2008 IEEE Intelligent Vehicles Symposium
Eindhoven University of Technology
Eindhoven, The Netherlands, June 4-6, 2008

Motion Segmentation and Scene Classification from 3D LIDAR Data

Dominik Steinhauser, Oliver Ruepp, and Darius Burschka
Department of Computer Science
Technische Universität München, Munich, Germany
dominik.steinhauser@mytum.de, {ruepp|burschka}@cs.tum.edu

***Abstract*— We propose a hierarchical data segmentation method from a 3D high-definition LIDAR laser scanner for cognitive scene analysis in context of outdoor vehicles. The proposed system abstracts the raw information from a parallel laser system (Velodyne system). It extracts essential information about drivable road segments in the vicinity of the vehicle and clusters the surrounding scene into point clouds representing static and dynamic objects which can attract the *attention* of the mission planning system.**

The system is validated on real data acquired from our experimental vehicle in urban, highway and cross-country scenarios.

I. INTRODUCTION AND MOTIVATION

Increasing density of urban traffic (Fig. 1) results in a high complexity of traffic patterns presented to the driver. Systems which support the driver in difficult decisions are becoming increasingly common in cars. They support the navigation of a vehicle detecting possible drive paths through a terrain and support the driver in recognizing dangerous situations.

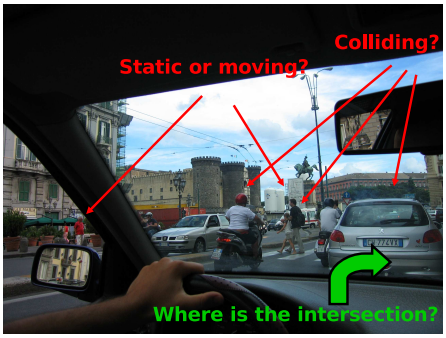

Fig. 1. A complex traffic situation.

The DARPA Grand Challenge was an important step to have autonomous ground vehicles that can navigate and drive across open and difficult terrain from city to city [4], [5]. Basic navigation tasks in an off-road or highway scenario can be solved solely based on the geometry of the terrain.

The next big leap will be an autonomous vehicle that can navigate and operate in urban traffic, a far more complex challenge for a 'robotic' driver. For such a system it is not sufficient to reconstruct the geometry of the terrain in order to avoid collisions, but it needs to *understand* the behavior of the objects in the surrounding world.

Fig. 2. 3D LIDAR scanner: (left) the experimental vehicle, (right) raw data from an urban scene.

A. Related Work

Some early work on laser data analysis for navigation is [11]. The method developed there is somewhat similar to the idea of occupancy grids [12], and thus has some limitations when the observed scenes are not planar. More recent approaches to using laser scanner data for navigation-related tasks have been described in, e.g., [9] or [10]. The algorithms developed therein mostly focus on detection and tracking of road boundaries, and in order to achieve this, rely on the presence of road curbs. Our approach is substantially different from that idea, because we simply try to identify areas that are "flat enough" for the car to drive on. Thus, we are also able to use the methods detailed herein for navigation on offroad terrain.

B. System Architecture

The laser range finder uses 64 lasers which radiate in different elevation angles, covering a total vertical range of approximately 25 degrees (Fig. 2). The lasers are also shifted horizontally by some degrees relative to the apparatus' rotational angle. To obtain data from the whole environment, the LIDAR scanner rotates at a speed of 10 Hz.

A data packet from the LIDAR system consists of the rotational angle of the scanner itself, the distance and the intensity measurement of each laser. From this data, a complete scan of the environment can be computed.

Our goal is now to use the information extracted from the LIDAR scanner in such a way that path planning is made possible. In our approach, we preprocess the data we get from the scanner to deskew distortion which is caused by the movement of the car during the scan. Then, a classification of the drivable road takes place. Afterwards, a clustering algorithm is run on the points which are not part

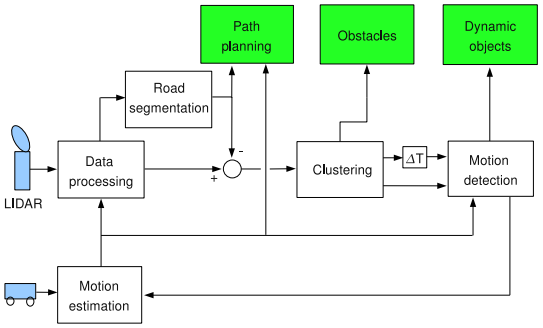

Fig. 3. The architecture of the system.

of the drivable surface. The resulting clusters represent the obstacles whose feature points are used as an input for the ego-motion computation algorithm. Two consecutive frames are processed and the ego-motion of the vehicle from one to the next frame is computed and non-stationary objects are detected. The ego-motion estimation, which can also be done by evaluating the IMU data, is now used to correct the next frame. A schematic overview of the process is shown in Figure 3.

II. APPROACH

A. Data processing

While our algorithms work mainly with Cartesian coordinates, the sensor readings deliver spherical point coordinates, so the first step is transforming that data into Cartesian space. The resulting set of 3D points which belong to a 360° scan is also called a frame.

We choose the Cartesian coordinate system such that its origin coincides with the center of the laser scanner, the x-axis points in forward driving direction, the z-axis points upwards, and the y-axis completes the axes such that they form a right-hand-system.

Denoting the elevation angles of a laser beam with θ, the rotational angle with ϕ and the distance measurement for that beam with r, the computation of the Cartesian coordinates is straightforward:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = r \begin{pmatrix} \cos(\theta)\cos(\phi) \\ \cos(\theta)(-\sin(\phi)) \\ \sin(\theta) \end{pmatrix}$$

Points which have been measured with only a low intensity are considered non-reliable and thus discarded. Knowing the position of the LIDAR scanner on the car, we can basically determine the point position relative to the car itself.

Due to the motion of the vehicle and the fact that the scanner takes a non-negligible amount of time to complete one rotation, the observed 3D point cloud will be distorted. Using information about the ego-motion of the car (see section II-D), we are able to level out the distortion. The resulting frame is an approximation of how the environment would have looked like if the car had not moved.

The scanner is rotating with a frequency of 10 Hz, so we can calculate the time elapsed in seconds for a data packet with rotational angle ϕ as $t = \frac{\phi}{20\pi}$. Furthermore, from the rotation matrix $\tilde{R}$ we can extract the roll, pitch and yaw angles α_{frame}, β_{frame} and γ_{frame} as described in [3].

To deskew the point positions, the rotation and translation for the time t of the point measurement have to be computed. The roll angle at a time t is approximated by linear interpolation as $\alpha(t) = 10t \cdot \alpha_{\text{frame}}$, the other angles $\beta(t)$ and $\gamma(t)$ are computed analogously.

From these angles, the rotation matrix $\tilde{R}(t)$ can be derived. To find the trajectory $\vec{x}(t)$ of the car, we proceed as follows: We assume that the car moves in x-direction (relative to the car's coordinate frame), so the rotated coordinate system's base vector

$$\tilde{R}(t) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos(\alpha(t))\cos(\beta(t)) \\ \sin(\alpha(t))\cos(\beta(t)) \\ -\sin(\beta(t)) \end{pmatrix}$$

is the tangent-vector to the trajectory $\vec{x}(t)$. The speed of the car is then described as

$$\vec{v}(t) = v_{\text{car}} \cdot \tilde{R}(t) \cdot (1,0,0)^T$$

And finally, the trajectory can be computed by integration:

$$\vec{x}(t) = \int \vec{v}(t)\, dt$$

After evaluating this integral, there are two unknowns left, namely v_{car} and the integration constant. The two constraints of the trajectory $\vec{x}(0) = \vec{0}$ and $\vec{x}(0.1) = \vec{T}$ can be used to solve for these two unknowns. Now the trajectory $\vec{x}(t)$ can be used to calculate the translation vector $\vec{T}_i(t)$.

Having computed rotation and translation for each data packet, the undistorted coordinates p'_{ij} of a point p_{ij} in the can be calculated as $p'_{ij} = \tilde{R}(t)(p_{ij} - \vec{T}(t))$ where p_{ij} denotes the j-th point of the i-th data packet.

B. Drivable road estimation

After computing the undistorted frame, we estimate the drivable surface. Obviously, this is useful for deciding where the car can go safely without crashing into obstacles.

But the information is also required for our clustering procedure: Having a 3D point cloud from a laser range finder, all objects that are located on the road and the road itself seem to form one connected structure. If there are, e.g., two trees on opposite sides of the road, they should obviously be detected as seperate objects. But because there is a stretch of road between them, the trees and the road would seem to belong to one connected component. By removing the road from the scene, the connected structure is broken up into the single obstacles of interest.

The algorithm used to estimate the drivable road classifies each point of the frame. A point can therefore be marked as a surface point or an obstacle point. Surface points can further be refined in flat surface points and critical surface points, which belong to the supporting plane below the vehicle, but not to the road surface. This is important because we need to

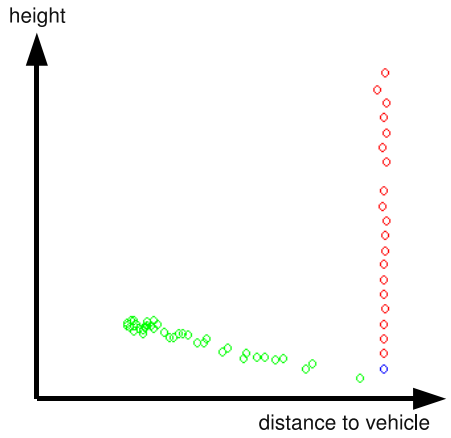

Fig. 4. Profile plot, points are classified into obstacle (red) and surface (green) points. Blue points are drivable road, lying behind obstacles. The height axis in the picture corresponds to the z-axis of the car's coordinate-system, the distance axis points in a certain rotational direction.

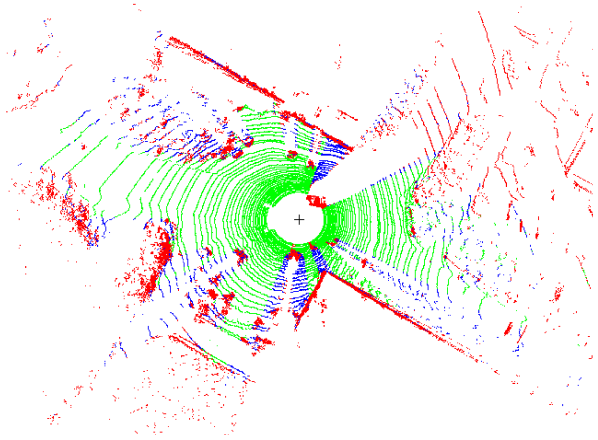

Fig. 5. Top down view on a frame, points are classified into obstacle (red) and surface (green) points, blue points are drivable road behind an obstacle. The + symbol in the middle marks the scanner's position.

know all of the points that belong to the supporting plane in order to be able to decompose the scene into seperate point clusters. The criteria for marking a point is the distance of the point to the estimated road. If the distance is below a certain threshold, the point is marked as a surface point. Furthermore, the point is marked as a critical surface point if the distance exceeds the threshold but lies under a second threshold, which is chosen depending on the car's offroad abilities. Otherwise the point is marked as an obstacle point. Moreover, the z-coordinate of the obstacle points are checked to refine these points. If the value lies above the car's height, the related obstacle point doesn't affect the cars ability to move but is of interest when clustering the obstacles.

The classification algorithm does the calculation for each set of points of a certain rotational angle. The algorithm considers the 64 points from the 64 different lasers pointing in some direction, originating from the frame's center, which is the center of the LIDAR scanner system.

So as input for the algorithm we are given a point set $\{p_i\}; i = 1, 2, ..., 64$. These points all lie on a plane that can be thought of as a profile cut from the point cloud (see Figure 4). The coordinates of the points in the plane are calculated as $p_{ix_{\text{plane}}} = \sqrt{p_{ix}^2 + p_{iy}^2}$ and $p_{iy_{\text{plane}}} = p_{iz}$. This set of points is sorted in ascending x-component direction.

In this set of points, we now search for line segments. Beginning with the first point p_1 in the set, a certain number of consecutive points is used to do a least squares linear regression for finding a model for a line segment. The parameters a and b for the line segment $y = ax + b$ are computed as

$$a = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \quad b = \bar{y} - b\bar{x}$$

where $\bar{x}$ and $\bar{y}$ denotes the mean value of the points x and y component.

The obtained line model has to fulfill several properties to be accepted as a representation for the surface: The gradient must not exceed a certain value, i.e., the line must not be too steep. Then also the quality of the line fitting is examined, which should not fall below a certain threshold. If the line passed these initial tests, subsequent points of the set will be examined and added to the line if they do not deviate too much from it. Whether a point should belong to the line or not is determined by computing its distance to the line. This process will be repeated until the line does not fulfill the above mentioned criteria any more or a new included point does not belong to the line model. The index of the last point of a segment will be stored. Then the algorithm starts again for the next line segment, beginning with the last point used in the previous run, and is repeated until the last point of the set is reached.

As result we get a set of line segments and the index of the starting point for each segment. For classification of the points, we compare the line segments to the corresponding point sets, from the marked beginning point up to the marked point of the next line segment. There are a few thresholds for the distance from the points to the line which indicate whether a point is a surface point, a critical surface point or an obstacle. If no lines were found, all points of the set are considered obstacle points.

Furthermore, the surface points that lie behind an obstacle (viewed from the center of the LIDAR scanner) are classified separately. These surface points are not used for finding drivable areas for the car, but they should obviously be excluded from the clustering procedure.

All in all, we obtain a matrix with the classification of each point of the frame, where the indices of each entry correspond to the same indices of the corresponding point in the data array mentioned in section II-A. A top down view shows the computed drivable road and the obstacle points in figure 5.

C. Clustering

After estimating the drivable road and removing the points classified as passable environment, the clustering of the

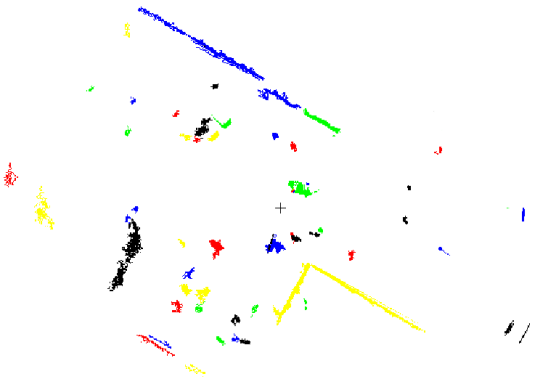

Fig. 6. Top down view on a frame, different colors indicate different clusters.

obstacles can be done. The criterion for deciding whether points belong to the same cluster is simply the Euclidean distance between those points.

As described in section II-A, the points are stored in a data array. If this array is interpreted as a gray scale image, we obtain a 360 degree picture of range information of the environment. In this representation, we would intuitively expect that all points of a cluster are neighboring pixels or at least quite close to each other.

Hence, we can do the clustering by using an approach similar to a region-growing-algorithm instead of exhaustively comparing points pairwise to check whether they belong to the cluster or not.

The algorithm works as follows. The data array is read line-wise starting from the index (0,0) of the data array. Note that this is the nearest point to the car at rotation angle zero degree. We assume that around the car and especially in front of the car, there will be usable road and therefore we start at this point.

When a not yet visited obstacle point is found, we start a new cluster, and search for points belonging to the cluster in the neighborhood of the obstacle point. In a first step, only the four-point-neighborhood is examined. As criterion for adding a point to the cluster, the Euclidean distance from the current point to the point in consideration is used. If the point has been classified as belonging to the cluster, the search continues with this point in a recursive manner. On the other hand, if none of the neighboring points belong to the cluster, we will not abort the search yet. Instead, a certain amount of points lying in the current search direction are checked for whether they can be added to the cluster, according to the Euclidean distance of the previously added point. This step is necessary for several reasons: sometimes, it is possible that the laser range finder fails to deliver measurements. Also, it is possible that the laser looks through an object, e.g. the wheel of a bike, so the cluster would be divided by parts of the estimated usable road.

A result of the clustering algorithm can be seen in figure 6, a top down view of the same frame used in the previous sections, different colors have been used to distinguish clusters.

D. Ego-motion detection and estimation

The basic idea of our ego-motion estimation is as follows: If we knew the movement of the stationary objects in the scene relative to our vehicle, we would obviously be able to deduce the ego-motion of the car itself. To accomplish this, there are two problems that need to be solved: First of all, we need some way to identify feature points and establish correspondences between these points in consecutive frames. When correspondences have been found, we still need to classify feature points as moving or non-moving.

Currently, we are using two different types of feature points: The centroids of point clusters, and intersections between planes in the scene. Computation of the centroids is straightforward, but using these simple features for the motion estimation introduces errors. Extraction of planes from the scene is more complicated and time-consuming, but yields very stable features.

The errors when using cluster centroids can be explained as follows: The underlying assumption made is that a centroid is stationary if the corresponding object is stationary. But that is not realistic: Under different viewpoints, different portions of the object will be visible to the laser, and the centroid is computed based only on the visible portion. This means that the centroid might move if the viewpoint changes.

When feature points and correspondences have been established, a RANSAC [2] algorithm is used to classify points as moving or non-moving. The model used in the RANSAC approach is a least-squares fitting of 3D-point sets suggested in [1] to estimate the motion from one to the next frame.

1) Corresponding feature points: Computation of the cluster centroids is straightforward. As mentioned above, errors are introduced when using this simple technique, but in practice, it turns out the error is mostly tolerable. A more advanced technique is the identification of planar surfaces in the scene, and then using the intersection points of those planes as landmarks. This yields very stable features, although of course it only works when enough planes (at least three, and all non-parallel) are present in the scene.

Planes are extracted using an approach similar to that explained in section II-B, adapted for searching planes instead of line segments. Not necessarily all points of the cluster belong to a plane, so a RANSAC approach is used to choose the points belonging to a plane without including outliers.

A plane is represented by its normal vector and the centroid of the points. To compute these parameters, the approach described in [8] is used, which is just a standard method for fitting a plane to a point of clouds. The average error used in the RANSAC algorithm for identifying outliers is the average distance of the points to the plane.

After the planes have been found, we still need to compute feature points, which are derived by computing intersection points between planes. Generally, three pairwise non-parallel planes will have exactly one intersection point.

The correspondences between features identified in two successive frames are found by comparing the Euclidean distances. For a point in the landmark set of the first frame, we assume that the closest point in the landmark set of the

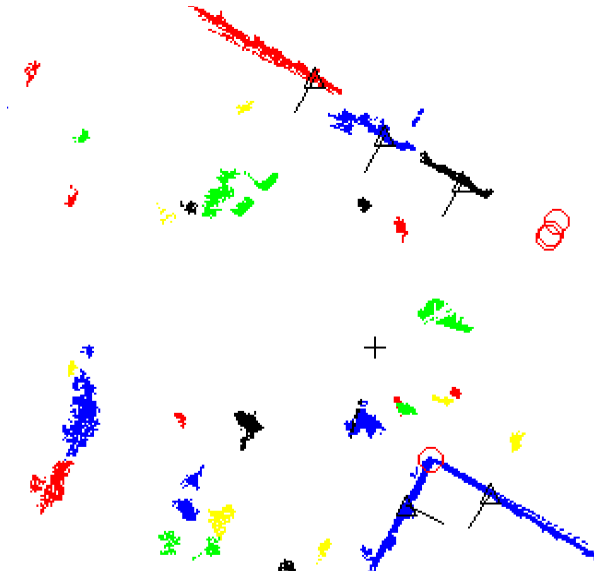

Fig. 7. The estimated planes in a frame. The triangle indicates the centroid of the points, the thin line the normal vector of the plane. The red circles are the intersection points.

repeat N times

- choose a random subset of corresponding points and calculate $\tilde{R}$ and $\vec{T}$ as described in section II-D.3
- compute the consensus set by applying the calculated rotation and translation to all points of O_{frame1} and then computing the distance to the corresponding points of set O_{frame2}, add points whose distance is below a certain threshold
- save the consensus set if its size is bigger than a certain number

end

If at least one consensus set is found, use the one with the most elements and calculate from these points the rotation $\tilde{R}$ and translation $\vec{T}$. If no consensus set is found, the algorithm fails.

Fig. 8. The RANSAC algorithm for finding static objects and doing ego-motion estimation.

second frame is its corresponding point. The resulting set of point corresponences is used as input for the RANSAC algorithm.

2) RANSAC: An overview of the algorithm is shown in figure 8. We denote with O_{frame1} and O_{frame2} two point sets containing features of two consecutive frames.

The computed rotation $\tilde{R}$ and translation $\vec{T}$ are applied to all corresponding points of the set O_{frame1} (see II-D.1). If the distance of the transformed points to the corresponding points in O_{frame2} is below a threshold, the cluster belonging to these points is classified as a static.

All other points in O_{frame1} that have a corresponding point in O_{frame2} belong to moving objects. Feature points without corresponding points in the other set can obviously not be classified.

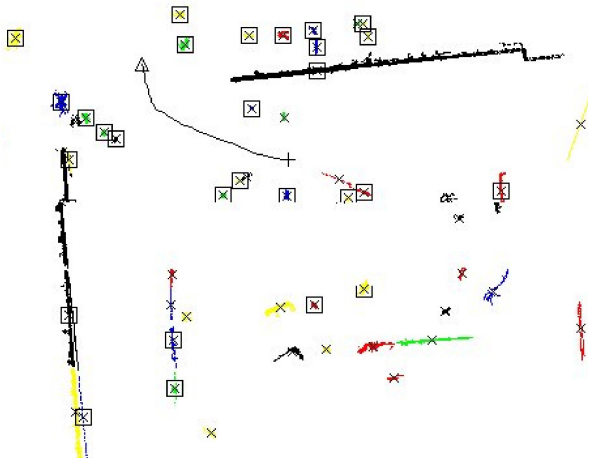

Fig. 9. The trajectory of the car, drawn in the current frame.

TABLE I

PROCESSING TIMES OF THE SYSTEM'S COMPONENTS.

data processing	0.55s
drivable road estimation	0,49s
clustering	0,09s
motion estimation	0,015s

3) Computing rotation and translation: The rotation $\tilde{R}$ and translation $\vec{T}$ between point of two consecutive frames is computed with the algorithm introduced in [1]. The algorithm involves the following steps:

1) Compute centroids p and p' of the two point sets.
2) Shift the points such that point sets $q_i = p_i - p$ and $q'_i = p'_i - p'$ are established.
3) Compute a 3x3 covariance matrix $H = \sum_{i=1}^{N} q_i q_i'^T$.
4) Using a singular value decomposition $H = U\Lambda V^T$, the rotation matrix will be $\tilde{R} = VU^T$ and the translation is given as $\vec{T} = p' - \tilde{R}p$.

Figure 9 shows an example run. Clusters are shown with different colors and the centroids of the clusters marked with crosses. Centroids marked with a rectangle indicate that these centroids are static objects and can be used for calculation of the ego-motion. The triangle indicates the start of the route. The black line represents the estimated trajectory of the car.

III. RESULTS

Data has been recorded using a 3D laser range finder, a high definition LIDAR system from Velodyne[1]. The scanner is mounted on top of an experimental vehicle, a VW Touareg, as pictured in Figure 2. Movement data is available from an Inertial Measurement Unit (IMU) which is also mounted on the car. The system was tested on an Intel CoreDuo@2GHz laptop computer running a Linux operating system. Computing times for each component of the system are shown in table I.

[1]www.velodyne.com/lidar

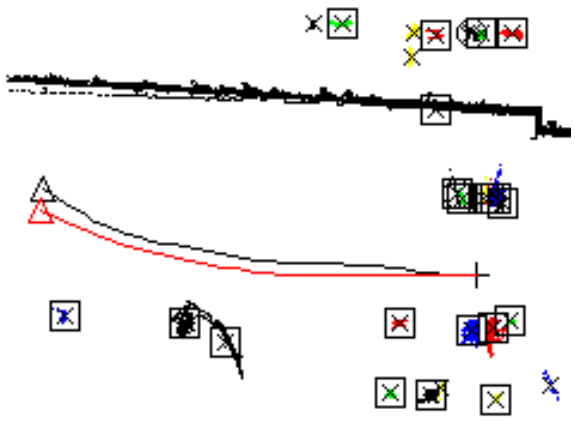

Fig. 10. The trajectory of the car from the motion estimation (black) and the car's IMU (red).

We tested the system with data from several trial runs on the university campus and some runs on forest tracks. The drivable road estimation generally gave good results, also on difficult surfaces like untarred roads. Figure 10 shows an example of a trajectory (black line) generated from a tour on the campus of the university. The triangles are placed at the respective estimated starting points, relative to the current reference frame. For comparison, the trajectory generated from the car's IMU data is shown as a red line. The distance covered is approximately 40 meters.

IV. CONCLUSION AND FUTURE WORK

In this work we have shown how the raw data of a 3D high definition LIDAR laser range finder can be used to classify the immediate environment of the vehicle into drivable road and obstacles. By using the static objects, ego-motion estimation is done, which can be used to deskew the data from the LIDAR scanner, distorted due to the vehicle's motion. Afterwards, obstacles can be classified into static and moving objects.

In actual applications, we want the system to run in real time for the LIDAR scanner's frequency of 10 Hz, so we have to improve on the running times of parts of our algorithms. Furthermore, we have to be able to cope with laser failures, because it happens that the LIDAR scanner fails for a range of about 10 degrees, so clusters are possibly divided in this area. Two problems arise in this case: No correspondences using centroids of clusters can be found, or a static object could be classified as a moving object. This fact will have to be taken into account when classifying objects into static and non-static, as described in section II-D.

For now, we only use cluster centroids and plane intersections as landmarks, which results in a notable error in the ego-motion estimation. To improve on that situation, it would certainly help to develop algorithms for finding different and hopefully more reliable static landmarks.

In the motion estimation, we assume that static objects like buildings or trees are present and distinguishable at all times. This is not always a valid assumption: On a highway with several lanes, e.g., where the vehicle is surrounded by moving cars and not many static land marks are seen, it seems likely that the RANSAC algorithm will fail. It is also difficult to drive on forest tracks, where the road is surrounded by dense trees and bushes that cannot be distinguished from each other easily, and thus may appear as just one big connected cluster to our system, which makes it very difficult to extract useful feature points.

Furthermore, it would be nice to be able to track the dynamic objects, estimating their trajectories or even deformations. Object recognition would also be an extremely useful feature, allowing for classification of clusters as, e.g., other moving cars, pedestrians, trees etc.

V. ACKNOWLEDGMENTS

This work was supported by the Cluster of Excellence for Cognitive Technical Systems (CoTeSys). We want to thank our partners from the University of the Federal Armed Forces in Munich for their support and for providing us with test data from their experimental vehicle.

REFERENCES

[1] K.S. Arun, T.S. Huang and S.D. Blostein, "Least square fitting of two 3-D point sets", *IEEE transactions of pattern analysis and machine intelligence*, 1987

[2] Martin A. Fischler and Robert C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography", *Communications of the ACM*, June 1981

[3] John J. Craig, "Introduction to Robotics, Third edition", *Pearson Prentice Hall*, 2005

[4] Sebastian Thrun, "Stanley: The Robot that won the DARPA Grand Challenge", *Journal of field robotics*, Stanford University 2005

[5] Michael Montemerlo, Sebastian Thrun, Hendrik Dahlkamp and David Stavens, "Winning the DARPA Grand Challenge with an AI robot", Stanford University 2005

[6] David M. Cole, Alastair R. Harrison and Paul M. Newman, "Using naturally salient regions for SLAM with 3D laser datas", Oxford University UK

[7] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald and Werner Stuetzle, "Surface Reconstruction from Unorganized Points", University of Washington, Seattle

[8] Craig M. Shakarji, "Least-Squares Fitting Algorithms of the NIST Algorithm Testing System", *Journal of Research of the National Institute of Standards and Technology* 1998

[9] K. Kodagoda, C.-C. Wang, and G. Dissanayake, "Laser-Based Sensing on Roads", In IEEE Intelligent Vehicles Symposium (IV2003), June, 2003

[10] W.S. Wijesoma, K.R.S. Kodagoda, A.P. Balasuriya, "Road-boundary detection and tracking using ladar sensing," Robotics and Automation, IEEE Transactions on , vol.20, no.3, pp. 456-464, June 2004

[11] M. Hebert and T. Kanade, "3-D Vision for Outdoor Navigation by an Autonomous Vehicle," Proceedings of the 1988 DARPA Image Understanding Workshop, April, 1988, pp. 593-601

[12] Thrun, S. 2003, "Learning Occupancy Grid Maps with Forward Sensor Models," Auton. Robots 15, 2 (Sep. 2003), 111-127