Real-time 3D Mapping of Rough Terrain: A Field Report from Disaster City

Johannes Pellenz
Bundeswehr Technical Center for Engineer and General Field Equipment
Universtitätstr. 5
56070 Koblenz, Germany
JohannesPellenz@bwb.org

Dagmar Lang, Frank Neuhaus and Dietrich Paulus
Active Vision Group
University of Koblenz-Landau
Universitätstr. 1
56070 Koblenz, Germany
dagmarlang,fneuhaus,paulus@uni-koblenz.de

Abstract — Mobile systems for mapping and terrain classification are often tested on datasets of intact environments only. The behavior of the algorithms in unstructured environments is mostly unknown. In safety, security and rescue environments, the robots have to handle much rougher terrain. Therefore, there is a need for 3D test data that also contains disaster scenarios. During the Response Robot Evaluation Exercise in March 2010 in Disaster City, College Station, Texas (USA), a comprehensive dataset was recorded containing the data of a 3D laser range finder, a GPS receiver, an IMU and a color camera. We tested our algorithms (for terrain classification and 3D mapping) with the dataset, and will make the data available to give other researchers the chance to do the same. We believe that this captured data of this well known location provides a valuable dataset for the USAR robotics community, increasing chances of getting more comparable results.

Keywords: Disaster City, 3D mapping, terrain classification

Introduction

The use of autonomous robots is nowadays often restricted to structured, well known indoor or outdoor environments. After an earthquake or other natural or man-made disasters, the world changes dramatically, and systems that might have worked in an intact world fail. Such scenarios are simulated in Disaster City [11], which is located in College Station, Texas (USA). Disaster City is a 52-acre training facility for USAR teams, but it is also used for the annual NIST Response Robot Evaluation Exercise. It features several full-scale collapsible structures simulating various levels of disaster and wreckage. For example, it provides collapsed buildings, rubble piles as well as car and train wreckage.

During the Response Robot Evaluation Exercise in March 2010, we collected 3D laser scan data (from a Velodyne HDL-64E S2) of the entire site, along with the data from an Garmin GPS receiver, an Xsens IMU device and a low-cost (non calibrated) webcam color camera. To give other researchers the opportunity to work with this unique dataset, we will make the sensor data available on the Robotic 3D Scan Repository[14].

We used the collected sensor data to evaluate our terrain classification algorithm (that can distinguish between flat ground, rough terrain and obstacles) and two common methods for 3D map generation: the standard “plain vanilla” iterative closest point (ICP) approach and a 6D SLAM approach [6]. To be able to run the 3D mapping in real-time, the terrain classification was used to extract systematically feature-rich regions from the environment.

The state of the art of strictly autonomously navigating systems was demonstrated at the DARPA Grand Challenge in 2004 and 2005, as well as in the DARPA Urban Challenge in 2007 [9]. In both Grand Challenges, the participants mostly acquired terrain drivability information using cameras, two-dimensional laser range finders, or a combination of both [9]. The advent of three-dimensional laser range finders such as the Velodyne HDL-64E S21 in the Urban Challenge yielded a much richer, thorough picture of the environment.

Several approaches were developed to solve the simultaneous localization and mapping (SLAM) problem in 2D for structured environments while complex 3D environments are an active field of research. In the resulting 3D maps, complex environments such as trees, bridges or underpasses can be represented. Furthermore, 3D mapping provides the basis for a better localization of the robot compared to methods using 2D maps.

The remainder of this paper is organized as follows: In Section II, we briefly describe the contents of the acquired dataset. Section III provides an overview of the robot and the used sensors. In Section IV we briefly introduce our terrain classification used for the drivability analysis and the data reduction. Section V gives a review of the approaches used in this field test. After describing the experiments in Section VI the results are presented in Section VII.

Dataset

The captured dataset covers a travel of about 15 minutes through Disaster City, with a speed of about 7 km/h. During this time, the Velodyne 3D LRF data was captured with 10 Hz (9169 records), the IMU data with 100 Hz (95936 records), the GPS data with 1 Hz (954 records) and the camera data with 30 Hz (26242 images). The file has an overall size of 4.7 GB; each record carries a timestamp. The dataset features several loops around Disaster City (on roads and through rough terrain...
The hierarchical PCA. Finally, an approach to analyze the grid-based PCA are introduced, followed by our extension, component analysis (PCA). In the following, the PCA and the almost zero-radius turns for advanced maneuverability. For the side of the vehicle. No odometry of the robot was available. The IMU was attached to the dashboard on the right of the robot, where also the antenna of the GPS receiver was installed. The V elodyne 3D sensor was mounted on the roof of the robot, where also the antenna of the GPS receiver was installed. The IMU was attached to the dashboard on the right side of the vehicle. No odometry of the robot was available.

III. ROBOT

The sensors were mounted on a robot from the MITRE Corporation called Centaur [12]. The Centaur has caterpillar-like tracks and is skid-steered, which allows the robot to do almost zero-radius turns for advanced maneuverability. For the data capturing, the robot was manually controlled by a human driver. The Velodyne 3D sensor was mounted on the roof of the robot, where also the antenna of the GPS receiver was installed. The IMU was attached to the dashboard on the right side of the vehicle. No odometry of the robot was available.

IV. TERRAIN CLASSIFICATION USING 3D LRF DATA

The analysis of the 3D point cloud is based on the principal component analysis (PCA). In the following, the PCA and the grid-based PCA are introduced, followed by our extension, the hierarchical PCA. Finally, an approach to analyze the roughness of the ground using the distribution of 3D laser distances is presented.

A. PCA

The PCA of the covariance matrix of a three-dimensional point cloud (which is in this specific case in fact equivalent to the eigendecomposition) yields three eigenvectors with corresponding eigenvalues. The eigenvalues indicate the variance of the point cloud along the corresponding axes; they can be used to determine the general shape of the point cloud.

The standard PCA method only yields good results for reasonably small point clouds. Those containing multiple planes/cylinders or other complex structures simply end up in the case where all three eigenvalues are large. In order to gain local information about a large point cloud (such as the one delivered by the 3D laser scanner), the data has to be subdivided into smaller chunks. There is a multitude of possibilities to subdivide the point cloud. We use a simple 2D grid, which is centered around the origin of the sensor. Other researchers such as [2] used 3D grids. The problem with these uniformly sized cloud-fragments is that the ever increasing spread field of view of the sensor makes information about far-away ground surfaces sparse. Selecting a small cell-size for the grid results in many distant cells being completely empty, simply because not a single laser beam endpoint happens to be inside this cell. Selecting a large cell size allows larger regions to be considered in the PCA step. However, this causes problems in regions close to the sensor: If a small obstacle is inside a cell, the whole cell is classified as occupied. Subsequent path planning algorithms operating exclusively on this grid would have to keep significantly more distance to obstacles than really needed. In some cases, such as narrow passages, where precise navigation is required, this methodology can even impede path-planning altogether, simply because almost everything is marked as impassable. What is really needed is an algorithm that combines the advantages of both, small and large cells.

B. Hierarchical PCA

Our hierarchical PCA algorithm recursively subdivides the point cloud. We use it on a 2D grid, however the extension to 3D is straightforward. The cell size is chosen to be relatively small (0.5 m × 0.5 m). The initial input of the algorithm is the whole grid. In each step, a PCA analysis on the points in the considered region is performed. If the point cloud contained in the region is not flat enough, the algorithm splits the region into two pieces and recursively calls itself on the two resulting fragments. The splitting simply occurs at the middle of the longest edge of the region. If the input region can no longer be subdivided, and it is still not flat enough, it is considered an obstacle. To quantify flatness, we first define the local height disturbance $h$ as:

$$h = \lambda_k \quad \text{with} \quad k = \arg\max_{i \in \{0,1,2\}} |e_i^T z|$$

where $z$ is a vector pointing up, $e_i$ are the eigenvectors, and $\lambda_i$ the eigenvalues of the point cloud in the current region.
A region is considered flat, if $h$ is below an experimentally
determined threshold $t$ (for example $80 \times 80$). This definition
makes sure, that even sloped but flat areas get low values of $h$.
However, slopes (with angle $\alpha$) that are steeper than 15°
are also considered as obstacles.

C. Roughness analysis

Even though a region may technically be drivable, it may
still be desirable to prefer one region over another. An example
of this is when the robot is driving on a track and next to this
track a still passable rubble pile is located. Obviously, one
expects the robot to continue its way on the track instead of
driving through the debris. As [4] have already noticed, a good
indicator of roughness is the local distance disturbance, since
small bumps and dents in the ground cause large changes in
distance of adjacent laser measurements because of the grazing
angles with which the laser rays hit the ground. This can
be seen in Fig. 2: The box represents the laser range finder,
mounted at a height $h_1$. The robot is standing in front of a
bumpy area. The bumps are assumed to have a fixed height $h_2$ in the considered region. Now each ray will either hit a bump or barely graze above one, hitting the ground behind.
The distance difference $e$ between a measurement that has hit the
ground, and one that has hit the bump is relatively large
compared to the height of the bump. In addition to that, it
linearly increases as the distance to the bump grows. This can
be seen in the following equation for $e$, which can be derived
using the intersecting lines theorem: $e = d \frac{h_1}{h_1 - h_2}$. Note how
$d$ scales up the remaining term, which in fact describes the
intensity of the bump.

We consider each of the 64 lasers in the 3D laser scan
separately. A heavily smoothed copy of the distance data is
made. Then the unfiltered version is subtracted from the low-
pass filtered version, forming a high-pass filtered version of
the data. Now, the distance deviation $\delta_i$ from the smoothed
version is known for each laser measurement. For each grid-
cell containing points, we compute the variance of distance
deviations $\sigma^2$ of all $n$ points in that cell, using

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} \delta_i^2 - \left( \frac{1}{n} \sum_{i=1}^{n} \delta_i \right)^2$$

(2)

However, $\sigma^2$ is really a sum of the inherent sensor noise $\sigma^2_i$ and the actual distance variance $\sigma^2$. Since we are only interested in the latter, and since we assume the sensor noise to be additive, we eliminate the sensor noise by computing $\sigma^2_i$ as $\max\{0, \sigma^2_i - \sigma^2\}$.

This measure is not yet independent to the distance, as we
have motivated above. Therefore, we obtain the local terrain roughness $r$ as we eliminate the previously computed measure
from the influence of the distance $r = \frac{\sqrt{\sigma^2}}{d_{cell}}$, where $d_{cell}$ is the distance of the laser to the respective grid-cell. The roughness $r$ can now be used to determine the drivability of the terrain:
High values of $r$ correspond to high terrain roughness, low
values to low roughness. Now a robot could, for example, try
to find a path, where the sum of all $r$s in the crossed cells is
minimal.

V. 3D MAPPING TECHNIQUES

The core of all 3D mapping techniques based on the
ICP algorithm is to find the correct transformations between
corresponding scans to achieve a globally consistent 3D map.
In this section we present our own approach and the 6D SLAM
of Nüchter et al.

A. Own Approach

In our approach we use the well-known ICP algorithm [3]
to calculate the transformation between two sets of 3D points
(scans) which correspond to a single shape. The two sets of
3D points are the model set $M$, with $|M| = N_m$, the data
set $D$, with $|D| = N_d$ and each 3D point $m_i$, $d_i$. The ICP
algorithm calculates the transformation $(R, t)$, consisting of
a rotation matrix $R$ and a translation vector $t$, as a minimum
of the following cost function:

$$E(R, t) = \sum_{i=1}^{N_m} \sum_{j=1}^{N_d} w_{i,j} \| m_i (R d_j + t) \|^2$$

(3)

$w_{i,j}$ are the weights for a point match and are set to 1, if
the $i$-th point of $M$ describes the same point in space as the
$j$-th point of $D$. The cost function is minimized iteratively.
The assumption of the algorithm is that after several iterations
the correspondences of the model set and data set is correct.
The iteration stops and the transformation is found if the cost
function $E(R, t)$ is lower than a predefined cost $\epsilon$. 

![Fig. 2. Geometry of distance disturbances.](image-url)
The transformation can be calculated in different ways [6, 7]. Our algorithm uses the algorithm of Walker et al. [10] which is based on dual quaternions. In contrast to other algorithms that minimize this algorithm the rotation and translation is calculated in one single step.

The quaternion is a 4D vector \( \hat{q} = (q_0, q_x, q_y, q_z)^T \), where \( q_0 \geq 0 \) and \( q_0^2 + q_x^2 + q_y^2 + q_z^2 = 1 \). respectively the pair \( (q_0, q) \). The dual quaternion \( \hat{q} \) is created by the quaternions \( \hat{q} \) and \( \hat{s} \) as \( \hat{q} = \hat{q} + \epsilon \hat{s} \). By adding the constraints of the transformation, (3) can be rewritten as

\[
E(\hat{q}, \hat{s}) = \frac{1}{N}[\hat{q}^T C_1 \hat{q} + N \hat{s}^T \hat{s} + \hat{s}^T C_2 \hat{q} + \text{const} + \lambda_1(\hat{q}^T \hat{q} - 1) + \lambda_2(\hat{s}^T \hat{s})] \tag{4}
\]

where \( \lambda_1 \) and \( \lambda_2 \) are the Lagrange multipliers. \( C_1, C_2 \) and \( \text{const} \) are defined using the matrix quaternion description \( M_i \) and \( D_i \) for the representation of the 3D points \( m_i \) and \( d_i \) as:

\[
\begin{align*}
C_1 &= -2 \sum_{i=1}^{N} M_i^T \bar{D}_i \\
C_2 &= 2 \sum_{i=1}^{N} \bar{D}_i - M_i \\
\text{const} &= 2 \sum_{i=1}^{N} (d_i^T \bar{d}_i + m_i^T \bar{m}_i)
\end{align*}
\]

By minimizing (4) the partial equation is built. For every iteration step of the ICP algorithm \( \hat{q} \) is calculated. As small changes in \( \hat{q} \) have significant effects on \( E(\hat{R}, t) \), the difference between the \( \hat{q} \) of the last iteration step and \( \hat{q} \) of the previous iteration step can be used as stop criterion. This stop criterion results in less computing time, since \( E(\hat{R}, t) \) has not to be calculated in every iteration step. For further details to the dual quaternion algorithm we refer to [10].

The ICP algorithm needs an initial pose value to obtain good estimates of the transformation between two sets of 3D points. As described in Section III, data of a GPS receiver and an IMU were measured during acquiring the Disaster City logfile. Therefore, the robot pose was estimated with 2 Hz by combining the orientation of the IMU device and the speed of the GPS. To deal with the GPS dropouts (see Fig. 3), we added a filter to obtain good pose estimations, even in situations with unreliable GPS measurements: During GPS dropout the speed reported by the GPS receiver changes dramatically. Therefore, if the speed changes by more than 6 km/h within a second, the speed will be set to the last reasonable measured value.

The 3D laser scanner reports about 1.5 million 3D points per second, which can hardly be handled in real-time by ICP algorithm. Therefore, the terrain classification [5] described in Section IV is used to extract systemically areas with significant, feature-rich objects. For the scan matching only 3D points from areas with obstacles are used. Using this reduction of laser points, the ICP algorithm can be applied in real-time.

B. 6D SLAM of Nüchter et al.

In order to compare the mapping results of our algorithm, we also calculated maps with the 6D SLAM software of Nüchter et al. which is available from [13]. The 3D mapping of this software is also based on the ICP algorithm, whereby the minimization of the error function is based on singular value decomposition, orthonormal matrices, unit quaternions or helical motion [6, 7] rather than dual quaternions. The software offers the option to choose between two methods to align 3D scans: Pairwise matching, where the new scan registered against the last scan only and incremental matching, where the new scan is registered against the union of all previous scans. This union of all previous scans is also called metascan. Both methods are accumulating the registration errors of each registration step. When the number of registered scans increases, the inconsistency of the map increases. When the robot is near a previously visited and mapped place, this error is observable. For this reason Nüchter et al. introduced a loop closing algorithm. If a loop closure is detected, the registration error is calculated and distributed over the transformations of the whole loop using the LUM algorithm (which is a GraphSLAM algorithm). The obtained results are transformed into a graph and the LUM algorithm is applied [1]. The LUM algorithm provides good results, but the computing time is high due to its brute force approach. In contrast, the ELCH algorithm optimizes the pose estimation by the ICP algorithm and thins out the graph [8].

VI. EXPERIMENTS

We performed several experiments with the captured data by applying our different algorithms to the data:

- We tested our hierarchical PCA-based terrain classification (see Section IV), focusing on uneven terrain and rubble piles.
- We mapped the area in 3D, using our plain vanilla ICP (see Section V-A) and an advanced 6D SLAM approach (see Section V-B). In the 6D SLAM software, loop closing, LUM and ELCH were used, but incremental scan matching was not.
Fig. 4. Driving on rough terrain next to a rubble pile and a forest. The elevated junction to the road (in the background) is also detected as drivable.

For the mapping experiment, the (filtered) pose estimation was used as a starting point for the ICP algorithm. Our software is able to replay the captured dataset in exactly the same way as the live application.

VII. RESULTS

We checked the result of the terrain classification by inspecting the output visually only, so the results presented here are preliminary. The main question was, if the algorithm will detect the unstructured obstacles correctly, and if it classifies the off-road paths through the unstructured environment as drivable, also with the whole robot vibrating and shaking due to the rough underground and the engine. A difficult situation is shown in Fig. 4.

The experiment showed that our hierarchical PCA-based terrain classification (for the obstacle detection) and the roughness analysis worked without any parameter tuning in the more unstructured areas. The ground was correct classified as rough, but drivable. We only found a wrong classification at the very beginning, when a rather slim robot (a Packbot without an arm) was scanned and detected as uneven ground and not as an obstacle.

For 3D mapping, two different ICP based algorithms were applied to the Disaster City logfile (see Section V). At several locations GPS dropouts occurred (see Fig. 3); there the GPS trajectory suddenly jumps and the GPS speed increases significantly. Therefore, for both algorithms the pose was estimated as described in Section V-A and the 3D scans were reduced using the terrain classification to speed up the computing time. The pose was estimated with 2 Hz, but the scan matching was performed with 1 Hz only, using the reduced set of datapoints. The drawback of the use of the terrain classification for the data reduction is that it introduces some aliasing into the data: 3D point clouds might be clipped at the border of a grid cell differently in two successive scans, resulting in non optimal matching. Also, because of the disregard of the floor points (flat areas are ignored), multiple planes can occur after the registration as illustrated in Fig. 5(c). Another error is visible at regions that are visited more than once; here walls show up multiple times. This is depicted in Fig. 5(a).

However, the result of the plain vanilla ICP is surprisingly good and locally consistent. The mean computation time for the terrain classification of each laser scan was 22.43 ms and the standard deviation 3.96 ms on an Intel(R) Core(TM) i7 QM with 1.73 GHz and 8 GB RAM. For the pairwise scan matching of the vanilla ICP the mean computation time was 209.06 ms with a standard deviation of 243.73 ms.

The 6D SLAM software with loop closing, LUM and ELCH enabled produced the consistent map shown in Fig. 5(b). However, the main disadvantage of the 6D SLAM software is that the 3D maps cannot be calculated in real time. Instead the time to generate the map – for the 13 min actual driving time – is about 60 min. Also, the optimal set of parameters for scan matching vary for different scenarios. In summary, the use of 6D SLAM software for mapping on an autonomous robot is right now not feasible, because of the lack of real-time capability and the need for parameter optimization depending on the environment.

VIII. CONCLUSION

This paper has presented a new, unique logfile from Disaster City, which will be provided to the research community on the Robotic 3D Scan Repository [14].

Our terrain classification was applied to this Disaster City logfile and worked well without any parameter tuning, even in the rougher areas. The algorithm performs in real-time. Furthermore, our 3D mapping algorithm using simple ICP works online (with 1 Hz), but the results are showing multiple planes and walls. The 6D SLAM software of Nüchter et al. [13] creates globally consistent 3D maps, but it does not work in real-time.

The quality and the robustness of the terrain classification has to be evaluated quantitatively. Our aim is to create high quality 3D maps in real-time. To enhance the quality of the ICP algorithm-based mapping, the reduction of the 3D point cloud has to be improved and the influences on the mapping algorithm has to be analyzed. Furthermore, a global optimization method has to be developed for outdoor mapping.
Fig. 5. 3D maps of the Disaster City logfile generated by different algorithms. Upper row ((a) and (b)): the whole maps. Lower row ((c) and (d)): detailed side view. Ground level points are colored in red, other points are colored from yellow to green to blue, depending on the height of the point above the ground.

So far, we did not use the information that can be extracted from the fiducials to evaluate the mapping. NIST is working on a software that can calculate the precision and accuracy of a map by measuring distances between different fiducials (and to other features).

ACKNOWLEDGEMENTS

We thank Robert Bolling and Chris Scrapper from The MITRE Corporation for providing the robot Centaur and all of the sensors. Also, we would like to thank Adam Jacoff from NIST for organizing the Response Robot Evaluation Exercise.

REFERENCES