Real-time Localization and Elevation Mapping within Urban Search and Rescue Scenarios

Alexander Kleiner
Institut für Informatik
University of Freiburg
79110 Freiburg, Germany
kleiner@informatik.uni-freiburg.de

Christian Dornhege
Institut für Informatik
University of Freiburg
79110 Freiburg, Germany
dornhege@informatik.uni-freiburg.de

Abstract

Urban Search And Rescue (USAR) is a time critical task. Rescue teams have to explore a large terrain within a short amount of time in order to locate survivors after a disaster. One goal in Rescue Robotics is to have a team of heterogeneous robots that explore autonomously, or partially guided by an incident commander, the disaster area. Their task is to jointly create a map of the terrain and to register victim locations, which can further be utilized by human task forces for rescue. Basically, the robots have to solve autonomously in real-time the problem of Simultaneous Localization and Mapping (SLAM), consisting of a continuous state estimation problem and a discrete data association problem. Extraordinary circumstances after a real disaster make it very hard to apply common techniques. Many of these have been developed under strong assumptions, for example, they require polygonal structures, such as typically found in office-like environments. Furthermore, most techniques are not deployable in real-time.

In this paper we propose real-time solutions for localization and mapping, which all have been extensively evaluated within the test arenas of the National Institute of Standards and Technology (NIST). We specifically deal with the problems of vision-based pose tracking on tracked vehicles, the building of globally consistent maps based on a network of RFID tags, and the building of elevation maps from readings of a tilted Laser Range Finder (LRF). Our results show that these methods lead under modest computational requirements to good results within the utilized testing arenas.

1 Introduction

Urban Search And Rescue (USAR) is a time critical task. Rescue teams have to explore a large terrain within a short amount of time in order to locate survivors after a disaster. In this scenario, the number of survivors decreases drastically by each day, due to hostile environmental conditions and injuries of victims. Therefore, the survival rate depends significantly on the efficiency of rescue teams. The idea behind Rescue Robotics is to adopt technologies from robotics for the direct support of rescue teams in the field. The main goal is to have a team of heterogeneous robots that explore autonomously, or partially guided by an incident commander, the disaster area. Their task is to jointly create a map of the terrain and to register victim locations, which can further be utilized by human task forces for rescue. Basically, the robots have to solve autonomously in real-time the problem of Simultaneous Localization and Mapping (SLAM), consisting of a continuous state estimation problem and a discrete data association problem. The state estimation problem, on the one hand, is hard due to the extremely unreliable odometry measurements usually found on robots operating within harsh environments. The data association problem, i.e. to recognize locations from sensor data, on the other hand, is challenging due to the unstructured environment, e.g. arbitrarily shaped debris from building collapse, and bad visibility conditions due to smoke and fire. These extraordinary circumstances make it very hard to apply common techniques from robotics. Many of these techniques have been developed under strong assumptions, for example, they require polygonal structures, such as typically found in office-like environments. Furthermore, most existing techniques are not deployable in real-time. In this paper we propose real-time solutions for localization and mapping, which all have been extensively evaluated within the test arenas of the National Institute of Standards and Technology (NIST) (Jacoff et al., 2001). We specifically deal with the problems of vision-based pose tracking on tracked vehicles, the building of globally consistent maps based on a network of RFID tags, and the building of elevation maps from readings of a tilted Laser Range Finder (LRF).

SLAM algorithms require a good estimate of the robot's pose, which is typically generated from the wheel odometry, e.g. measured by shaft encoders mounted on the robot's wheels, or from a scan matching algorithm applied to LRF measurements. However, wheel odometry becomes arbitrarily inaccurate if the robot has to drive on slippery ground or even has to climb over obstacles. Scan matching can only reliably be applied if the scanner continuously captures features, such as lines from corners and walls, which are not necessarily present in unstructured environments. We propose two approaches for the pose tracking problem, one for wheeled robots and one for tracked-robots, respectively. On the one hand, we have developed a wheeled robot with over-constrained odometry, i.e. four shaft-encoders instead of two, for the detection of slippage of the wheels. On the other hand, we solve the problem on tracked robots by utilizing a consumer-quality camera and an Inertial Measurement Unit (IMU), which provides estimates of the three Euler angles *yaw*, *roll*, and *pitch*. The basic idea is to continuously track salient features with the KLT feature tracker (Tomasi and Kanade, 1991) over images taken by the camera, and to calculate from the tracked features difference vectors that indicate the robot's motion. Here it suffices to determine only translations from the images, since rotations are measured by the IMU. Vectors that are affected by rotations are filtered out in advance. From the filtered set of vectors the translation of the robot is determined based on the individual voting of single translation vectors. Each vector votes for one of the possible translations according to a trained *tile coding* classificator (Sutton and Barto, 1998). It is assumed that the robot moves according to its heading and the underlying surface, i.e. it does not translate sidewards, downwards, and upwards. Furthermore, revolutions of the tracks are limited to constant velocities, either *forward*, *backward*, or *none*.

Data association, i.e. to recognize places that have been visited before, is a challenging problem after a disaster. Firemen at 9/11 reported that they had major difficulties to orientate themselves after leaving collapsed buildings. The arbitrary structure of the environment, and limited visibility conditions due to smoke, dust, and fire, prevent an easy distinction of different places. This problem is also relevant to standard approaches for SLAM, which recognize places by associating vision-based and LRF-based features. Therefore, we propose to solve the problem of data association by the active distribution and recognition of RFID tags. RFID tags have a worldwide unique number, and thus offer an elegant way to label and to recognize locations within harsh environments. Their size is already below $0.5mm$, as shown by the μ-chip from *Hitachi* (Hitachi, 2003), and their price is lower than 13 Cents (AlienTechnology, 2003). Passive RFID tags do not require to be equipped with a battery since they are powered by the reader if they are within a certain distance. Their reading and writing distance, which depends on the employed communication frequency, can be assumed to be within a range of meters.

Besides the solution of the data association problem, the RFID-technology based approach comes with three further advantages: First, in a multi-robot exploration scenario, maps from multiple robots can easily be merged to one consistent map by utilizing found correspondences from RFID tags registered on those maps. Furthermore, they can be utilized for a communication-free coordination of these robots (Ziparo et al., 2007). Second, RFID tags that have been put into the environment can be used in a straightforward manner by humans to follow routes towards victim location, i.e. they do not need to localize themselves within a metric map. Third, RFID tags can be used by human task forces to store additional user data, such as the number of victims located in a room or an indication of a hazardous area.

The basic idea behind the proposed RFID-based SLAM is to build successively a graph $G = (V, E)$ consisting of vertices V and edges E, where each vertex represents a RFID tag, and each edge $(V_i, V_j) \in E$ represents an estimate of the relative displacement $(\Delta x, \Delta y, \Delta \theta)^T$ with covariance matrix $\Sigma_{(\Delta x, \Delta y, \Delta \theta)}$ between the two RFID tags associated with the two vertices V_i and V_j, respectively. The relative displacement between two tags is estimated by a Kalman filter, which integrates pose corrections from the robot's pose tracking module, e.g. based on visual-odometry, an Inertia Measurement Unit (IMU), and laser-based scan-matching. RFID tags are actively deployed by the robot at adequate locations, as for example narrow passages that are likely to be passed. We utilized the algorithm of Lu and Milios (Lu and Milios, 1997) for calculating globally consistent maps from the online constructed RFID graph.

Pose tracking and data association lead to a consistent trajectory of the robot. This trajectory is typically taken as a basis for subsequently integrating readings from the LRF into an occupancy grid map (Moravec, 1988), leading to a two-dimensional map representation of the environment. However, in the disaster response context, 2D grid maps are not an adequate representation of the environment. In this context, robots are confronted with 3D structures, such as ramps and stairs, which they have to overcome. We propose a fast method for building elevation maps, which allows mapping in real-time, e.g. to build the map while the robot is in motion. Height values are estimated by a Kalman filter that integrates readings from a 35° downwards tilted LRF, with respect to the robot's full 3D pose, represented by six Degrees Of Freedoms (DOFs). The 3D pose of the robot is continuously tracked from the *pitch* and *yaw* angles measured by the IMU, and also continuously updated from height observations that have been registered on the map. Due to the integration of the full 3D pose, the method allows to create elevation maps while the robot traverses a three-dimensional surface, as for example while driving over ramps and stairs.

The remainder of this paper is structured as follows. In Section 2 we discuss related work. In Section 3 we introduce the sensors and experimental platforms utilized for the evaluation of the introduced methods. In the Sections 4, and Section 5 we describe the approaches for pose tracking on wheeled robots and tracked robots, respectively. The RFID technology-based SLAM approach is introduced in Section 6 and the real-time building of elevation maps is described in Section 7. Finally, we provide results from real world experiments in Section 8 and conclude in Section 9.

2 Related Work

The approach of visual odometry has extensively be studied in the past. Corke and colleagues introduced a solution for a planar rover equipped with an omni-directional vision system (Corke et al., 2004). In contrast to our work, which also aims at indoor application, they assume that the robot operates in an open space, as it is usually the case on planetary analog environments. Nister and colleagues presented a system for visual odometry that works with both mono and stereo vision (Nister et al., 2004). Their results generally show that the data processing of a stereo system leads to a highly accurate estimate of the robot's pose, which has also been confirmed by other researchers' work (Helmick et al., 2004; Milella and Siegwart, 2006). The usage of a stereo system has generally the advantage that the depth information of features tracked by the vision system can be utilized for computing the velocity of the robot. Results proposed in this paper show that with the simplified kinematics of tracked robots, a single but lightweight camera solution can also lead to sufficiently accurate pose estimates.

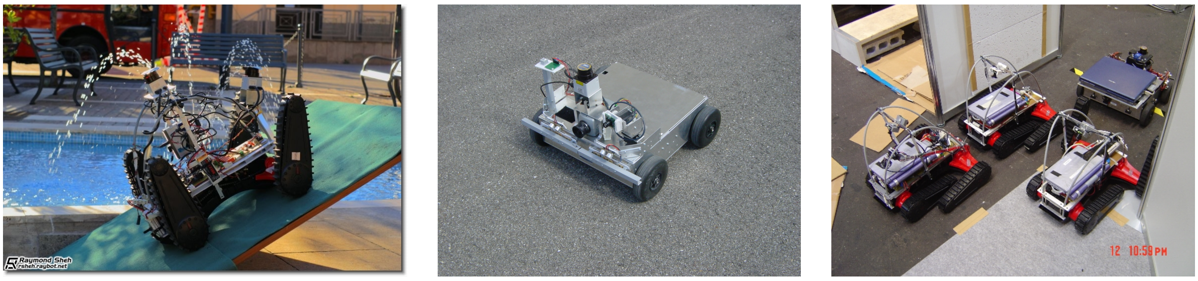
Hähnel and colleagues (Hähnel et al., 2004) successfully utilized Markov localization for localizing a mobile robot in an office environment. Their approach deals with the problem of localization within a map previously learned from laser range data and known RFID positions, whereas the work presented in this paper describes a solution that performs RFID-based localization and mapping simultaneously while exploration. Also sensor networks-based Markov localization for emergency response has been studied (Kantor et al., 2003). In this work existing sensor nodes in a building are utilized for both localization and for the computation of a temperature gradient from local sensor node measurements. Bohn and colleagues examined localization based on super-distributed RFID tag infrastructures (Bohn and Mattern, 2004). In their scenario tags are deployed beforehand in a highly redundant fashion over large areas, e.g. densely integrated into a carpet. They outline the application of a vacuum-cleaner robot following these tags. Miller and colleagues examined the usability of various RFID systems for the localization of first responders within different building classes (Miller et al., 2006). During their experiments persons where tracked with a Dead Reckoning Module (DRM) while walking through a building. They showed that the trajectories can be improved by utilizing the positions of RFID tags detected within the building. While these map improvements have been carried out with only local consistency, the approach presented in this work yields a globally consistent map improvement.

Elevation maps are indispensable, particularly for robots operating within unstructured environments. They have been utilized on wheeled robot platforms (Pfaff et al., 2005; Wolf et al., 2005), on walking machines (Krotkov and Hoffman, 1994; Gassmann et al., 2003), and on car-like vehicles (Thrun et al., 2006; Ye and Borenstein, 2003). These methods differ in the way how range data is acquired. If data is acquired from a 3D scan (Pfaff et al., 2005; Krotkov and Hoffman, 1994), it usually suffices to employ standard error models, which reflect uncertainty from the measured beam length. Data acquired from a 2D LRF, e.g. tilted downwards, requires more sophisticated error models, such as the compensation of pose uncertainty (Thrun et al., 2006), and handling of missing data by map smoothing (Ye and Borenstein, 2003). Furthermore, full 3D data processing is usually not always possible in real-time, instead it requires the robot to wait until the full 3D scan is taken. In contrast to previous work, our approach deals with the problem of building elevation maps in real-time and with respect to the full 6 DOF pose of the robot.

3 Experimental platform

The work proposed in this paper has been extensively tested on two different robot platforms, a 4WD (four wheel drive) differentially steered robot for the autonomous team exploration of large office-like areas, and a tracked robot for climbing 3D obstacles. Figure 1(a) shows the tracked *Lurker* robot, which is based on the *Tarantula* R/C toy. Although based on a toy, this robot is capable of climbing difficult obstacles, such as stairs, ramps, and random stepfields. This is possible due to its tracks, which can operate independently on each side, and the "Flippers" (i.e. the four arms of the robot), which can be freely rotated by 360°. We heavily modified the base in order to enable autonomous operation. First, we added a 360° freely turnable potentiometer to each of the two axes for measuring the angular position of the flippers. Second, we added touch sensors to each flipper, allowing the robot to measure force when touching the ground or an object. Furthermore, the robot is equipped with a 3-DOF Inertial Measurement Unit (IMU) from *Xsens*, allowing drift-free measurements of the three Euler angles *yaw*, *roll*, and *pitch*, and two *Hokuyo URG-X004* Laser Range Finders (LRFs), one for scan matching, and one for elevation mapping, which can be tilted in the pitch angle within 90°. For feature tracking, as proposed in Section 5, we utilized a *Logitech QuickCam Pro 4000* web cam (Logitech, 2006).

Figure 1(b) shows the *Zerg* robot, a 4WD differentially steered platform, which has been completely hand-crafted. The 4WD drive provides more power to the robot and therefore allows to drive up ramps and to operate on rough terrain. Each wheel is driven by a *Pitman GM9434K332* motor with a *19.7:1* gear ratio and a shaft encoder. The redundancy given by four encoders allows to detect heavy slippage and situations in which the robot gets stuck, as we will show in Section 4. In order to reduce the large odometry error that naturally arises from a four-wheeled platform, we also utilized an Inertial Measurement Unit (IMU) from *XSens*. Moreover, the robot is equipped with a *Thermal-Eye* infrared thermo camera for victim detection, and also with a *Hokuyo URG-X004* LRF. Localization experiments have been carried out with Ario RFID chips from *Tagsys* with a size of $1.4 \times 1.4cm$, $2048Bit$ RAM, and a response frequency of 13.56MHz. For the reading and writing of these tags, we employed a Medio S002 reader, likewise from *Tagsys*, which operates within a range



(a) (b) (c)

Figure 1. Robots designed for rescue scenarios: (a) The *Lurker* robot, and (b) the *Zerg* robot. (c) The team of robots waiting for the mission start during the RoboCup competition in Osaka 2005 (Pictures a,c were taken by Raymond Sheh, and Adam Jacoff, respectively).

of approximately $30cm$ while consuming less than $200mA$. The antenna of the reader is mounted in parallel to the ground. This allows to detect any RFID tag lying beneath the robot. The active distribution of these tags is carried out by a self-constructed actuator based on a metal slider that can be moved by a conventional servo. The slider is connected with a magazine that maximally holds around 100 tags. Each time the mechanism is triggered, the slider moves back and forth while dropping a single tag from the magazine.

4 Wheel odometry-based pose tracking

The two-dimensional pose of the robot can be represented by the vector $l = (x, y, \theta)^T$. In order to represent uncertainties, the pose is modeled by a Gaussian distribution $N(\mu_l, \Sigma_l)$, where μ_l is the mean and Σ_l a 3×3 covariance matrix (Maybeck, 1979). Robot motion is measured by the odometry and given by the traveled distance d and angle α, likewise modeled by a Gaussian distribution $N(u, \Sigma_u)$, where $u = (d, \alpha)$ and Σ_u is a 2×2 covariance matrix expressing odometry errors. The pose at time t can be updated from input u_t as follows:

$$l_t = F(l_{t-1}, u_t) = \begin{pmatrix} x_{t-1} + \cos(\theta_{t-1}) d_t \\ y_{t-1} + \sin(\theta_{t-1}) d_t \\ \theta_{t-1} + \alpha_t \end{pmatrix}, \tag{1}$$

$$\Sigma_{l_t} = \nabla F_l \Sigma_{l_{t-1}} \nabla F_l^T + \nabla F_u \Sigma_u \nabla F_u^T, \tag{2}$$

$$where \quad \Sigma_u = \begin{pmatrix} d\sigma_d^2 & 0 \\ 0 & \alpha\sigma_\alpha^2 \end{pmatrix} \tag{3}$$

and ∇F_l and ∇F_u are partial matrices of the Jacobian matrix ∇F_{lu}.

If the robot operates on varying ground, as for example concrete or steel sporadically covered with newspapers and cardboard, or if it is very likely that the robot gets stuck within obstacles, odometry errors are not Gaussian distributed anymore, but are dependent on the particular situation. Therefore, we designed the Zerg robot with an over-constrained odometry for the detection of slippage of the wheels by utilizing four shaft-encoders, one for each wheel. From these four encoders we recorded data while the robot was driving on varying ground, and labeled the data sets with the classes $C = (slippage, normal)$. This data was taken to learn a decision tree (Quinlan, 2003) with the inputs $I = (\Delta v_{Left}, \Delta v_{Right}, \Delta v_{Front}, \Delta v_{Rear})$,

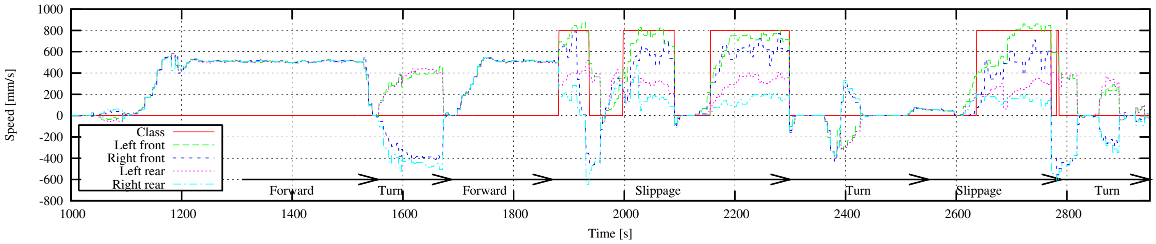


Figure 2. Odometry measurements from a 4WD robot, each dashed line corresponds to one of the four wheel velocities. The black arrows indicate the true situation, e.g. driving forward, slippage, etc., and the red line corresponds to the automatic classification into *slippage* (800) and *normal* (0), given the four wheel velocities.

representing the velocity differences of the four wheels, respectively. As depicted in Figure 2, the trained classifier reliably detects this slippage from the velocity differences. Given the detection of slippage, the odometry measurement of the traveled distance d is set to zero and the robot's pose is updated according to Equation 3, however, with $d\sigma_d^2$ replaced by the term σ_d^2, within covariance matrix Σ_u, in order to increase uncertainty in translation, also if d cannot be measured. Note that the rotation update stays unmodified since the traveled angle α is measured by the IMU and thus not influenced by slippage of the wheels.

4.1 Pose improvement by laser-based scan matching

Pose tracking from odometry data can be further improved by utilizing range measurements from the LRF. We utilize an incremental scan matching method (Hähnel, 2005) for pose tracking and fuse the result from both the scan matcher and odometry with a Kalman filter. The scan matching technique determines from a sequence of previous scan observations $z_t, z_{t-1}, ..., z_{t-n}$ subsequently for each time point t an estimate of the robot's pose k_t. This is carried out by incrementally building a local grid map from the n most recent scans and estimating the new pose k_t of the robot by maximizing the likelihood of the scan alignment of the latest scan z_t in the current map. The robot pose $N\left(l_t, \Sigma_{l_t}\right)$ is fused with the pose of the scan matcher $N\left(k_t, \Sigma_{k_t}\right)$ by:

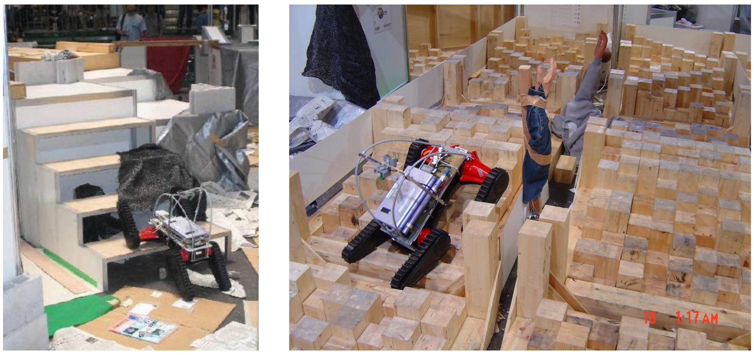
$$l_{t+1} = \left(\Sigma_{l_t}^{-1} + \Sigma_{k_t}^{-1}\right)^{-1} \left(\Sigma_{l_t}^{-1} l_t + \Sigma_{k_t}^{-1} k_t\right) \quad (4)$$

$$\Sigma_{l_{t+1}} = \left(\Sigma_{l_t}^{-1} + \Sigma_{k_t}^{-1}\right) \quad (5)$$

Scan matching-based pose tracking leads to good results if the scanner captures sufficient features in the environment, such as walls and corners. However, if the LRF has a restricted field of view, e.g. a range limit at four meters in case of the *Hokuyo* LRF utilized in our system, scanning leads to a large number of *far readings*, i.e. measurements at maximum range, and hence an insufficient amount of features. We solve this problem by increasing the variance Σ_{k_t} of the scan matcher with respect to the number of detected far readings, which leads to an increase of the influence of the odometry estimate within update Equation 5. A series of experiments has shown that the result of scan matching can reliably be fused as long as the amount of far readings does not exceed 20% of the scan.

5 Visual odometry-based pose tracking

Figure 3 depicts 3D structures, such as stairs (Figure 3(a)) and random stepfields (Figure 3(b)), as they are typically found within the test arenas from NIST. On such obstacles, tracks are very likely to slip during locomotion, and thus the measurement of their revolutions is not sufficient for tracking the robot's pose. Also pose tracking from LRF data is much harder since 2D LRFs insufficiently reflect the environmental structure, e.g. minor variations in the vehicle's roll might lead to major variations in the range measurements obtained from the LRF.



(a) (b)

Figure 3. The *Lurker* robot during the RoboCup Rescue competition in Osaka. (a) climbing stairs, and (b) searching for victims in a large random stepfield (Pictures b was taken by Adam Jacoff).

In this section we describe a solution to pose tracking on tracked robots for allowing autonomous and semi-autonomous behavior on complex obstacles, as shown in Figure 3, as well as SLAM on tracked vehicles. We assume that the robot most likely moves according to its heading and the underlying surface, i.e. it does not translate lateral, downwards, and upwards. We also assume that the IMU provides sufficiently accurate estimates of the three Euler angles *yaw*, *roll*, and *pitch*. Furthermore, revolutions of the tracks are limited to constant velocities, either *forward*, *backward*, or *none*.

The tracking is carried out by subsequently processing images from a consumer-quality monocular camera. The basic idea is to continuously track salient features with the KLT feature tracker (Tomasi and Kanade, 1991) over multiple images taken by the camera and to calculate from the tracked features difference vectors that indicate the robot's motion. Since we estimate rotations by the IMU and thus are only interested in determining translations from the images, vectors that are affected by rotations are filtered out in advance. From the filtered set of vectors the true translation of the robot is determined based on the individual voting of single translation vectors. Each vector votes for one of the possible translations according to a previously trained *tile coding* classificator (Sutton and Barto, 1998).

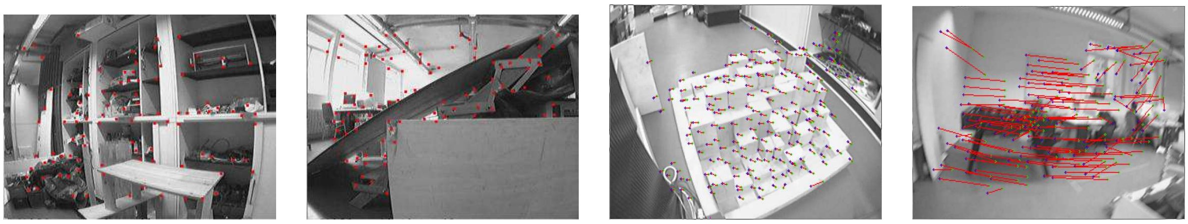
In general, an image sequence can be described by a discrete valued function $I(x, y, t)$, where x, y describe the pixel position and t describes the time. We assume that features detected in an image also appear in the subsequent image, however translated by $d = (\xi, \eta)^T$:

$$I(x, y, t + \tau) = I(x - \xi, y - \eta, t) \tag{6}$$

Usually, a feature tracker determines this translation by minimizing the squared error ϵ over a tracking window. For brevity we define $I(x, y, t + \tau)$ as $J(x)$ and $I(x - \xi, y - \eta, t)$ as $I(x - d)$, leading to the following error measure with a weighting function w (Tomasi and Kanade, 1991).

$$\epsilon = \int_{\mathcal{W}} [I(x - d) - J(x)]^2 w dx \tag{7}$$

To facilitate the process of feature tracking, the selection of appropriate features, i.e. features that can easily be distinguished from noise, is necessary. Hence, features that show light-dark changes, e.g. edges, corners, and crossings, are selected with high probability by the KLT feature tracker. In Figure 4, examples of KLT's adaptive feature selection and the tracking over a series of images are shown. For our purpose we use an implementation of the KLT tracker by Stan Birchfeld (Birchfeld, 1996).



(a) (b) (c) (d)

Figure 4. KLT feature tracking: (a,b) Features (red dots) are adaptively selected within images. (c) Feature tracking over two subsequent images. The vectors between two corresponding features, shown by red lines, indicate the movement of the camera. (d) The tracking over a series of five images.

5.1 Tracking over a series of images

Our main goal is to determine the robot's forward or backward movement. However, when traversing obstacles, the robot's motion is not an exclusively forward or backward motion. It is overlaid with noise that originates from slippage of the tracks and shaking of the robot's body due to rough terrain, leading to jitter effects. Since the above described effects usually do not accumulate over time, and hence can be reduced, our method generates trackings over multiple frames, rather than performs single frame trackings only. This is achieved by saving single trackings between two subsequent images in a buffer, up to a maximal amount. If trackings of the same feature coexist over more than two images, their corresponding translation vectors $d_i, d_{i+1}, ..., d_k$ are replaced by a single translation vector d_{ik}, consisting of the vector sum of all trackings between d_i and d_k. The summed translation vector is more robust compared to single trackings, since it averages out irregular jitter effects. We assign a weight $w_{ik} = |k - i|$ to each tracking in order to reward trackings over multiple frames [1].

5.2 Filtering of Rotations

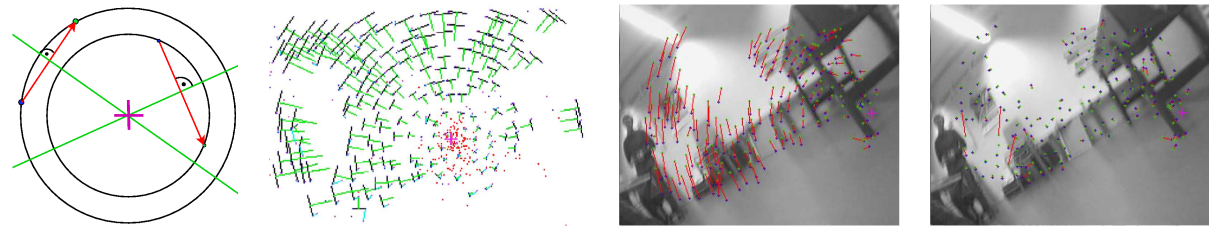
Since we focus on the translation estimation from image sequences, rotations have to be filtered out in advance. However, due to the high latency time of the employed camera system

[1] Note that these weights are used during the voting process, which will be described in Section 5.3.

(a web cam connected via USB 1.1), this can only be accurately achieved on the image data directly, rather than by combining image data with rotation angles from the IMU sensor. Given a feature tracking between two images of the form $(x_i, y_i) \rightarrow (x_j, y_j)$, which includes a rotation around the point r_x, r_y with angle α, one can derive a corresponding rotation free tracking $(x_i, y_i) \rightarrow (x'_j, y'_j)$ after the following equation, with given rotation matrix $R(.)$:

$$(x'_j, y'_j) = (r_x, r_y) + R(-\alpha) \cdot (x_j - r_x, y_j - r_y) \tag{8}$$

Therefore, in order to perform the filtering of rotations, one has to determine the rotation center (r_x, r_y) and rotation angle α. Rotating points of different radii describe concentric circles around the rotation center. When considering two feature trackings whose features are lying on a circle, one can see that the perpendicular bisectors of the two lines, respectively connecting start- and endpoint of each feature tracking, subtend in the rotation center, as shown in Figure 5(a).



(a) (b) (c) (d)

Figure 5. (a) The perpendicular bisectors (green) of the tracking vectors (red) subtend at the center of the circle (magenta). (b) Example of the Monte Carlo algorithm: The perpendicular bisectors (green) point to the center of rotation (magenta). Red dots depict the sampled intersection points. (c,d) Example of the rotation correction while the robot changes the angles of its front flippers. The feature vectors before (c) and after (d) the correction.

Algorithm 1: Sample up to n possible centers of rotation

Input: A set of feature trackings: T
Output: A set of calculated intersection points: C

C = $\emptyset$;
for i = 0; $i < n$; i++ **do**
    $t_1 \leftarrow$ `selectRandomFeatureTracking(`T`);`
    $t_2 \leftarrow$ `selectRandomFeatureTracking(`T`);`
    $s_1 \leftarrow$ `calculatePerpendicularBisector(`t_1`);`
    $s_2 \leftarrow$ `calculatePerpendicularBisector(`t_2`);`
    $(cut, det) \leftarrow$ `calculateIntersectionPoint(`s_1, s_2`);`
    if $det < minDeterminant$ **then**
        continue ;
    end
    C $\leftarrow$ **C** $\cup$ **cut;**
end

We exploit this property with a Monte Carlo algorithm for estimating the true center of rotation (see Figure 5(b)). First, up to n possible centers of rotation are sampled from the set of feature trackings T by algorithm 1. Second, all sampled centers of rotation are put into a histogram, whereas the final center is determined by the histogram's maximum.

Furthermore, one has to determine the rotation angle, which can be done by calculating the vector cross product. Given a feature tracking $(x_i, y_i) \rightarrow (x_j, y_j)$ rotated around (r_x, r_y) by α, one can calculate the cross product by considering the start- and endpoint of the feature trackings as endpoints of vectors starting at the rotation center. Suppose $v_i = (x_i - r_x, y_i - r_y)^T$ and $v_j = (x_j - r_x, y_j - r_y)^T$ are vectors derived from tracking images I and J, respectively. Then, the angle between these vectors $\alpha = \angle(v_i, v_j)$ can be calculated from the cross product as follows.

$$v_i \times v_j = ||v_i|| \cdot ||v_j|| \cdot \sin(\alpha) \tag{9}$$

Given the rotation center (r_x, r_y) from the previous estimation, one can determine the true rotation angle α by averaging over rotation angles from all single feature trackings. Finally, it is necessary to prevent the algorithm from being executed on rotation-free sequences. This is achieved by only adding a center of rotation to the histogram, if it is located within the bounding box of the image. Center of rotations that are far from the bounding box are most likely due to quasi-parallel feature translations, which in turn indicate a rotation-free movement. If the number of centers of rotation is below a threshold λ, the transformation of Equation 8 is not applied. We determined experimentally $\lambda = 10$.

5.3 Classification

From the set of filtered translation vectors, one can determine the robot's translation. However, the projection from translation vectors of the vision system to the robot's translation depends on the intrinsic parameters of the camera, e.g. focal length and lens distortion, and on the extrinsic parameters of the camera, e.g. the translation and rotation relative to the robot's center. This projection can either be determined analytically or by a mapping function. Due to the assumption of a simplified kinematic model, we decided to learn this mapping with a function approximator described in this section.

5.3.1 Learning classification probabilities

The learning is based on data collected and automatically labeled during teleoperation runs under mild conditions, i.e. without heavy slippage. During a second phase, the data labeling has been verified on a frame to frame basis. This procedure allows the efficient labeling of thousands of trackings since single images contain several features. Each labeled tracking is described by the class assignment $c \in C$ and the vector $v = (x, y, l, \alpha)^T$, where x, y denotes the origin in the image, l denotes the vector length and α denotes the vector heading. As already mentioned, class assignments are concerning the robot's translation, e.g. $C = \{forward, backward, ...\}$. Given the labeled data, tile coding function approximation is used for learning the probability distribution

$$P(c \mid x, y, l, \alpha). \tag{10}$$

Tile coding is based on tilings which discretize the input space in each dimension. Shape and granularity of these discretizations can be adjusted according to the task. For example, the discretizations regarding the origin x, y has been designed coarser due to minor local differences regarding the correlation with the class assignment. Furthermore, tilings are overlaid with a randomized offset in order to facilitate generalization. During learning, each tile is updated according to:

$$w_{i+1} = w_i + \alpha \cdot (p_{i+1} - w_i), \tag{11}$$

where w_i is the weight stored in the tile, $p_i \in \{0, 1\}$ is the class probability, and α is the learning rate, which is set to $\alpha = \frac{1}{m}$, where m is the number of total update steps, in order to ensure normalized probabilities.

5.3.2 Classification by voting

Based on the probability distribution in equation 10, each vector v_i votes individually for a class assignment c_i with respect to its location, length and heading:

$$c_i = \underset{c \in C}{\operatorname{argmax}}\, P(c \mid x_i, y_i, l_i, \alpha_i) \tag{12}$$

Let $c_i^k = I\left(c_i = k\right)$ be the class indicator function, which returns 1 if $c_i = k$ and otherwise 0. Then, the final classification a can be decided based on the maximal sum of weighted individual votes from each vector:

$$a = \underset{k \in C}{\operatorname{argmax}} \sum_{i=1}^{N} c_i^k \cdot w_i \tag{13}$$

Note that w_i increases according to the number of times the underlying feature has successfully been tracked by the feature tracker described in Section 5.1.

5.4 Generating odometry data

In order to determine the distance d traveled between two images I and J, we assume a constant translational velocity v_T of the robot [2]. Given time stamp t_j and t_i of image I and J, respectively, d can be calculated by:

$$d = \begin{cases} v_T \cdot (t_j - t_i) & \text{if class} = \text{forward} \\ -v_T \cdot (t_j - t_i) & \text{if class} = \text{backward} \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Finally, from the yaw angle θ of the IMU and the robot's last pose $(x_{old}, y_{old}, \theta_{old})^T$ we calculate the new pose of the robot:

$$(x_{new}, y_{new}, \theta_{new})^T = (x_{old} + d \cdot \cos(\theta), y_{old} + d \cdot \sin(\theta), \theta)^T \tag{15}$$

A more detailed description of this approach is found in (Dornhege and Kleiner, 2006).

6 RFID Technology-based SLAM

SLAM consists of a state estimation problem and a data association problem. The techniques described so far solve the state estimation problem by continuously tracking the pose of the robot. However, as commonly known, these methods suffer from the problem of error accumulation, i.e. the quality of the pose estimate will decrease according to the length of the traveled trajectory. This problem is typically solved by recognizing (associating) environmental structures, such as features extracted from range scans and color images that

[2]Note that this value could also be automatically adjusted according to the vehicles current set-velocity.

have been observed on the trajectory before. However, particularly within the context of disaster response, data association from range scans and color images can only limitedly be applied. Fireman reported that even for them the recognition of previously visited places is extremely hard, which is due to limited visibility caused by smoke, fire, or dust, and also due to the lacking of recognizable features, such as corridors and doorways.

RFID tags have a world-wide unique encoded number and thus provide an elegant way to mark places with bounded uncertainty in perception. The basic idea of the proposed approach is to actively distribute these tags in the environment, i.e. to place them automatically on the ground, and to measure the relative distances between them. Figure 6 (a) depicts the hand-crafted RFID deploy device and Figure 6 (b) shows the utilized RFIDs. From the correspondences of recognized RFID tags and the measured distances from the robot's trajectory, a globally consistent map is calculated according to the method introduced by Lu and Milios (Lu and Milios, 1997). This method can be illustrated by considering the analogy to a spring-mass system (see Figure 6(c)). Consider the locations of RFIDs as masses and the measured distances between them as springs, whereas the uncertainty of a measurement corresponds to the hardness of the spring. Then, finding a globally consistent map is equivalent to finding an arrangement of the masses that requires minimal energy.

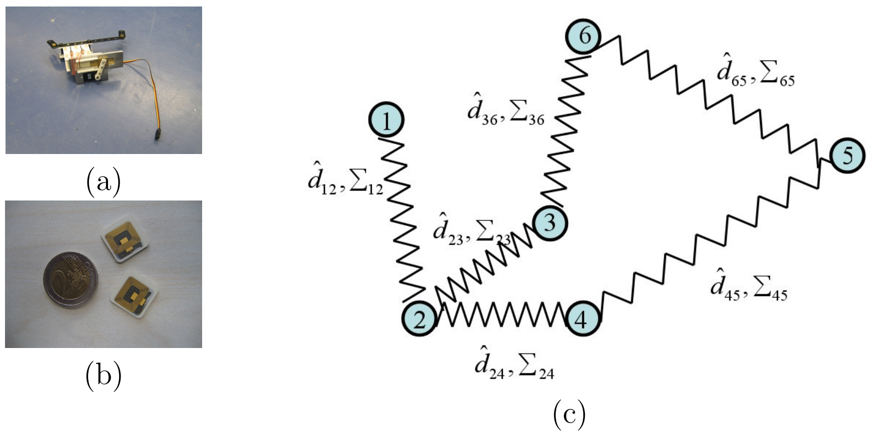


Figure 6. RFID technology-based SLAM: (a) The RFID tag deploy device. (b) The utilized RFID tags. (c) The spring-mass analogy to the generated RFID graph. Vertices represent RFIDs (masses) and edges between them represent measured distances with covariances (springs).

The proposed method successively builds a graph $G = (V, E)$ consisting of vertices V and edges E, where each vertex represents an RFID tag, and each edge $(V_i, V_j) \in E$ represents a measurement $\hat{d}_{ij}$ of the relative displacement $(\Delta x, \Delta y, \Delta\theta)^T$ with covariance matrix $\Sigma_{(\Delta x, \Delta y, \Delta\theta)}$ between the two RFID tags associated with the two vertices V_i and V_j, respectively. The relative displacement between two tags is estimated by a Kalman filter, which integrates pose corrections from the robot's wheel odometry, an Inertia Measurement Unit (IMU), and laser-based scan-matching as described in Section 4. If the robot passes a tag, the Kalman Filter is reset in order to estimate the *relative* distance $\hat{d}_{ij}$ to the subsequent tag on the robot's trajectory.

We denote the true pose vectors of $n + 1$ RFID nodes with $x_0, x_1, \ldots, x_n$, and the function calculating the true distance between a pair of nodes (x_i, x_j) as measurement function d_{ij}. The noisy measurement of the distance between two nodes (x_i, x_j) is denoted by $\hat{d}_{ij} = d_{ij} + \Delta d_{ij}$. We assume that the error Δd_{ij} is normally distributed and thus can be modeled by a Gaussian distribution with zero mean and covariance matrix Σ_{ij}. Our goal is to find the true locations of the x_{ij} given the set of measurements $\hat{d}_{ij}$ and covariance matrices Σ_{ij}. This can be achieved with the maximum likelihood concept by minimizing the following Mahalanobis-distance:

$$\mathbf{x} = \arg\min_{\mathbf{x}} \sum_{i,j} (d_{ij} - \hat{d}_{ij})^T \Sigma_{ij}^{-1} (d_{ij} - \hat{d}_{ij}), \tag{16}$$

where $\mathbf{x}$ denotes the concatenation of poses $x_0, x_1, \ldots, x_n$. Moreover, we consider the graph as fully connected, and if there does not exist a measurement between two nodes, the inverse covariance matrix Σ_{ij}^{-1} is set to zero. Note if the robot's pose is modeled without orientation θ, e.g. because measurements from the IMU are sufficiently accurate, the optimization problem can be solved linearly by inserting $d_{ij} = x_i - x_j$ in Equation 16:

$$\mathbf{x} = \arg\min_{\mathbf{x}} \sum_{i,j} (x_i - x_j - \hat{d}_{ij})^T \Sigma_{ij}^{-1} (x_i - x_j - \hat{d}_{ij}). \tag{17}$$

Since measurements are taken relatively, we assume without loss of generality that $x_0 = 0$ and $x_1, \cdots, x_n$ are relative to x_0. In order to solve the minimization problem analytically, Equation 17 can be rewritten in matrix form:

$$\mathbf{x} = \arg\min_{\mathbf{x}} (\hat{\mathbf{d}} - \mathbf{hx})^T \mathbf{\Sigma}^{-1} (\hat{\mathbf{d}} - \mathbf{hx}), \tag{18}$$

where $\mathbf{hx}$ denotes the measurement function in matrix form with $\mathbf{h}$ as an index function whose elements are $\{1, -1, 0\}$ and $\mathbf{x}$ as the concatenation of pose vectors. Furthermore, $\hat{\mathbf{d}}$ denotes the concatenation of observations $\hat{d}_{ij}$, and $\mathbf{\Sigma}^{-1}$ denotes the inverse covariance matrix of $\hat{d}_{ij}$, consisting of the inverse sub-matrices Σ_{ij}. Finally, the minimization problem can be solved by:

$$\mathbf{x} = (\mathbf{h}^T \mathbf{\Sigma}^{-1} \mathbf{h})^{-1} \mathbf{h}^T \mathbf{\Sigma}^{-1} \hat{\mathbf{d}} \ . \tag{19}$$

and covariance of $\mathbf{x}$ can be calculated by:

$$\mathbf{c_x} = (\mathbf{h}^T \mathbf{\Sigma}^{-1} \mathbf{h})^{-1} \tag{20}$$

Equation 19 can be solved in $O(n^3)$ if the covariances Σ_{ij} are invertible. In practice, we assume that measurements are independent from each other, consequently the Σ_{ij} are given as diagonal matrices. Moreover, since many nodes in the graph are unconnected, most Σ_{ij}^{-1} are set to zero. Therefore, $\mathbf{\Sigma}$ is a sparse matrix and can in general be inverted efficiently. In order to utilize Equation 19 for the correction of the orientation angle θ, measurement equation d_{ij} has to be linearized by a Taylor expansion (Lu and Milios, 1997). Since the linearization leads to errors, the procedure has to be applied iteratively. We noticed during our practical experiments that five to six iterations are sufficient.

In case the robot detects the same RFID tag consecutively, it is necessary to account for the spacial expansion of the utilized RFID antenna. We use an antenna with an expansion of approximately $20 \times 3cm$, mounted parallel to the ground. RFID tags beneath the robot within this expansion are successfully detected. Unfortunately, it is not possible to tell the exact position of the detection within this range. However, in the average case, RFIDs are detected within the antenna's center. Therefore, we model the distance between identical tags by $\hat{d}_{ii} = (0, 0, \Delta\theta)$ and covariance matrix $\Sigma_{Antenna}$, which reflects the shape of the antenna and the orientation of the robot, i.e. it has a low uncertainty into the robot's direction, and a high uncertainty orthogonal to it. See (Kleiner et al., 2006) for a more detailed description of this approach.

7 Building Elevation Maps in real-time

In this section we describe a Kalman filter-based approach for building elevation maps by integrating range measurements from a LRF tilted downwards, whereas the map is incrementally built in real-time, e.g. while the mobile robot explores an uneven surface. The necessity of computing elevation maps in real time is to enable the robot to plan its trajectory continuously on the incrementally constructed map during execution.

An elevation map is represented by a two-dimensional array storing for each global location (x^g, y^g) a height value h with variance σ_h^2. In order to determine the height for each location, endpoints from the LRF readings are transformed from robot-relative distance measurements to global locations, with respect to the robot's global pose, and the pitch (tilt) angle of the LRF (see Figure 7). This section is structured as follows. In Section 7.1 we describe the update of single cell values relative to the location of the robot, in Section 7.2 we show the filtering of the map with a convolution kernel and in Section 7.3 we describe an algorithm for the estimation of the robot's 3D pose from dead reckoning and map observations.

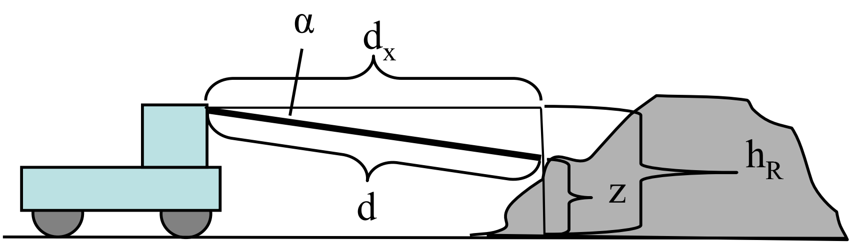


Figure 7. Transforming range measurements to height values.

7.1 Single cell update from sensor readings

Our goal is to determine the height estimate for a single cell of the elevation map with a Kalman filter (Maybeck, 1979), given all height observations of this cell in the past. We model height observations z_t by a Gaussian distribution $N\left(z_t, \sigma_{z_t}^2\right)$, as well as the current estimate $N\left(\hat{h}\left(t\right), \sigma_{\hat{h}(t)}^2\right)$ of each height value. Note that the height of cells cannot be observed directly, and thus has to be computed from the measured distance d and LRF pitch angle α. Measurements from the LRF are mainly noisy due to two error sources. First, the returned distance depends on the reflection property of the material, ranging from very good reflections, e.g. white sheet of paper, to nearly no reflections, e.g. black sheet of paper. Second, in our specific setting, the robot acquires scans while navigating on rough terrain. This will lead to strong vibrations on the LRF, causing an oscillation of the laser around the servo-controlled pitch angle. Consequently, we represent measurements from the LRF by two normal distributions, one for the measured distance $N(\mu_d, \sigma_d)$, and one for the pitch angle $N(\mu_\alpha, \sigma_\alpha)$.

The measurements from the LRF are transformed to robot-relative locations (x^r, y^r). First, we compute the relative distance d_x and the height z of each measurement according to the following equation (see Figure 7):

$$\begin{pmatrix} d_x \\ z \end{pmatrix} = F_{d\alpha} \begin{pmatrix} d \\ \alpha \end{pmatrix} = \begin{pmatrix} d \cos \alpha \\ h_R - d \sin \alpha \end{pmatrix}, \tag{21}$$

where h_R denotes the height of the LRF mounted on the robot. Second, from distance d_x and the horizontal angle β of the laser beam, the relative cell location (x^r, y^r) of each cell can be calculated by:

$$x^r = d_x \cos \beta \tag{22}$$

$$y^r = d_x \sin \beta \tag{23}$$

Equation 21 can be utilized for computing the normal distributed distance $N(\mu_{d_x}, \sigma_{d_x})$, and height $N(\mu_z, \sigma_z)$, respectively. However, since this transformation is non-linear, $F_{d\alpha}$ has to be linearized by a Taylor expansion at μ_{d_x}, μ_z:

$$\begin{pmatrix} \mu_{d_x} \\ \mu_z \end{pmatrix} = F_{d\alpha} \begin{pmatrix} d \\ \alpha \end{pmatrix} \tag{24}$$

$$\Sigma_{d_x z} = \nabla F_{d\alpha} \Sigma_{d\alpha} \nabla F_{d\alpha}^T \tag{25}$$

$$with \quad \nabla F_{d\alpha} = \begin{pmatrix} \cos \alpha & -d \sin \alpha \\ -\sin \alpha & -d \cos \alpha \end{pmatrix} \tag{26}$$

$$and \quad \Sigma_{d\alpha} = \begin{pmatrix} \sigma_d^2 & 0 \\ 0 & \sigma_\alpha^2 \end{pmatrix} \tag{27}$$

Then, the height estimate $\hat{h}$ can be updated from observation z_t, taken at time t, with the following Kalman filter:

$$\hat{h}(t) = \frac{1}{\sigma_{z_t}^2 + \sigma_{\hat{h}(t-1)}^2} \left(\sigma_{z_t}^2 \hat{h}(t-1) + \sigma_{\hat{h}(t-1)}^2 z_t \right) \tag{28}$$

$$\sigma_{\hat{h}(t)}^2 = \frac{1}{\frac{1}{\sigma_{\hat{h}(t-1)}^2} + \frac{1}{\sigma_{z_t}^2}}, \tag{29}$$

Equation 28 cannot be applied if the tilted LRF scans vertical structures since they lead to different height measurements for the same map location. For example, close to a wall the robot measures the upper part, far away from the wall the robot measures the lower part. We restrict the application of the Kalman Filter by the Mahalanobis distance. If the Mahalanobis distance between the estimate and the new observation is below a threshold c, the observation is considered to be within the same height. We use $c = 1$, which has the effect that all observations with a distance to the estimate that is below the variance $\sigma_{\hat{h}}^2$, are merged. Furthermore, we are mainly interested in the maximum height of a cell, since this is exactly what elevation maps represent. These constraints lead to the following update rules for cell height values:

$$\hat{h}(t) = \begin{cases} z_t & \text{if } z_t > \hat{h}(t) \wedge d_M\left(z_t, \hat{h}(t)\right) > c \\ \hat{h}(t-1) & \text{if } z_t < \hat{h}(t) \wedge d_M\left(z_t, \hat{h}(t)\right) > c \\ \frac{1}{\sigma_{z_t}^2 + \sigma_{\hat{h}(t-1)}^2}\left(\sigma_{z_t}^2 \hat{h}(t-1) + \sigma_{\hat{h}(t-1)}^2 z_t\right) & else, \end{cases} \tag{30}$$

and variance $\sigma_{\hat{h}(t)}^2$ with:

$$\sigma_{\hat{h}(t)}^2 = \begin{cases} \sigma_{z_t}^2 & \text{if } z_t > \hat{h}(t) \wedge d_M\left(z_t, \hat{h}(t)\right) > c \\ \sigma_{\hat{h}(t-1)}^2 & \text{if } z_t < \hat{h}(t) \wedge d_M\left(z_t, \hat{h}(t)\right) > c \\ \frac{1}{\frac{1}{\sigma_{\hat{h}(t-1)}^2} + \frac{1}{\sigma_{z_t}^2}} & else, \end{cases} \tag{31}$$

where d_M denotes the Mahalanobis distance, defined by:

$$d_M\left(z_t, \hat{h}(t)\right) = \sqrt{\frac{\left(z_t - \hat{h}(t)\right)^2}{\sigma_{\hat{h}(t)}^2}}. \tag{32}$$

The cell update introduced so far assumes perfect information on the global pose of the robot. However, since we integrate measurements from the robot while moving in the environment in real-time, we have to account for positioning errors from pose tracking that do accumulate over time [3]. We assume that the positioning error linearly grows with the accumulated distance and angle traveled. Hence, observations taken in the past lose significance with the distance the robot traveled after they were made. This can be reflected in the Kalman update by increasing the uncertainty of former height estimates according to the accumulated distance and angle:

$$\sigma_{\hat{h}(t)}^2 = \sigma_{\hat{k}(t)}^2 + \sigma_d^2 d(t-k) + \sigma_\alpha^2 \alpha(t-k), \tag{33}$$

where t denotes the current time, k denotes the time of the last height measurement at the same location, $d(t-k)$ and $\alpha(t-k)$ denotes the traveled distance and angle within the time interval $t-k$, and $\sigma_d^2, \sigma_\alpha^2$ are variances that have to be determined experimentally. Since it would be computational expensive to update the variances of all grid cells each time the robot moves, updates according to Equation 33 are only carried out on variances before they are utilized for a Kalman update with a new observation. The traveled distances can efficiently be generated by maintaining the integral functions $I_d(t)$ and $I_\alpha(t)$ that provided the accumulated distance and angle for each discrete time step t , respectively. Then, for example, $d(k-t)$ can be calculated by $I_d(k) - I_d(t)$. The integrals are represented by a table, indexed by time t with a fixed discretization, e.g. $\Delta t = 1s$.

7.2 Map filtering with a convolution kernel

The limited resolution of the LRF occasionally leads to missing data in the elevation map, e.g. conspicuous by surfaces holes. Furthermore, the effect of "mixed pixels", which frequently

[3]Note that global localization errors in the map can also be reduced by data association, i.e. by re-computing the elevation map based on a corrected trajectory, which, however, can usually not be applied in real-time.

happens if the laser beam hits edges of objects, whereas the returned distance measure is a mixture of the distance to the object and the distance to the background , might lead to phantom peaks within the elevation map (Ye and Borenstein, 2003). Therefore, the successively integrated elevation map has to be filtered.

In computer vision, filtering with a convolution kernel is implemented by the convolution of an input image with a convolution kernel in the spatial domain, i.e. each pixel in the filtered image is replaced by the weighted sum of the pixels in the filter window. The effect is that noise is suppressed and the edges in the image are blurred at the same time. We apply the same technique on the elevation map in order to reduce the errors described above. Hence, we define a convolution kernel of the size of 3×3 cells, whereas each value is weighted by its certainty and distance to the center of the kernel. Let $h(x+i, y+j)$ denote a height value relative to the kernel center at map location (x, y), with $i, j \in \{-1, 0, 1\}$. Then, the weight for each value is calculated as follows:

$$w_{i,j} = \begin{cases} \frac{1}{\sigma^2_{h(x+i,y+j)}} & \text{if } |i| + |j| = 0 \\ \frac{1}{2\sigma^2_{h(x+i,y+j)}} & \text{if } |i| + |j| = 1 \\ \frac{1}{4\sigma^2_{h(x+i,y+j)}} & \text{if } |i| + |j| = 2 \end{cases} \tag{34}$$

Consequently, the filtered elevation map h_f can be calculated by:

$$h_f\,(x, y) = \frac{1}{C} \sum_{i,j} h(x+i, y+j) w_{i,j}, \tag{35}$$

whereas $C = \sum w_{i,j}$.

7.3 3D Pose estimation

So far we have shown an incremental procedure for updating elevation map cells relative to the coordinate frame of the robot. In order to update map cells globally, the full 3D pose of the robot has to be considered, which is described by the vector $l = (x, y, h, \theta, \phi, \psi)^T$, where θ denotes the yaw angle, ϕ denotes the pitch angle, and ψ denotes the roll angle, respectively. We assume that IMU measurements of the three orientation angles are given with known variance. The position (x, y, h) is estimated by dead reckoning, which is based on the pitch angle and traveled distance measured by scan matching. However, since scan matching estimates the relative displacement δ on the 3D surface, displacement δ has to be projected onto the plane, as depicted by Figure 8. Given the input $u = (\theta, \phi, \delta)^T$, represented by the Gaussian distribution $N(\mu_u, \sigma_u)$, the projected position $l = (x^p, y^p, h^p)^T$, represented by the Gaussian distribution $N\,(\mu_l, \Sigma_l)$, can be calculated as follows:

$$\begin{pmatrix} x^p_t \\ y^p_t \\ h^p_t \end{pmatrix} = F_{lu} \begin{pmatrix} x^p_{t-1} \\ y^p_{t-1} \\ h_{t-1} \\ \phi \\ \theta \\ \delta \end{pmatrix} = \begin{pmatrix} x^p_{t-1} + \delta \cos\theta \cos\phi \\ y^p_{t-1} + \delta \sin\theta \cos\phi \\ h^p_{t-1} + \delta \sin\phi \end{pmatrix} \tag{36}$$

$$\Sigma_{lu} = \nabla F_{lu} \Sigma_{lu} \nabla F^T_{lu} \tag{37}$$

$$\Sigma_{lu} = \nabla F_l \Sigma_l \nabla F^T_l + \nabla F_u \Sigma_u \nabla F^T_u, \tag{38}$$

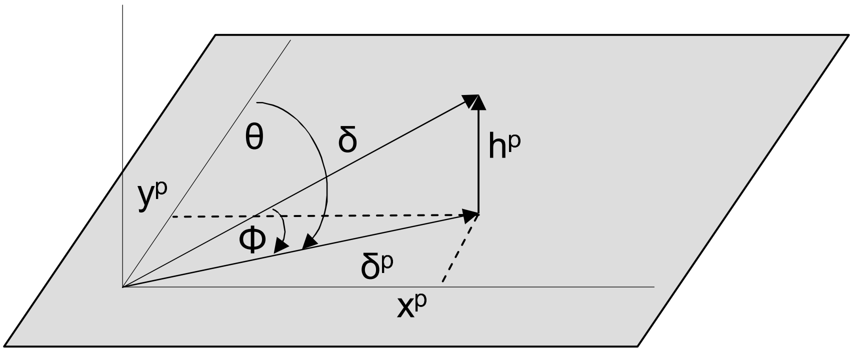


Figure 8. Dead reckoning of the projected Cartesian position (x_p, y_p, h_p) from yaw angle θ, pitch angle ϕ, and traveled distance δ.

where

$$\nabla F_l = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{39}$$

$$\nabla F_u = \begin{pmatrix} -\delta\cos\theta\sin\phi & -\delta\sin\theta\cos\phi & \cos\theta\cos\phi \\ -\delta\sin\theta\sin\phi & \delta\cos\theta\cos\phi & \sin\theta\cos\phi \\ \delta\cos\phi & 0 & \sin\phi \end{pmatrix}, \tag{40}$$

$$\Sigma_u = \begin{pmatrix} \sigma_\phi^2 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 \\ 0 & 0 & \sigma_\delta^2 \end{pmatrix}, \tag{41}$$

$$\Sigma_l = \begin{pmatrix} \sigma_{x^p}^2 & \sigma_{x^p y^p}^2 & \sigma_{x^p h^p}^2 \\ \sigma_{x^p y^p}^2 & \sigma_{y^p}^2 & \sigma_{y^p h^p}^2 \\ \sigma_{x^p h^p}^2 & \sigma_{y^p h^p}^2 & \sigma_{h^p}^2 \end{pmatrix} \tag{42}$$

Equation 36 allows to predict the current height of the robot. However, due to the accumulation of errors, the accuracy of the height estimate will continuously decrease. Therefore, it is necessary to update this estimate from direct observation. For this purpose, we utilize the height estimate $\left(\hat{h}(t), \sigma_{\hat{h}(t)}^2\right)$ at the robot's position from Equation 30 and 31, respectively. Then, the new estimate can be calculated by inserting $\left(\hat{h}(t), \sigma_{\hat{h}(t)}^2\right)$ and the predicted height estimate $(h^p, \sigma_{h^p}^2)$ into the Kalman filter shown by Equation 28.

The global location (x^g, y^g) of a measurement, i.e. the elevation map cell for which the height estimate $\hat{h}(t)$ will be updated, can be calculated straightforward by:

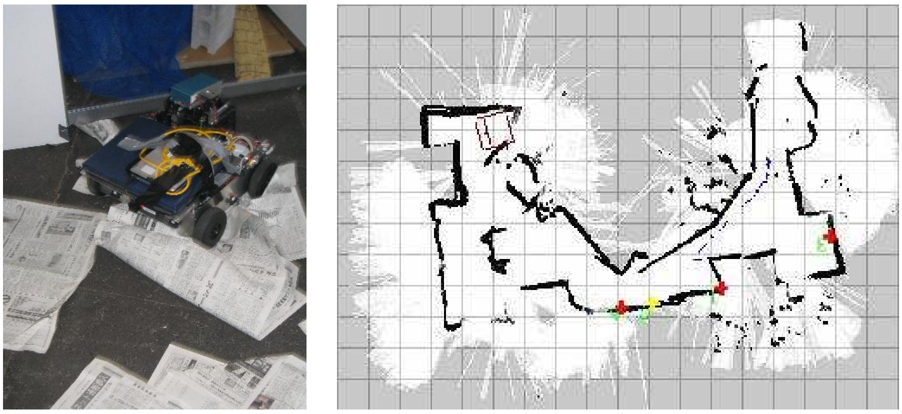
$$x^g = x^r + x^p \tag{43}$$

$$y^g = x^r + x^p \tag{44}$$

8 Experimental results

In this section we provide results from both simulated and real-robot experiments. All real-robot experiments have been carried out on the robot platforms described in Section 3 within testing arenas that are equal or similar to those proposed by NIST. In Sections 8.1 and 8.2 results from wheeled pose tracking and visual odometry-based pose tracking, respectively, are shown. In Section 8.3 we provide results from a RFID technology-based SLAM experiment in a cellar environment, and in Section 8.4 results from elevation mapping during the Rescue Robotics Camp 2006 in Rome, are presented.

8.1 Results from wheel odometry-based pose tracking



(a) (b)

Figure 9. *Zerg* robot during the final of the *Best in Class autonomy* competition at RoboCupRescue 2005 in Osaka: (a) slipping on newspapers and (b) the autonomously generated map. Red crosses mark locations of victims which have been found by the robot.

Figure 9 depicts the *Zerg* robot during the final of the "Best in Class Autonomy" competition, held in the NIST arena for Urban Search and Rescue (USAR) (Jacoff et al., 2001) during RoboCup 2005. In that scenario robots had to explore an unknown area within 20 minutes autonomously, to detect all victims, and finally to deliver a map sufficient for human teams to locate and rescue the victims. Conditions for exploration and SLAM were intentionally made difficult. For example, the occurrence of wheel spin was likely due to newspapers and cardboards covering the ground, which was partially made of steel and concrete. Stone bricks outside the robot's FOV caused the robot to get stuck, and walls made of glass caused the laser range finder to frequently return far readings. As shown in Figure 9, the system was able to cope with these difficulties and also to build a map reliably, augmented with victim locations detected by the robot. Finally, the system won the autonomy competition in 2005.

8.2 Results from visual odometry-based pose tracking

The approach of visual odometry has been extensively tested on both the tracked robot *Lurker*, operating on 3D obstacles, and the wheeled robot *Zerg*, operating on flat surfaces. The 2D setting has the advantage that results from the visual odometry system can directly be compared with ground truth data. We determine position ground truth from shaft encoder and IMU-based dead reckoning, as well as LRF-based scan matching. Ground truth on 3D obstacles was measured manually.

Table 1 gives an overview on the measured mean and standard deviation of the relative distance error from visual odometry and wheeled odometry on both robot platforms. Since the method has been mainly developed for tracked vehicles, the *Zerg* kinematics has been modified in order to be similar to that of the evaluated tracked vehicle, i.e. to allow only a subset of possible velocities, which are in case of the *Lurker* robot: *stop*, *forward*, and *backward*. The results clearly show that on the *Zerg* platform the visual odometry reaches

Run	Trav. dist. [m]	Vis. odo. [cm/m]	Wh. odo. [cm/m]
lab_1 (2D)	91.53	7.82 ± 1.84	6.17 ± 1.54
lab_2 (2D)	73.72	8.25 ± 2.46	7.59 ± 1.94
cellar (2D)	98.40	10.72 ± 4.68	11.77 ± 4.42
ramp (3D)	6.36	13.28 ± 9.2	-
palette (3D)	2.37	22.08 ± 8.87	-

Table 1. Relative error of the visual odometry and wheeled odometry compared to ground truth data (either manually measured for 3D runs or estimated from odometry and scan matching for 2D runs).

an accuracy comparable to the conventional odometry and thus could possibly replace it. In the cellar environment, the visual odometry turned out to be even superior, which can be explained by the higher degree of wheel slippage that we noticed within this environment. Figures 10 (a) and (b) depict the accumulation of the distance error of both the visual

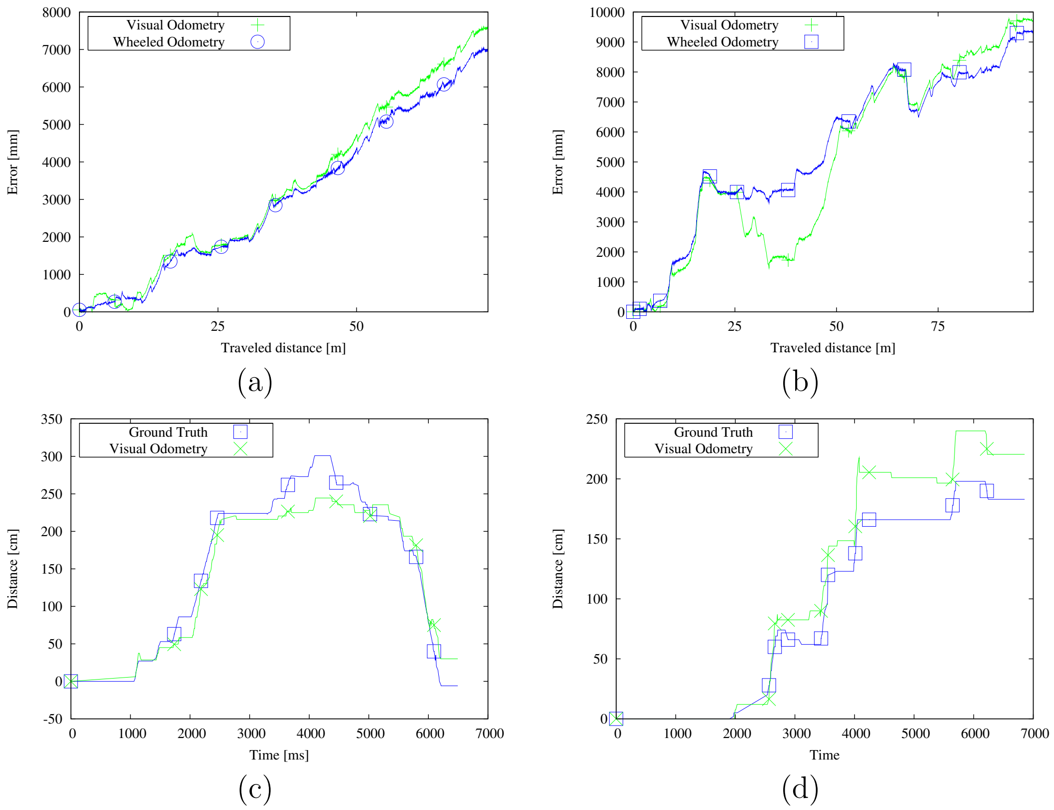


Figure 10. The accumulating distance error of the visual odometry method compared to ground truth data: (a) measured in the robotic lab, a $5m \times 5m$ squared area, (b) measured in a cellar of $15m \times 50m$. Results from driving forward and backward on a ramp (c), and climbing over a wooden palette (d). The blue curve indicates the manually measured ground truth, and the green curve indicates the distance estimated by visual odometry, respectively.

odometry and wheeled odometry, within the lab and cellar environment, respectively. The real advantage of the visual odometry, however, reveals if the robot operates on 3D obstacles. The results in Table 1 indicate that the introduced method, when applied while operating on 3D obstacles, provides usable measurements of the robot's motion. Figures 10 (c) and (d) depict the accumulation of the distance error during locomotion over 3D obstacles. The results indicate that, in case of tracked robots, the *tile coding* classification and voting applied to a simple kinematic model lead to sufficiently accurate results. From log files it has been determined that during the *cellar* run 87% (96%), the *ramp* run 81% (93%), and the *palette* run 94% (99%) of the classifications detected the correct motion of the robot, whereas numbers in brackets denote the voting-based improvement. While processing an image resolution of 320 × 240 on a $IntelPentiumM, 1.20GHz$, we measured an average processing time of $24.08 \pm 0.64ms$ for the complete processing without KLT feature tracking. This leads, together with the feature tracker, to a maximal frame rate of $5.34 \pm 1.17Hz$. If processing an image resolution of 160 × 120, the complete processing without KLT feature tracking needs $8.68 \pm 0.3ms$ and allows a total frame rate of $17.27 \pm 1.81Hz$. Experiments proposed in this paper were carried out with the higher resolution. However, experiments with the lower resolution showed that this results lead to a comparable accuracy.

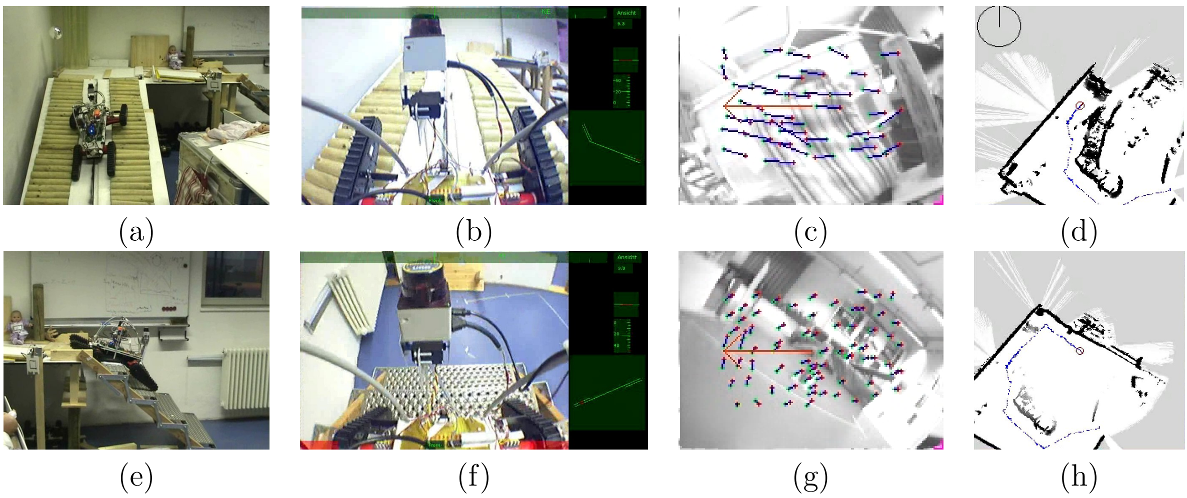


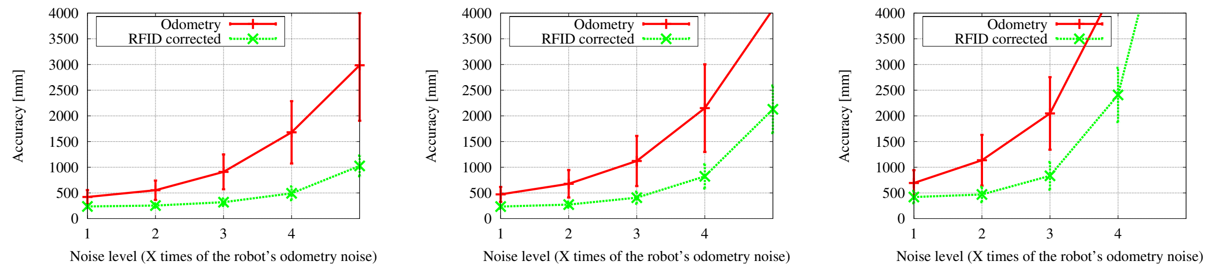
Figure 11. Lurker robot performing vision-based pose tracking while overcoming 3D obstacles: (a,e) the 3D obstacles, (b,f) the perspectives from the front camera, (c,g) the features and classified directions generated from a side camera, and (d,h) the grid maps generated at these locations, respectively.

Finally, we conducted a SLAM experiment on the Lurker robot by utilizing the visual odometry together with the scan matching algorithm. In this experiment the LRF sensor was automatically controlled by the measured pitch orientation of the robot in order to stay continuously within a horizontal position. This allows the LRF to perceive the environment independently from the robot's orientation, i.e. to return the same laser scan at the same location also if the orientation differers. The result is shown by the image series in Figure 11 (a-h). In (a) and (e) an overview on both obstacles is given, and in (b) and (f) the according perspective from the robot. Note that in (b) and (f) also the automatically adjustment of the LRF to the robot's pitch angle can be seen. Figure 11 (c,g) depicts the generated features, and Figure 11 (d,h) shows the generated maps, at the corresponding positions, respectively.

As shown by the "black wall" in front of the robot (Figure 11 (d)), there are situations in which the 2D LRF cannot provide sufficient evidence on the robot's motion. In this particular case measurements of the laser are nearly independent of the robot's location on the ramp, whereas the visual odometry (see Figure 11) (c) provides clear motion evidence. Note that the map representation shown in (d) and (h) does not suffice for the particular task since it does not distinguish between measurements of different height values at same location, i.e. parts of the map (d) have been deleted in the map (h). This problem can be solved by utilizing elevation maps, as shown in Section 7.

8.3 Results from RFID technology-based SLAM

The proposed method for RFID-based SLAM has extensively be tested with data generated by a simulator (Kleiner and Buchheim, 2003) as well as on the *Zerg* robot platform. The simulated robot explored three different building maps, a *small* map, *normal* map, and *large* map of the sizes $263m^2$, $589m^2$ $1240m^2$, respectively, while automatically distributing RFID tags in the environment. Figures 12 (a-c) show the averaged results from 100 executions of RFID-based SLAM on the three maps at five different levels of odometry noise. We



(a) (b) (c)

Figure 12. (a) - (c) Result from applying RFID-based SLAM at different levels of odometry noise on (a) the small map ($263m^2$), (b) the normal map ($589m^2$), and (c) the large map ($1240m^2$).

measured a computation time of 0.42 seconds on the small map, 2.19 seconds for the normal map, and 13.87 seconds for the large map, with a *Pentium4 2.4GHz*. The small map after and before the correction is shown in Figure 13 (b,d). For this result, the robot distributed approximately 50 RFID tags.

In another experiment we collected data from a real robot autonomously exploring in a cellar for 20 minutes while detecting RFID tags on the ground. The robot continuously tracked its pose as described in Section 4. As depicted by Figure 13 (a,c), the non-linear method successfully corrected the angular error. The correction was based on approximately 20 RFID tags.

8.4 Results from elevation mapping

Elevation mapping has been evaluated on a Lurker robot that was driving through a test arena consisting of rolls and ramps. The arena has been installed by NIST during the Rescue Robotics Camp 2006 in Rome. During all experiments the robot was equipped with a IMU

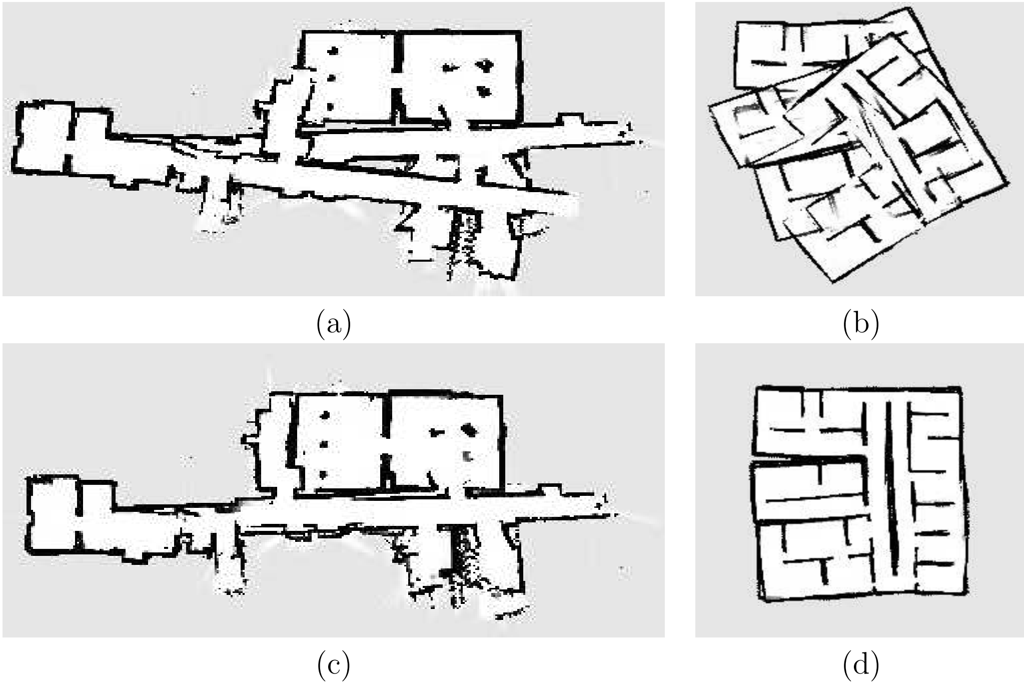


Figure 13. (a,c) Result from RFID-based SLAM on a robot driving in a cellar: (a) the noisy map and (c) the corrected map. (b,d) Result from applying the non-linear mapper to data generated in the simulation. (a) The small map with odometry noise and (b) the corrected map.

sensor, a side camera for visual odometry, and two LRFs, one for scan matching, and one for elevation mapping. The latter sensor has been tilted downwards by 35°.

Figure 14 depicts the Kalman filter-based pose estimation of the robot, as described in Section 7.3. For this experiment, conditions have been made intentionally harder. Map smoothing has been turned off, which had the effect that missing data, due to a limited resolution of 2D scans, lead to significant holes on the surface of the map. Furthermore, we added a constant error of $-2°$ to pitch angle measurements of the IMU. As shown in Figure 14, the Kalman filter was nevertheless able to deal with these errors, and finally produced a trajectory close to ground truth.

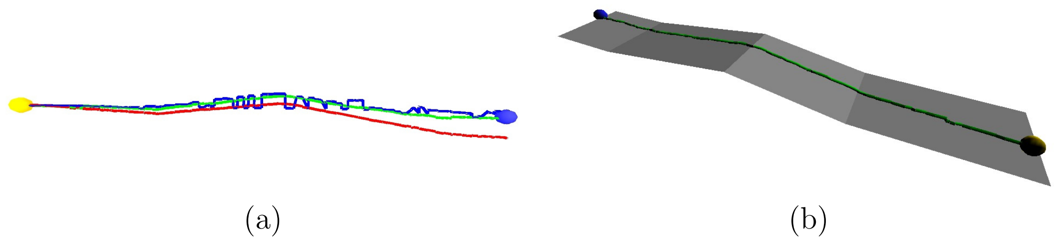
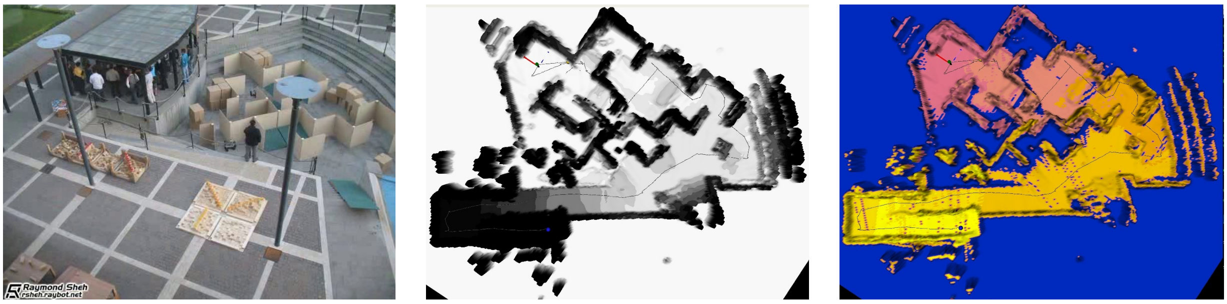


Figure 14. Evaluation of the efficiency of the Kalman filter for estimating the robot's height. (a) Height values predicted from the IMU (red line) are merged with height values taken from the generated map (blue line). Errors from inaccuracies in the map, as well as a simulated continuously drift error of the IMU sensor are successfully reduced (green line). (b) Merged trajectory compared to ground truth (grey ramp).

Figure 15 and 16 show the final result from applying the proposed elevation mapper during



(a) (b) (c)

Figure 15. Elevation mapping during the Rescue Robotics Camp 2006 in Rome: (a) The arena build-up by NIST, (b) The corresponding digital elevation model (DEM), build by the *lurker* robot, going from white (low altitude) to black (hight altitude). (c) The variances of each height value, going from pink (hight variance) to yellow (low variance).

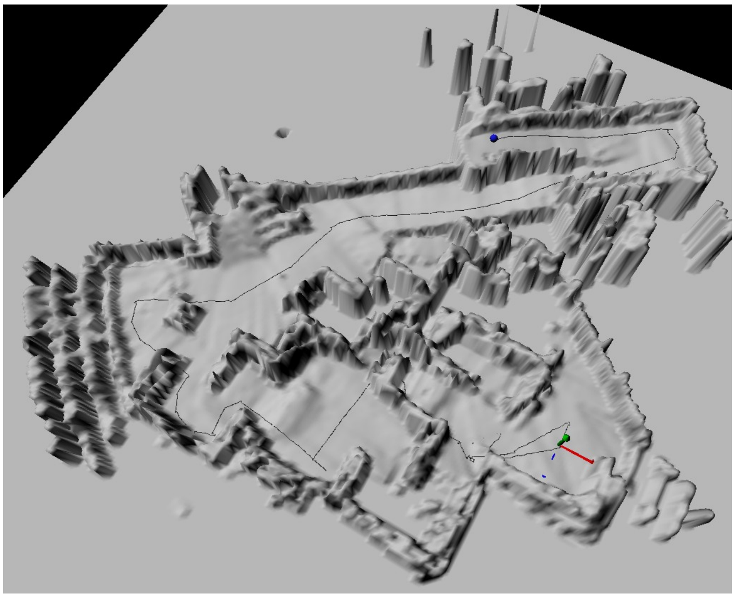
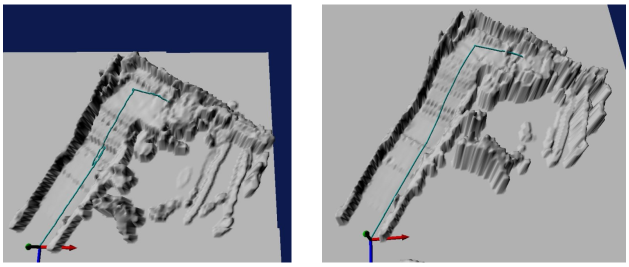


Figure 16. Elevation mapping during the Rescue Robotics Camp 2006 in Rome: 3D perspective.

the Rescue Robotics camp. Figure 15 (a) depicts an overview on the arena, and Figure 15 (b) shows the calculated height values, whereas the height of each cell is indicated by a gray value, as darker the cell, as bigger the elevation. Figure 15 (c) depicts the variance of each height cell, going from pink (hight variance) to yellow (low variance), whereas the current position of the robot is indicated by a blue circle in the lower left corner. As more far away cell updates on the robot's trajectory, as lower their variance (see Section 7.1). Figure 16 shows a 3D visualization of the generated elevation map. Structures, such as the long ramp at the end of the robot's trajectory, and the stairs, can clearly be identified. We measured on a $AMD64X23800+$ a total integration time (without map smoothing) of $1.88 \pm 0.47ms$ for a scan measurement with 683 beams, including $0.09 \pm 0.01ms$ for the 3D pose estimation. Map smoothing has generally the time complexity of $O\left(N^2M\right)$, where N is the number of rows and column of the map and M the size of the kernel. We measured on the same architecture $34.79 \pm 14.84ms$ for smoothing a map with $N = 300$ and $M = 3$. However, this can be significantly be improved during runtime by only smoothing recently modified map cells and their immediate neighbors within distance M.



(a) (b)

Figure 17. Comparing elevation mapping based on scan matching only (a) and scan matching combined with visual odometry (b). Scan matching without visual odometry support does not correctly reflect the true length of the ramp, insufficient motion evidence causes the map to be partially compressed.

The last experiment demonstrates the influence of visual odometry on elevation mapping. Figure 17 depicts two elevation maps of the same ramp, on the one hand, with support of visual odometry, and on the other hand, without. Mapping based on scan matching only, yields a compressed map since in this environment 2D laser scans do not provide sufficient information on the motion of the robot, whereas the map generated based on visual odometry reveals the true size of the ramp.

9 Conclusion

We proposed solutions to the problems of vision-based pose tracking on tracked vehicles, the building of globally consistent maps based on a network of RFID tags, and the building of elevation maps from readings of a tilted Laser Range Finder (LRF). The experimental results showed that these methods lead under modest computational requirements to good results within the testing arenas proposed by NIST for Urban Search and Rescue. Although challenging for methods currently developed in robotics, the NIST benchmark does not yet capture the whole magnitude of problems that robots encounter after a real disaster. Here the plan is to continuously increase the difficulty year by year in order to promote stepwise research in the field of Rescue Robotics. Currently, methods from robotics for Urban Search and Rescue are just in the beginning, and up to now, robots that have been deployed after a real disaster were mainly teleoperated by human beings.

We showed that RFID-based SLAM allows the fast generation of maps without explicit need for communication. This has the advantage that, particularly in the context of disaster response, this method can also be applied if communication is disturbed by building debris and radiation. The practical advantage is that human rescue teams can easily be integrated into the search. They can receive data from the RFIDs with a PDA and thus localize themselves within the map and also leave information, related to the search or to victims, behind. Furthermore, they can easily reach victims, which have been found by the robots, by following plans consisting of RFID tags and directions. The idea of labeling locations with information that is important to the rescue task, has already be applied in practice. During the disaster relief in New Orleans in 2005, rescue task forces marked buildings with information concerning, for example, hazardous materials or victims inside the buildings. A more detailed description can be found in a document published by the U.S Dep. of Homeland Security (of Homeland Security, 2003). The autonomous RFID-based marking of locations is a straight forward extension of this concept. In future work we will evaluate the usability of our method for localizing first responders equipped with a Personal Dead Reckoning Module (PDRM), which consists of an IMU sensor and a acceleration measurement based step counter.

We believe that elevation maps provide the right trade-off between computational complexity and expressiveness. We have shown that they can be reliably generated in real-time while the robot is continuously in motion. In future work we will deal with the classification of map cells according to whether they are drivable, climbable, or impassable (obstacle negotiation). Furthermore, we will close the loop by adding a planner component, allowing the robot to autonomously explore the environment by choosing actions according to the classified terrain complexity.

Acknowledgments

The authors thank the many enthusiastic students that contributed to the development of the system architecture. Rainer Kümmerle, Daniel Meyer-Delius, Johann Prediger, Michael Ruhnke, and Bastian Steder of the University of Freiburg.

References

AlienTechnology (2003). Alien technology. `http://http://www.alientechnology.com/`.

Birchfeld, S. (1996). Derivation of kanade-lucas-tomasi tracking equation. `http://www.ces.clemson.edu/~stb/klt/`.

Bohn, J. and Mattern, F. (2004). Super-distributed rfid tag infrastructures. In *Proceedings of the 2nd European Symposium on Ambient Intelligence (EUSAI 2004)*, number 3295 in Lecture Notes in Computer Science (LNCS), pages 1–12, Eindhoven, The Netherlands. Springer-Verlag. Springer-Verlag.

Corke, P., Strelow, D., and Singh, S. (2004). Omnidirectional visual odometry for a planetary rover. In *Proceedings of IROS 2004*.

Dornhege, C. and Kleiner, A. (2006). Visual odometry for tracked vehicles. In *Proc. of the IEEE Int. Workshop on Safty, Security and Rescue Robotics (SSRR)*, Gaithersburg, Maryland, USA.

Gassmann, B., Frommberger, L., Dillmann, R., and Berns, K. (2003). Real-time 3d map building for local navigation of a walking robot in unstructured terrain. In *IROS*, volume 3, pages 2185–2190.

Hähnel, D. (2005). *Mapping with Mobile Robots*. Dissertation, Universität Freiburg, Freiburg, Deutschland.

Hähnel, D., Burgard, W., Fox, D., Fishkin, K., and Philipose, M. (2004). Mapping and localization with rfid technology. In *In Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*.

Helmick, D., Chang, Y., Roumeliotis, S., Clouse, D., and Matthies, L. (2004). Path following using visual odometry for a mars rover in high-slip environments. In *IEEE Aerospace Conference*.

Hitachi (2003). *Mu Chip Data Sheet*. http://www.hitachi-eu.com/mu/products/mu_chip_data_sheet.pdf.

Jacoff, A., Messina, E., and Evans, J. (2001). Experiences in deploying test arenas for autonomous mobile robots. In *Proceedings of the 2001 Performance Metrics for Intelligent Systems (PerMIS) Workshop*.

Kantor, G., Singh, S., Peterson, R., Rus, D., Das, A., Kumar, V., Pereira, G., and Spletzer, J. (2003). Distributed search and rescue with robot and sensor team. In *Proceedings of the Fourth International Conference on Field and Service Robotics*, pages 327–332. Sage Publications. In draft proceedings distributed only at the conference.

Kleiner, A. and Buchheim, T. (2003). A plugin-based architecture for simulation in the F2000 league. In *Proc. Int. RoboCup Symposium '03*. Padova, Italy.

Kleiner, A., Prediger, J., and Nebel, B. (2006). Rfid technology-based exploration and slam for search and rescue. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*.

Krotkov, E. and Hoffman, R. (1994). Terrain mapping for a walking planetary rover. *IEEE-TRANS*, 10:728–739.

Logitech (2006). *Logitech QuickCam Pro 4000*. http://www.logitech.com/index.cfm/products/details/US/EN,CRID=2204,CONTENTID=5042.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Auton. Robots*, 4:333–349.

Maybeck, P. S. (1979). *Stochastic models, estimation, and control*, volume 141 of *Mathematics in Science and Engineering*.

Milella, A. and Siegwart, R. (2006). Stereo-based ego-motion estimation using pixel tracking and iterative closest point. In *IEEE International Conference on Computer Vision Systems ICVS '06*, pages 21– 21.

Miller, L., Wilson, P. F., Bryner, N. P., Francis, Guerrieri, J. R., Stroup, D. W., and Klein-Berndt, L. (2006). Rfid-assisted indoor localization and communication for first responders. In *Proc. of the Int. Symposium on Advanced Radio Technologies*.

Moravec, H. (1988). Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74.

Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, volume 1, pages 652–659.

of Homeland Security, U. D. (2003). *National Urban Search and Rescue (USR) Response System - Field Operations Guide*.

Pfaff, P., Triebel, R., and Burgard, W. (2005). An efficient extension to elevation maps for outdoor terrain mapping and loop closing. In *International Journal of Robotics Research (IJRR) - Special Issue of International Conference on Field and Service Robotics (FSR)*.

Quinlan, J. R. (2003). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning).* The MIT Press.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P. (2006). Winning the darpa grand challenge. *JFR*. accepted for publication.

Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University.

Wolf, D., Sukhatme, G., Fox, D., and Burgard, W. (2005). Autonomous terrain mapping and classification using hidden markov models. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*.

Ye, C. and Borenstein, J. (2003). A new terrain mapping method for mobile robots obstacle negotiation. In Gerhart, G. R., Shoemaker, C. M., and Gage, D. W., editors, *Unmanned Ground Vehicle Technology V.*, volume 5083, pages 52–62.

Ziparo, V., Kleiner, A., Nebel, B., and Nardi, D. (2007). Rfid-based exploration for large robot teams. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*. to appear.