be the center of the target area as explained in section 3.5. The location has been taken from commercially available navigational maps.

The planner computes the best trajectory based on each representation, and the trajectories can be compared using different measures. We focus only on the spatial aspect of these trajectories: the only dynamic property of the car that is taken into account is its speed, which is modeled implicitly, as it is dependent on its yaw rate. To evaluate stability over time, the most straightforward measurement is to compare the displacement of corresponding nodes over time. When we compare trajectories computed in different frames, it can happen that the initial position of the ego-car advances as much as to need one or more nodes less. In this case, we eliminate the first nodes of the longer trajectory to make them equal. However, we can also extract more condensed information from each trajectory. For example, we can define the "turning point" of a single trajectory as the location of the node that exhibits the highest yaw rate (in case there are multiple nodes with the same highest yaw rate, the turning point is the first one). We will measure both the location of the turning point and its displacement over time, to ensure that the trajectory is precise and stable. Another possible feature is the "end point", which we define as the last node of a trajectory. Since the destination is defined as an area, the actual end of the maneuver can vary, and it is desirable that it moves as little as possible, to ensure a stable maneuver.

4.1. Parameter Evaluation

In this section we compare the results of trajectory planning with variable cost parameters J and K (see section 3.5). The goal of this experiment is to find the optimal parameters in terms of stability over time of the resulting trajectories using on one hand our representation and on the other hand the baseline representation. It makes sense to find the best parameters for each of them separately, since in a real application the system would be optimized for its own representation only. In order to perform this evaluation, we set up a preliminary experiment. The system runs on a short stream with both representations computed at every frame. The trajectory planner computes the optimal trajectory to reach an area of the road to the right side of the intersection, at 4 specific frames that are separated by 0.5s from each other and starting 0.5s after the start of the stream itself. The planner uses 25 different parameters combinations (5 x 5), for a total of 200 computed trajectories. In order to identify the parameter set yielding the most stable results, we evaluated the obtained trajectories with three measures of stability. The first one is the standard deviation of the turning point location over the 4 computed frames of the stream. The second measure is the standard deviation of the end node, while the final measure is the average standard deviation of all corresponding nodes of the trajectories. Since all three measurements refer to a spatial displacement, establishing an importance priority between them would be difficult. We chose as the best parameter set the one that exhibits the lowest sum of all three measures (in meters). Table 1 shows the value for all sets.

K J	4	8	12	16	20		4	8	12	16	20
10	3.75	4.73	5.02	4.82	3.59		3.18	3.84	5.03	9.81	5.85
15	3.32	3.75	3.57	4.73	4.82		5.57	3.61	6.84	6.77	11.3
20	2.81	3.82	3.75	3.48	4.73		5.58	4.80	3.14	6.38	6.77
25	2.83	3.23	3.22	5.06	3.44		5.52	6.41	5.52	4.51	6.03
30	5.16	3.23	3.75	3.75	3.59		6.67	6.57	3.86	3.15	6.05

Table 1 Cumulative score of all parameters sets. On the left, results of our full system. On the right, results of the direct detection only configuration. The best score is marked in yellow.

4.2. Trajectory Evaluation

In this section we apply the trajectory planner to two streams using the parameters obtained in the previous section. The planner computes a trajectory on both representations obtained at 10 specific frames separated by 0.2s each. The computation of the representations started 0.5s before the first of these frames. The first stream (Figure 4) presents a scene where the ego-car starts roughly 40m from the intersection. The target area was centered at [-20, 38]m with respect to the initial reference system of the ego-car, lying on the leftmost lane of a wide two-lane road (one way). Figure 4 shows the location of the turning point in each computed trajectory. While the standard deviation is comparable in the two configurations (3.8m for the baseline, 4m for the full representation), the average location is very different ([-4.2, 33.1]m for the baseline, [-5.9, 40.2]m for the full representation). In Figure 6 we show an example of the computed trajectory for each representation in three different frames. Our full representation clearly show the destination lane, detected thanks to a vehicle traversing the intersection on it, and the planner correctly turns in the intersection. Conversely, the baseline representation cannot see the side road, and the planner turns very early, taking the shortest path in correspondence to an enlargement of the road in proximity of the intersection. The end point of both configurations is sufficiently stable, although it does show a slight advantage for our full representation (standard deviation of 0.9m against the 1.4m of the baseline configuration).

The second stream shows a similar scene, where the ego-car has to turn left to a target area centered in [-20, 35]m in the ego-car initial reference system,but this time the intersection is more complicated, since the target road is not perpendicular to the initial one. The intersection itself starts much closer to the ego-car than in the previous example, as it can be clearly seen in Figure 7 from the road enlargement detected by RTDS. Here the two configurations show significantly different results. Although the

Copyright © 2017 Society of Automotive Engineers of Japan, Inc. All rights reserved

location of the turning point shown in figure 8 differs less with respect to the previous example (average [-2.2, 26.9]m for the baseline, [-4.7, 31.4]m for the full representation), in this one the full representation allows for a very stable trajectory (standard deviation of 3.9m for the baseline, 2.4m for the full representation). The turning point does not change anymore after the third frame, while the baseline representation still produces unstable trajectories. The end point produced with the full representation is also much more stable than its counterpart, with a standard deviation of 0.53m against a baseline of 1.60m.



Fig. 4 [Intersection 1] Picture of the intersection as seen by the ego-car.

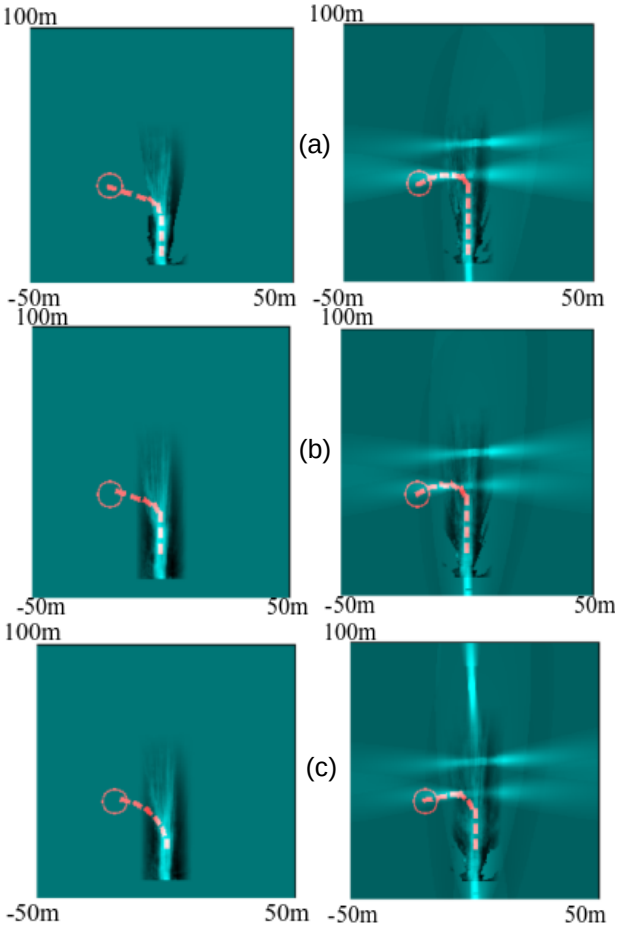

Fig. 5 [Intersection 1] Example of computed trajectory (red) on baseline representation (left) and full representation (right). The target area is marked as a red circle. Frames #2(a), #5(b), #8(c).

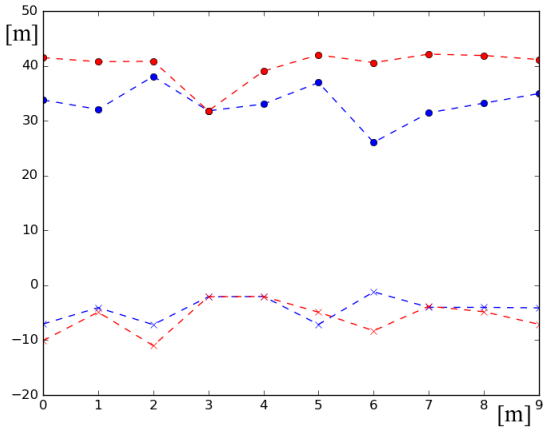

Fig. 6 [Intersection 1] Turning point location (x coordinate as 'x' markers, z coordinate as point markers) of the trajectories computed with the baseline representation (blue) and the full representation (red), over all 10 frames (from 0 to 9).

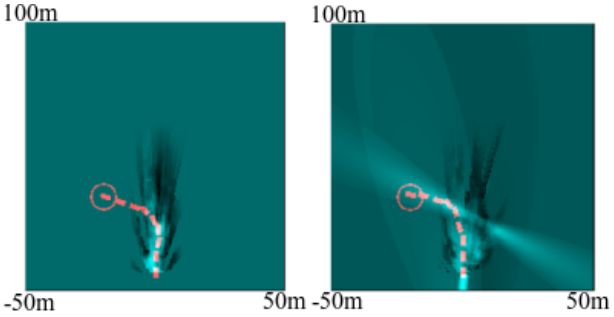

Fig. 7 [Intersection 2] Example of computed trajectory (red) on baseline representation (left) and full representation (right). The target area is marked as a red circle. Frame #5.

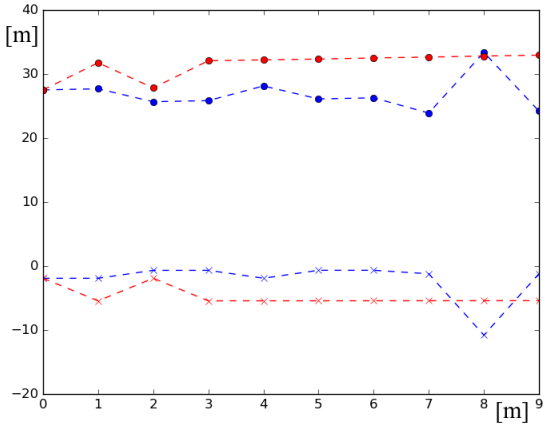

Fig. 8 [Intersection 2] Turning point location (x coordinate as 'x' markers, z coordinate as point markers) of the trajectories computed with the baseline representation (blue) and the full representation (red), over all 10 frames (from 0 to 9).

5. Conclusion

We presented an overview of a system computing a grid-based road representation created using both direct detection (in the form of a feature-based road detection system) and a behavior-

Copyright © 2017 Society of Automotive Engineers of Japan, Inc. All rights reserved

based indirect detection method. The system includes a custom trajectory planner computing spatial trajectories based on the aforementioned representation. We ran our system on real world data to evaluate the trajectories computed with our representation against those computed with a baseline representation created with only direct detection. We previously showed that our representation achieves a substantial increase in spatial detection range and quality of road detection[10], and here we showed that the additional information that it produces are relevant for the purpose of trajectory planning. In particular, the effect is greater in intersections where the destination is occluded. Experiments show the potential of our representation, in terms of allowing trajectories that are more spatially accurate and more stable over time respect to the baseline. Future works will need to expand the dataset and perform a full evaluation on a more complex trajectory planner, possibly including the dynamics of the ego-car. This result can potentially help autonomous driving systems to achieve a more human-like behavior, as their improved road awareness allows them to plan ahead, including areas they do not see yet.

References

(1) Alin, Butz, Fritsch: Tracking moving vehicles using an advanced grid-based Bayesian filter approach. IEEE IV 2011.
(2) Alvarez, Lopez, Gevers, Lumbreras: Combining Priors, Appearance, and Context for Road Detection. IEEE TITS 15(3), 2014.
(3) Bacha, Bauman, Faruque, ..., Webster: Odin: Team VictorTango's entry in the DARPA Urban Challenge. J. F. Robot. 2008.
(4) Badino, Franke, Mester: Free Space Computation Using Stochastic Occupancy Grids and Dynamic Programming. ICCV Work. Dyn. Vis. 2007.
(5) Bender, Ziegler, Stiller: Lanelets: Efficient map representation for autonomous driving. IEEE IV 2014.
(6) Björnsson , Enzenberger , Holte , Schaeffer . Fringe Search: Beating A* at Pathfinding on Game Maps. IEEE CIG 2005
(7) Bonnin, Weisswange, Kummert, Schmuedderich: Pedestrian Crossing Prediction using Multiple Contextbased Models. IEEE ITSC 2014.
(8) Bouzouraa, Hofmann: Fusion of occupancy grid mapping and model based object tracking for driver assistance systems using laser and radar sensors. IEEE IV 2010.
(9) Casapietra, Weisswange, Goerick, Kummert. Enriching a Spatial Road Representation with Lanes and Driving Directions, IEEE ITSC 2016.
(10) Casapietra, Weisswange, Goerick, Kummert: Inferring a grid-based road representation from the behavior of real world traffic participants. IEEE IV 2016.
(11) Caltagirone, Scheidegger, Svensson, Wahde: Fast LIDAR-based Road Detection Using Fully Convolutional Neural Networks. arXiv:1703.03613 [cs.CV]
(12) Danescu, Oniga, Nedevschi: Modeling and Tracking the Driving Environment With a Particle-Based Occupancy Grid. IEEE TITS 12(4), 2011.
(13) Fatemi, Hammarstrand, Svensson, Garcıa-Fernandez: Road Geometry Estimation Using a Precise Clothoid Road Model and Observations of Moving Vehicles. IEEE ITSC 2014.
(14) Fritsch, Kuehnl, Kummert: Monocular Road Terrain Detection by Combining Visual and Spatial Information. IEEE TITS 2014.
(15) Geiger, Lauer, Wojek, Stiller, Urtasun: 3D Traffic Scene Understanding from Movable Platforms. IEEE TPAMI 36(5), 2013.
(16) Gindele, Brechtel, Schroeder, Dillmann: Bayesian Occupancy Grid Filter for Dynamic Environments Using Prior Map Knowledge. IEEE IV 2009.
(17) Kang, Chung: Stereo-vision based free space and obstacle detection with structural and traversability analysis using probabilistic volume polar grid map. IEEE RAM 2011.
(18) Kuehnl, Kummert, Fritsch: Monocular Road Segmentation using Slow Feature Analysis. IEEE IV 2011.
(19) Liebner, Baumann, Klanner, Stiller: Driver intent inference at urban intersections using the intelligent driver model. IEEE IV 2012.
(20) O'Callaghan, Singh, Alempijevic, Ramos: Learning navigational maps by observing human motion patterns. IEEE ICRA 2011.
(21) Miller, Campbell, Huttenlocher, ..., Fujishima: Team Cornell's Skynet: Robust perception and planning in an urban environment. J. F. Robot. 2008.
(22) Sivaraman, Trivedi: Dynamic Probabilistic Drivability Maps for Lane Change and Merge Driver Assistance. IEEE ITSM 15(5), 2014.
(23) Urmson, Anhalt, Bagnell, ..., Ferguson: Autonomous driving in urban environments: Boss and the Urban Challenge. J. F. Robot. 2008.
(24) Weiss, Schiele, Dietmayer: Robust Driving Path Detection in Urban and Highway Scenarios Using a Laser Scanner and Online Occupancy Grids. IEEE IV 2007.
(25) Weisswange, Bolder, Fritsch, Hasler, Goerick: An integrated ADAS for assessing risky situations in urban driving. IEEE IV 2013.
(26) Weigel, Kramer, …, Saroldi: Accurate road geometry estimation for a safe speed application. IEEE IV 2006.
(27) Ziegler, Bender, Dang, Stiller: Trajectory Planning for BERTHA - a Local, Continuous Method. IEEE IV 2014.

Copyright © 2017 Society of Automotive Engineers of Japan, Inc. All rights reserved