Subterranean positioning for a semi-autonomous robot supporting emergency task forces

Eva Reitbauer
Institute of Geodesy
Graz University of Technology
Graz, Austria
eva.reitbauer@tugraz.at

Christoph Schmied
Institute of Geodesy
Graz University of Technology
Graz, Austria
schmied@tugraz.at

Hamid Didari
Institute of Software Technology
Graz University of Technology
Graz, Austria
hamid.didari@ist.tugraz.at

***Abstract*—This paper proposes a positioning algorithm for a semi-autonomous robot in subterranean scenarios. The robot is equipped with positioning sensors, imaging sensors, and sensors to detect hazardous materials. The sensors can be used to automatically generate a site map to increase safety for emergency forces. To create an accurate map, the position and attitude of the robot have to be determined. This is done using an extended Kalman filter which fuses data from LIDAR, wheel odometry, and a MEMS IMU. Tests were carried out in a tunnel in Eisenerz, Austria. To evaluate the achievable accuracy, the estimated position of the filter is compared to a ground truth. The results show that with the developed sensor fusion algorithm, a horizontal positioning error of 1.07% of the traveled distance can be achieved.**

***Index Terms*—subterranean positioning, semi-autonomous robot, extended Kalman filtering, odometry, IMU, LIDAR, GICP**

I. Introduction

Accidents in tunnels our underground infrastructures are extremely demanding for emergency task forces. Poor visibility, smoke, high temperatures, released hazardous materials and structural hazards are among the factors that not only challenge the emergency personnel, but also push equipment and devices to a limit.

A study by Njå and Svela (2018) [1] investigates concerns about tunnel fire safety from the first responders' perspective. It analyses that there are many uncertainties for firefighters when entering a tunnel after an accident: there might be no means of communication or it may be unclear whether the ventilation system works. Furthermore, there are uncertainties regarding the fire intensity and toxicity, since it is unclear which (potentially hazardous) goods were transported into the tunnel before the accident.

One approach to assist first responders during these dangerous and time-sensitive operations is to rapidly map the environment using robots, which was also the goal of the DARPA Subterranean Challenge [2]. A similar approach is taken by the research project ROBO-MOLE: here, a wheeled robot is equipped with positioning sensors, imaging sensors, and sensors to detect hazardous materials. The aim of the project is to automatically generate a site map so that emergency task forces get a rapid and clear overview of the current situation. To accurately map the environment, the position and attitude of the robot have to be determined.

The position of a robot can be estimated in various ways depending on what type of sensors are in use. Each sensor type (e.g., laser scanner, radar, camera, GNSS, etc.) has advantages and disadvantages. While Global Navigation Satellite Systems (GNSS) provide high absolute accuracy in outdoor scenarios, they cannot be used in subterranean environments and tunnels.

Typically, methods of position fixing and dead reckoning are combined [3] to optimally estimate the position of a robot. Among the technologies that can be used for position fixing in tunnels are WiFi, RFID, Ultra-Wideband (UWB) [4]. However, all these technologies depend on infrastructure (e.g. power supply), which might not be working in the case of a tunnel fire.

The selection of positioning sensors heavily depends on the environment the sensor should be used in. This work focuses on tunnels, which typically have low lighting conditions. Additionally, there will be smoke in the case of tunnel fires, which hinders the use of cameras. In that regard laser scanners have proven to be a reliable technology to use in underground scenarios. While a laser scanner can cope with problems like smoke and bad lighting conditions, the uniform structure of a tunnel means that data from different locations are very homogeneous and hard to distinguish.

A study by Zhao et al. (2021) [5] investigated the performance of different LIDAR Visual-Inertial-Odometry algorithms in challenging environments. The authors tested LOAM [6], LIO-SAM [7], VINS-Mono [8], Depth-enhanced VINS and Super-Odometry [5] in a long corridor in an apartment building. Similar to a tunnel, a long corridor lacks distinct geometric features and is challenging for LIDAR-based algorithms. The maximum error of LOAM was 9.44 m, of LIO-SAM 6.52 m, of VINS-Mono 9.02 m and of VINS-Depth 6.15 m. Super-Odometry, the algorithm that won the DARPA Subterranean Challenge, had a maximum error of 0.35 m.

Studies that focus on high-precision navigation of autonomous robots with laser scanners in tunnels ([9], [10]) use Simultaneous Localization and Mapping (SLAM) in combination with reflective beacons every 15 m to improve precision. Again, in the case of a tunnel accident we cannot assume to have beacons available.

To support the laser scanner, we can use information of dead reckoning sensors, which are infrastructure-independent. Although dead reckoning sensors have a high short-term accuracy, they have a poor long-term accuracy, which means that they start to drift with time or distance covered. In this study, we use an Inertial Measurement Unit (IMU) as well as the steering angles and wheel velocities to obtain odometry information.

The aim of the paper is to develop an Extended Kalman Filter that fuses data from a laser scanner, wheel odometry, and an IMU to estimate the position, velocity and attitude of a mobile robot. The developed algorithms are tested in a real tunnel scenario and evaluated regarding the achievable accuracy for position and attitude.

The paper is structured as follows: in section II we describe our Methodology, where we first give an overview on the used robot and navigation sensors in subsection II-A. We then explain the Extended Kalman Filter (EKF) and how we process the navigation sensor data to estimate position, velocity and attitude of the robot in subsection II-B. Subsection II-C describes the software development in ROS and subsection II-D gives an overview of the field tests. In section III we present the results and analyse which positioning accuracy can be achieved by comparing the estimated position to a ground truth. In section IV we discuss the results and conclude the paper.

II. Methodology

A. Robot and Positioning Sensors

A custom-built robot [11] by the Institute of Software Technology of Graz University of Technology is used as a mobile platform (see Figure 1). The robot has four wheels that can be steered independently.

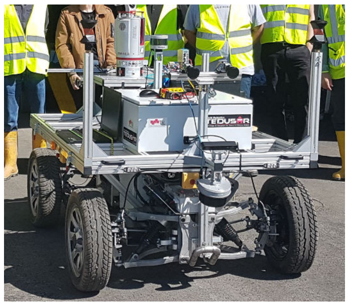

Fig. 1. Picture of the wheeled robot used in this study. Photo by Disaster Competence Network Austria (DCNA).

The following positioning sensors are mounted on the robot and are used in this study to determine the robot's position in a subterranean environment: a LIDAR by Velodyne (Velodyne Puck, VLP-16) and an Inertial Measurement Unit (IMU) by XSens (XSens MTi-G-710). To obtain the steering angles and wheel speeds that are needed to compute odometry information, readings are taken from sensors that are directly installed on the robot. For the steering angles, the positions of the linear actuators are measured and converted to angles; for the wheel speeds, the rotary encoders (3 Hall sensors within the motors of the wheels) are used to measure the revolutions, which are then converted to speeds in metres per second.

B. Extended Kalman Filter for Estimating the Position and Attitude of the Robot

To estimate the position and attitude of the robot, we use an Extended Kalman Filter (EKF) to fuse measurements from the laser scanner, wheel encoders, and the IMU. A Kalman filter is a Bayesian filter which estimates the state vector in three recursive steps, which are explained in the following. In the first step, the so-called time update or prediction step, the estimated state vector $\hat{\mathbf{x}}$ and its corresponding covariance matrix $\mathbf{P}$ of the previous epoch $k-1$ are predicted to the current epoch k with

$$\tilde{\mathbf{x}}_k = \boldsymbol{\varphi}_{k-1}(\hat{\mathbf{x}}_{k-1}), \tag{1}$$

$$\widetilde{\mathbf{P}}_k = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}\mathbf{\Phi}_{k-1}^T + \mathbf{Q}_{k-1}. \tag{2}$$

Here, $\tilde{\mathbf{x}}_k$ and $\widetilde{\mathbf{P}}_k$ are the predicted state vector and predicted covariance matrix at the current epoch k. $\boldsymbol{\varphi}$ is a vector containing the dynamic functions [12] describing how the state changes from epoch $k-1$ to k. $\mathbf{\Phi}_{k-1}$ contains the linearised dynamic functions with respect to the estimated state vector $\hat{\mathbf{x}}$ of the previous epoch $k-1$. $\mathbf{Q}$ is the system noise covariance matrix.

In the second step, called gain computation, the Kalman gain matrix is computed from

$$\mathbf{K}_k = \widetilde{\mathbf{P}}_k\mathbf{H}_k^T\left(\mathbf{H}_k\widetilde{\mathbf{P}}_k\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}, \tag{3}$$

where $\mathbf{H}_k$ contains the linearised observation functions with respect to the predicted state vector $\tilde{\mathbf{x}}$ at epoch k and $\mathbf{R}_k$ is the covariance matrix containing the observation noise. Furthermore, the reduced observations $\delta\mathbf{z}_k$ are computed from

$$\delta\mathbf{z}_k = \mathbf{H}_k\tilde{\mathbf{x}}_k - \mathbf{z}_k, \tag{4}$$

where $\mathbf{z}_k$ are the observations or measurements at epoch k.

In the third step of the Kalman filter, called measurement update or correction step, the error state vector is computed from

$$\delta\hat{\mathbf{x}}_k = \mathbf{K}_k\delta\mathbf{z}_k, \tag{5}$$

and then used to update the state vector by

$$\hat{\mathbf{x}}_k = \tilde{\mathbf{x}}_k - \delta\hat{\mathbf{x}}_k. \tag{6}$$

The covariance matrix of the state vector is updated by

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\widetilde{\mathbf{P}}_k. \tag{7}$$

In our study, we use the described EKF to estimate the error state vector of the robot. We define the error state vector as

$$\delta\mathbf{x} = \begin{pmatrix} \delta\boldsymbol{\rho} & \delta\mathbf{v} & \delta\mathbf{e} & \delta\mathbf{b_g} \end{pmatrix}^T, \tag{8}$$

where $\delta\boldsymbol{\rho}$ stands for the position errors ($\delta\varphi$, $\delta\lambda$, δh), $\delta\mathbf{v}$ stands for the velocity errors in a North-East-Down-frame, $\delta\mathbf{e}$ contains the attitude errors in Euler angle (roll, pitch, yaw) notation, and $\delta\mathbf{b_g}$ is the gyro bias error.

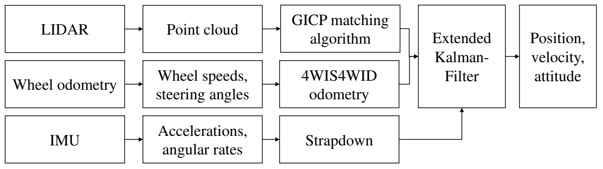

Fig. 2. Filter architecture for LIDAR, wheel odometry and IMU data to estimate position and attitude of the robot.

To fuse information from all sensors, we use a cascaded filter architecture [3], which is depicted in Figure 2. Here, wheel odometry and LIDAR are used as aiding sensors, which means they are used in the measurement update or correction step of the EKF. The IMU is used as the reference navigation sensor, i.e. it is used in the time update or prediction step to propagate the state vector. How the measurements are preprocessed before being inserted into the filter will be explained in the following.

1) LIDAR as Aiding Sensor: One way of estimating the robot's pose is utilising the point cloud from the laser scanner. A transformation matrix ($\mathbf{T}$) can be estimated by comparing the current point cloud to the previous one. Multiplying these transformation matrices over time will result in the estimated pose of the robot ($\mathbf{X}$) at epoch k:

$$\mathbf{X}_k = \prod_{i=0}^{k} \mathbf{T}_i. \tag{9}$$

Iterative Closest Point (ICP) was introduced as a method of matching a source point cloud (P) and a target point cloud (Q) [13]. First, ICP finds the corresponding points ($\mathbf{p_i}$ and $\mathbf{q_i}$) in P and Q. Then, it estimates the optimal transformation $\mathbf{T}$ to minimise the distance between matched pairs of points. These two steps are executed iteratively until some termination criterion is satisfied. Two variations of ICP are commonly used: ICP point-to-point and ICP point-to-surface. These variations have different cost functions.

The ICP point-to-point extracts the corresponding points with the nearest neighbor method. Then, $\mathbf{T}$ can be estimated by utilising Singular Value Decomposition (SVD) [14] to minimise the cost function f_{p2p}:

$$f_{p2p} = \sum_{i=1}^{N} ||\mathbf{p}_i - \mathbf{T}\mathbf{q}_i||^2, \tag{10}$$

where N is the number of points in the point cloud. Outlier points result in a noticeable noise with this cost function. The algorithm's sensitivity to noise can be reduced by using the local neighbourhood in the target point cloud [15]. In the ICP point-to-surface method, the local neighbourhood points of $\mathbf{q_i}$ are marked as a surface to be used instead. The normal vector $\mathbf{n_i}$ of a surface can be extracted with the help of the eigenvectors of the covariance matrix $\mathbf{C_i}$ of the surface.

Now, instead of minimising the distance between corresponding points, the algorithm tries to minimise the projection of the distance onto the local surface. The cost function for ICP point-to-surface is given in the Equation 11.

$$f_{p2s} = \sum_{i=1}^{N} ||(\mathbf{p}_i - \mathbf{T}\mathbf{q}_i)\mathbf{n}_i|| \tag{11}$$

Using a surface for target point cloud improves the robustness against noise, but the ICP point-to-surface assumes a noise-free source point cloud. A real-world scenario would have noise in both source and target point clouds. Thus, to adequately counter the effect of noise, Generalised-ICP (GICP) matches surface $\tilde{\mathbf{p}}_\mathbf{i}$ to $\tilde{\mathbf{q}}_\mathbf{i}$ [16]. These surfaces are defined by their covariance matrices, calculated based on neighbourhood points as given in

$$\begin{array}{c} \tilde{\mathbf{p}}_\mathbf{i} \sim \mathcal{N}(\hat{\mathbf{p}}_\mathbf{i}, \mathbf{C}_\mathbf{i}^\mathbf{p}) \\ \tilde{\mathbf{q}}_\mathbf{i} \sim \mathcal{N}(\hat{\mathbf{q}}_\mathbf{i}, \mathbf{C}_\mathbf{i}^\mathbf{q}) \end{array}, \tag{12}$$

where $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$ are the mean value for the neighbourhood points. So, the transformation error can be defined as:

$$\mathbf{d_i} = \tilde{\mathbf{p}}_\mathbf{i} - \mathbf{T}\tilde{\mathbf{p}}_\mathbf{i}. \tag{13}$$

The distribution of $\mathbf{d_i}$ can be calculated as:

$$\begin{aligned} \mathbf{d_i} &\sim \mathcal{N}(\hat{\mathbf{q}}_\mathbf{i} - \mathbf{T}\hat{\mathbf{p}}_\mathbf{i},\ \mathbf{C}_\mathbf{i}^\mathbf{q} + \mathbf{T}\mathbf{C}_\mathbf{i}^\mathbf{p}\mathbf{T}^\mathbf{T}) \\ &= \mathcal{N}(\mathbf{0},\ \mathbf{C}_\mathbf{i}^\mathbf{q} + \mathbf{T}\mathbf{C}_\mathbf{i}^\mathbf{p}\mathbf{T}^\mathbf{T}) \end{aligned} \tag{14}$$

The cost function for GICP is given in the Equation 15.

$$f_{GICP} = \sum_{1} \mathbf{d}_\mathbf{i}^\mathbf{T} (\mathbf{C}_\mathbf{i}^\mathbf{q} + \mathbf{T}\mathbf{C}_\mathbf{i}^\mathbf{p}\mathbf{T}^\mathbf{T})^{-1} \mathbf{d_i} \tag{15}$$

This paper utilises GICP to improve robustness against noise in a more realistic scenario.

The estimated pose from Equation 9 contains the position and attitude of the robot:

$$\mathbf{X}_k = \begin{pmatrix} \boldsymbol{\rho}_{GICP,k} \\ \mathbf{e}_{GICP,k} \end{pmatrix} \tag{16}$$

The reduced observations for the measurement update of the Kalman filter are computed from

$$\delta\mathbf{z}_{GICP,k} = \begin{pmatrix} \tilde{\boldsymbol{\rho}}_{IMU,k} - \boldsymbol{\rho}_{GICP,k} \\ \tilde{\mathbf{e}}_{IMU,k} - \mathbf{e}_{GICP,k} \end{pmatrix}, \tag{17}$$

where $\tilde{\boldsymbol{\rho}}_{IMU,k}$ is the predicted position and $\tilde{\mathbf{e}}_{IMU,k}$ is the predicted attitude at epoch k. $\boldsymbol{\rho}_{GICP,k}$ is the computed position from GICP and $\mathbf{e}_{GICP,k}$ is the computed attitude from GICP at epoch k.

The design matrix H for the measurement update with GICP has the following form:

$$\mathbf{H}_{GICP,k} = \begin{pmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{pmatrix}. \tag{18}$$

$\mathbf{H}_{GICP,k}$ is used in Equation 3 to compute the Kalman gain. The measurement noise matrix $\mathbf{R}_{GICP,k}$ is modelled as a diagonal 6×6-matrix with the variances $\sigma_x^2 = \sigma_y^2 = (1.4[m])^2$, $\sigma_z^2 = (10[m])^2$ for position and $\sigma_e^2 = (0.032[rad])^2$ for the Euler angles. The gain matrix and the reduced observations in Equation 17 are then used to estimate the error state vector with Equation 5.

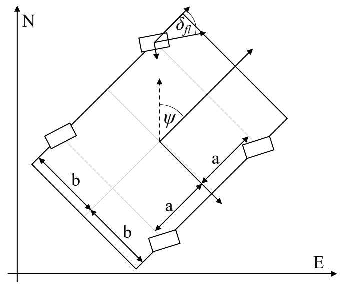

Fig. 3. Model of a 4WIS4WID robot. The wheel positions in the robot's body frame can be determined from the wheelbase (2a) and the trackwidth (2b).

2) Odometry as Aiding Sensor: When odometry is used to compute position changes or velocity of a robot, a suitable kinematic model has to be found. The robot used in this study is a four-wheel-independent steering and four-wheel-independent driving (4WIS4WID) robot, which means that all four wheels can be steered independently. We can measure the individual steering angles $\delta_{fl}, \delta_{fr}, \delta_{rl}, \delta_{rr}$, as well as the individual wheel speeds $v_{fl}, v_{fr}, v_{rl}, v_{rr}$, where the indices indicate the position of the wheel in the body frame of the robot: front left (fl), front right (fr), rear left (rl), and rear right (rr).

We use a model proposed by [17] to compute the heading rate ($\dot{\psi}$) as well as the velocities (v_N, v_E) of the robot in a North-East-Down frame:

$$\begin{pmatrix} v_{N,odo} \\ v_{E,odo} \\ \dot{\psi}_{odo} \end{pmatrix} = \mathbf{J} \cdot \mathbf{v_{wheels}}, \tag{19}$$

with

$$\mathbf{v_{wheels}} = \begin{pmatrix} v_{fl} & v_{fr} & v_{rl} & v_{rr} \end{pmatrix}^T, \tag{20}$$

and

$$\mathbf{J} = \begin{pmatrix} \frac{\cos(\delta_{fl}+\psi)}{4} & \frac{\sin(\delta_{fl}+\psi)}{4} & \frac{b\cos(\delta_{fl})+a\sin(\delta_{fl})}{4a^2+4b^2} \\ \frac{\cos(\delta_{fr}+\psi)}{4} & \frac{\sin(\delta_{fr}+\psi)}{4} & \frac{-b\cos(\delta_{fr})+a\sin(\delta_{fr})}{4a^2+4b^2} \\ \frac{\cos(\delta_{rl}+\psi)}{4} & \frac{\sin(\delta_{rl}+\psi)}{4} & \frac{b\cos(\delta_{rl})-a\sin(\delta_{rl})}{4a^2+4b^2} \\ \frac{\cos(\delta_{rr}+\psi)}{4} & \frac{\sin(\delta_{rr}+\psi)}{4} & \frac{-b\cos(\delta_{rr})-a\sin(\delta_{rr})}{4a^2+4b^2} \end{pmatrix}^T, \tag{21}$$

where a is half of the robot's wheel base and b is half of the robot's trackwidth. a and b determine the coordinates of the wheels in the robots body frame (see Figure 3): the front left wheel has the coordinates $(a, -b)$, the front right wheel (a, b), the rear left wheel $(-a, -b)$, and the rear right wheel $(-a, b)$.

In the next step, the reduced observations for the Kalman filter are computed from

$$\delta\mathbf{z}_{odo,k} = \begin{pmatrix} \tilde{v}_{N,IMU,k} - v_{N,odo,k} \\ \tilde{v}_{E,IMU,k} - v_{E,odo,k} \\ \tilde{v}_{D,IMU,k} - 0 \\ -\dot{\psi}_{odo} \cdot \delta t \end{pmatrix}, \tag{22}$$

where $\tilde{v}_{N,IMU,k}$ is the north-component of the predicted velocity from the IMU data at epoch k and $v_{N,odo,k}$ is the north-component of the velocity computed from odometry with Equation 19 at epoch k. The subscripts E and D indicate the east- and down-component of the velocity vector. Note that no down-component of velocity can be computed from odometry, so a zero is used as a pseudo-observation. The fourth row of the reduced observations vector in Equation 22 only contains the negative integrated heading rate from odometry ($-\dot{\psi}_{odo} \cdot \delta t$) since this term already describes the difference to the prediction.

The design matrix for the measurement update odometry data has the following form:

$$\mathbf{H}_{odometry,k} = \begin{pmatrix} \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{1\times3} & \mathbf{0}_{1\times3} & \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} & \mathbf{0}_{1\times3} \end{pmatrix}. \tag{23}$$

The design matrix $\mathbf{H}_{odometry,k}$ is then used in Equation 3 to compute the Kalman gain matrix. The measurement noise matrix $\mathbf{R}_{odometry,k}$ is modelled as a 4×4 diagonal matrix, where the first three elements contain the variances for speed ($\sigma_v^2 = (0.025[m/s])^2$) and the fourth element contains the variance for the heading ($\sigma_\psi^2 = (0.017[rad])^2$). The Kalman gain matrix and the reduced observations from Equation 22 are then used in Equation 5 to estimate the error state vector.

3) IMU as Reference Navigation Sensor: When the IMU is used as a reference navigation sensor to propagate the filter state, the filter state is propagated using a strapdown algorithm. For the formulation of the dynamic model and the linearised transition matrix, see [18].

Whenever an IMU is used for navigation, it is crucial to accurately detect the phases where the vehicle is not moving. When wheel speeds are present from encoders, they can be used to determine whether the vehicle is standing still or moving. However, simply making sure that all wheel speeds are zero is not sufficient. When the robot brakes, there is a short period of time where the wheel speeds are zero but the robot does not immediately come to a complete stop. This braking motion can be seen in the IMU data, as an angular rate about the vehicle's cross-axis (see Figure 4). Therefore, both the wheel speeds and the angular rate about the vehicle's cross-axis are combined and checked whether they are below a threshold. If they are lower than a threshold, a static phase is detected.

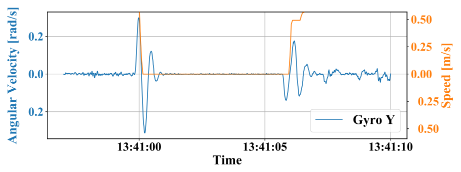

Fig. 4. Angular velocity about the vehicle's cross-axis measured by the IMU's gyro (blue) and mean wheel speed (orange) during a braking phase.

During static phases, the gyro bias is estimated. For online bias estimation, a circular buffer is filled with the IMU's gyroscope measurements in static phases. As a MEMS IMU is used in our study, we neglect the Earth's rotation rate and compute the gyro bias as a simple mean of the values in the circular buffer.

Note that no online accelerometer bias was estimated, but values from a previous calibration were used. Since the accelerometer bias is correlated with the velocity, the low resolution of the velocity measured by the rotary encoders leads to an erroneous estimation of the accelerometer biases and would deteriorate the overall result.

C. Software Development

The software development was done in ROS (Robot Operating System). This allows for a flexible architecture in real-time scenarios as well as in post-processing. As it is a node-based framework, it allows for easy adaption of filter parameters and quick iteration of software versions.

D. Field tests

The field tests for this study were conducted in July 2021 at Zentrum am Berg in Eisenerz, Austria. During these tests, the robot equipped with the before mentioned sensors was steered via remote control into the tunnel. A ground trouth was obtained by tracking two Leica 360°-prisms that were mounted on the robot with two robotic total stations. The mounting points of the prisms can be seen in Figure 5.

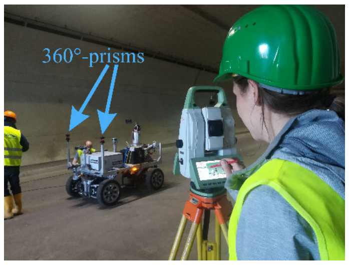

Fig. 5. Picture of how reference trajectory was obtained. Photo by Disaster Competence Network Austria (DCNA).

Figure 6 shows the trajectory of the robot during the tests in blue. The start point of the trajectory is shown in red and the walls of the tunnel are shown in black. The robot drove curved path along the tunnel axis. The two total stations were positioned at the tunnel entrance, close to the starting point. After the robot drove approximately 140 metres into the tunnel, the total stations lost the line-of-sight to the prisms.

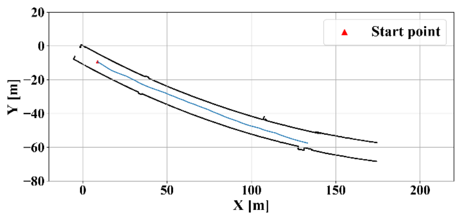

Fig. 6. Trajectory of the robot during the field tests.

Figure 7 shows a time series of the average wheel speeds obtained from the encoders. In the beginning, the speed is negative, which means that the robot drove backwards for a short time. Then, it drove forward with about 0.6 m/s and stopped after 160 seconds before it accelerated to 0.8 m/s. In total, it took the robot 240 seconds to travel 140 metres into the tunnel.

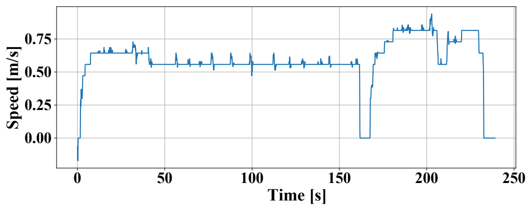

Fig. 7. Average of the four wheel speeds obtained from the rotary encoders during the tests.

During the tests, the point cloud data from the LIDAR was recorded with 10 Hz, wheel speeds and steering angles from odometry were recorded with 60 Hz and accelerations and angular rates from the IMU were recorded with 100 Hz.

For these first tests, the filter was initialised with the first ground truth position recorded by the total stations. In the future, it is planned that the robot is also equipped with a GNSS receiver. It will then use the last valid GNSS position before going into the tunnel as a filter estimate.

III. Results

The following section presents the results of the field tests. It will compare the GICP stand-alone solution and the proposed EKF solution that supports GICP with IMU and wheel odometry. To analyse the accuracy of both methods, the estimated position as well as the heading is compared to the ground truth.

Figure 8 shows the position error of GICP as a function of the distance travelled into the tunnel. The 2D or horizontal position error is depicted in blue, the 3D position error is depicted in orange. It is clearly visible that the position error grows with the distance traveled. The uniform structure of the tunnel with few distinguishable features makes it difficult to register the point clouds. The 2D error grows to 7.1 metres after 140 metres of going into the tunnel, the 3D error is higher than 10 metres.

Figure 9 shows the position error of the EKF that fuses GICP, wheel odometry, and IMU data. Again, the 2D error ist depicted in blue, while the 3D error is shown in orange. As with the GICP standalone solution, the error grows with traveled distance. This is to be expected, as both inertial

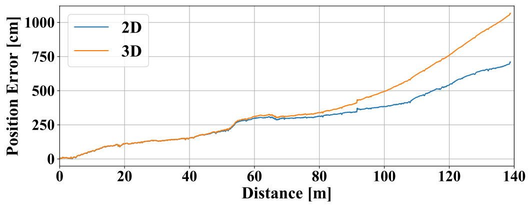

Fig. 8. Position and heading error of GICP stand-alone solution. The x-axis reflects distance traveled in metres.

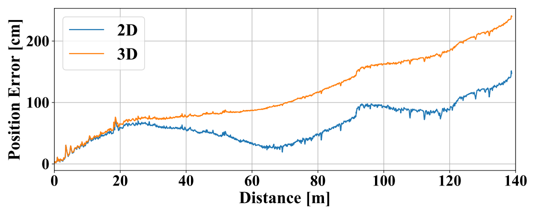

Fig. 9. Position and heading error of the EKF solution. The x-axis reflects distance traveled in metres.

TABLE II
Mean and standard deviation for filter results

	GICP	*Odometry & IMU*	*EKF*
Position Error 2D $\mu \pm \sigma$ *[m]*	3.0 ± 1.9	6.7 ± 6.2	0.6 ± 0.3
Position Error 3D $\mu \pm \sigma$ *[m]*	3.7 ± 2.8	6.7 ± 6.3	1.1 ± 0.6
Heading Error $\mu \pm \sigma$ *[°]*	-0.3 ± 1.1	-8.8 ± 5.3	-0.4 ± 1.2

navigation with an IMU and wheel odometry are methods of dead reckoning. All methods of dead reckoning only yield position changes and therefore compute the new position by adding the position change to the previous position. Errors therefore accumulate and the position error grows with time or distance traveled. However, when comparing the filter results to the GICP standalone solution in Figure 8, it can be seen that adding wheel odometry and IMU data significantly improves the positioning error.

Table I shows the maximum errors of the GICP solution, a solution using only odometry and the IMU, and the EKF fusing all sensors. As the error grows with distance in all solutions, the maximum position error corresponds to the error that occurs at the end of the trajectory. While the maximum 2D positioning error is 7.1 metres for the GICP standalone solution, and 21.3 metres for the solution using only odometry and the IMU, the EKF reduces the error to only 1.5 metres. This is because the GICP solution mainly drifts in the along-track component, while the combined odometry & IMU solution drifts in the cross-track component. Through a closed-loop EKF, both the along- and cross-track errors are minimized. The maximum 3D error of 10.7 metres for GICP standalone, or 21.4 metres for the combined odometry and IMU solution, is reduced to 2.4 metres by the EKF. The maximum heading error is biggest for the solution using only odometry and the IMU. Adding LIDAR data therefore improves the heading estimation.

TABLE I
Maximum error for filter results

	GICP	*Odometry & IMU*	*EKF*
Max. Position Error 2D [m]	7.1	21.3	1.5
Max. Position Error 3D [m]	10.7	21.4	2.4
Max. Heading Error [°]	7.6	19.1	8.1

Table II shows the mean position and heading errors as well as their standard deviations for the GICP standalone solution, the combined odometry and IMU solution, and the EKF fusing information from all available sensors. It can be seen that combining all sensors with an EKF also leads to a lower mean error and a lower standard deviation of the error.

IV. Discussion

The aim of this paper was to develop an algorithm to estimate position and attitude of a wheeled robot that can be used in emergency situations such as tunnel fires. This environment is particularly challenging for navigation as we cannot rely on GNSS, nor on any other infrastructure-based positioning methods.

The navigation sensors used in this study were a laser scanner by Velodyne (Puck VLP-16), an IMU by XSens (MTi-G-710), as well as the rotary encoders and linear actuators of the robot's wheels.

An Extended Kalman Filter is used to fuse information from all sensors and estimate the position and attitude of the robot. Wheel odometry and the laser scanner are used in the measurement update of the filter, and the IMU is used as a reference navigation sensor to propagate the filter state in the time update. A cascaded filter architecture is used, which means that the measurements of wheel odometry and the laser scanner are pre-processed before they are inserted into the filter. For wheel odometry, a 4WIS4WID-model is used to compute velocity and heading change; the point clouds of the laser scanner are registered using GICP.

The developed algorithm was tested at Zentrum am Berg, a tunnel research facility in Eisenerz, Austria. Two total stations were used to track the robot and obtain a ground truth. To analyse the achievable accuracy for position and heading, the results of the filter were compared to the ground truth.

They show that when only GICP with the point cloud recorded by the laser scanner is used to estimate the pose of the robot, the error accumulates quickly. After 140 metres of going into the tunnel, the 3D positioning error is bigger than 10 metres and the horizontal positioning error is 7.1 metres.

When an IMU and odometry information are used in an EKF to support the GICP estimate, the position error is reduced significantly. The 3D positioning error after 140 metres in the tunnel is reduced to 2.4 metres and the horizontal positioning error is reduced to 1.5 metres, which corresponds to an error of only 1.07% of the traveled distance.

In the future, the enhanced position estimate of the EKF will be fed back to the GICP as an initial guess to speed up registration and create a more accurate site map. Furthermore, the position estimate will be used to georeference detected hazardous substances.

This rapidly created site map using only onboard sensors of the robot and no external infrastructure should support emergency task forces in tunnel accidents. It should help to reduce uncertainties and enhance safety for emergency personnel.

Acknowledgment

This research was funded by the Austrian Federal Ministry for Agriculture, Regions and Tourism via the Österreichische Forschungsförderungsgesellschaft (Austrian Research Promotion Agency) (879693). The authors would like to thank the project partners at Montanuniversität Leoben - Subsurface Engineering, Disaster Competence Network Austria, Austrian Institute of Technology – Digital Safety and Security, JOANNEUM RESEARCH Forschungsgesellschaft mbH – Digital, Riegl Research Forschungsgesellschaft mbH, IQSoft GmbH (IQSOFT), CBRN Protection GmbH, E-NETIC, the Austrian Federal Ministry of Defence – Armed Forces, and the professional firebrigades of Graz, Linz, and Innsbruck.

References

[1] Ove Njå and Mona Svela. "A review of competencies in tunnel fire response seen from the first responders' perspectives". In: *Fire Safety Journal* 97 (2018), pp. 137–145. ISSN: 03797112. DOI: 10.1016/j.firesaf.2017.05.005.

[2] Timothy Chung. *DARPA Subterranean (SubT) Challenge*. URL: https://www.darpa.mil/program/darpa-subterranean-challenge (visited on 02/22/2022).

[3] Paul D. Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. second edition. Artech House GNSS library. Boston and London: Artech House, 2013. ISBN: 1608070050.

[4] Alberico Sonnessa et al. "Indoor Positioning Methods – A Short Review and First Tests Using a Robotic Platform for Tunnel Monitoring". In: *Computational Science and Its Applications – ICCSA 2020*. Ed. by Osvaldo Gervasi et al. Cham: Springer International Publishing, 2020, pp. 664–679. ISBN: 978-3-030-58811-3.

[5] Shibo Zhao et al. "Super Odometry: IMU-centric LiDAR-Visual-Inertial Estimator for Challenging Environments". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 8729–8736. DOI: 10.1109/IROS51168.2021.9635862.

[6] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time." In: *Robotics: Science and Systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9.

[7] Tixiao Shan et al. "LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 5135–5142.

[8] Tong Qin, Peiliang Li, and Shaojie Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020. DOI: 10.1109/TRO.2018.2853729.

[9] Konstantinos Loupos et al. "Autonomous robotic system for tunnel structural inspection and assessment". In: *International Journal of Intelligent Robotics and Applications* 2.1 (2018), pp. 43–66. ISSN: 2366-5971. DOI: 10.1007/s41315-017-0031-9.

[10] Elisabeth Menendez et al. "Tunnel structural inspection and assessment using an autonomous robotic system". In: *Automation in Construction* 87 (2018), pp. 117–126. ISSN: 09265805. DOI: 10.1016/j.autcon.2017.12.001.

[11] Richard Halatschek et al. "Universal Offroad Robot Platform for Disaster Response". In: *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics*. Vol. 2020 IEEE International Symposium on Safety, Security, and Rescue Robotics. 2020.

[12] Bernhard Hofmann-Wellenhof, Klaus Legat, and Manfred Wieser. *Navigation*. Vienna: Springer Vienna, 2003. ISBN: 978-3-211-00828-7. DOI: 10.1007/978-3-7091-6078-7.

[13] P. J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: 10.1109/34.121791.

[14] Sam Marden and Jose Guivant. "Improving the Performance of ICP for Real-Time Applications using an Approximate Nearest Neighbour Search". In: *Australasian Conference on Robotics and Automation* (2012).

[15] Kok-Lim Low. "Linear least-squares optimization for point-to-plane icp surface registration". In: *Chapel Hill, University of North Carolina* 4.10 (2004), pp. 1–3.

[16] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. "Generalized-ICP". In: *Robotics: Science and Systems V*. Robotics: Science and Systems Foundation, 2009. ISBN: 9780262514637. DOI: 10.15607/RSS.2009.V.021.

[17] Ming–Han Lee and Tzuu–Hseng S. Li. "Kinematics, dynamics and control design of 4WIS4WID mobile robots". In: *The Journal of Engineering* 2015.1 (2015), pp. 6–16. ISSN: 2051-3305. DOI: 10.1049/joe.2014.0241.

[18] Aboelmagd Noureldin, Tashfeen B. Karamat, and Jacques Georgy. *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN: 978-3-642-30465-1. DOI: 10.1007/978-3-642-30466-8.