SmartTrolley: An Experimental Mobile Platform for Indoor Localization in Warehouses

Abdelhak Bougouffa, Emmanuel Seignez, Samir Bouaziz, Florian Gardes

SmartTrolley: An Experimental Mobile Platform for Indoor Localization in Warehouses

Abdelhak Bougouffa*†, Emmanuel Seignez*, Samir Bouaziz* and Florian Gardes†
*Université Paris-Saclay, ENS Paris-Saclay, CNRS, SATIE
91190 Gif-sur-Yvette, France. [firstname.lastname]@universite-paris-saclay.fr
† ez-Wheel SAS., 16400 La Couronne, France. {a.bougouffa | f.gardes}@ez-wheel.com

***Abstract*—Warehouses and industrial sites are getting more and more interest in automating their workflow; in such an environment, a robust localization method is required to accomplish safe navigation indoors. One widely used scheme is the usage of custom AGVs and dedicated infrastructures to automate moving goods within the warehouse; however, such a solution needs to modify the infrastructure or to make custom robots that fit the existing infrastructure, which requires an important investment. In this paper, we present and validate the *SmartTrolley*, a generic, modular, and scalable experimental platform for usage in warehouses and industrial sites; able to localize itself in the environment using a scan matching and EKF based indoor Simultaneous Localization and Mapping (SLAM) algorithm.**

***Index Terms*—Mobile robot; Indoor Localization; LiDAR; Sensor Fusion; SLAM; NDT; PSO; EKF; SmartTrolley;**

I. Introduction

A mobile robot needs to interact with its environment; it needs a set of actuators to act on its environment and needs sensors to perceive the environment and make decisions depending on the sensed information. In order to achieve a given task, a mobile robot must locate itself in its environment; given this knowledge, the robot can plan, navigate, and interact with the real world.

Indoor localization has been subject to several research works since there is no standardized technology for that purpose; researchers use existing technologies (like Vision, laser ranging, WiFi, and Bluetooth) to provide localization methods that address specific use cases, and responds to specific accuracy requirements.

In mobile localization, the robot estimates its state (*i.e., the position and orientation in the map reference frame*) using known and available information about the environment (*i.e. a map with an attached reference frame*).

In some cases, having a predefined map of the environment can be challenging; in such a case, we often do both localization and map building at the same time, this is known as *Simultaneous Localization And Mapping* (SLAM). Laser scanners are very famous sensor types for SLAM; several methods have been proposed, based on scan matching [1]–[3], or filtering [4]. *Structure From Motion* (SFM) based SLAM is proposed in [5]; using a monocular camera, the algorithm extracts features using the *Shi-Tomasi* operator, then uses a *Particle Filter* to estimate depth data from a stream of 2D images, and an *Extended Kalman Filter* (EKF) to estimate the camera motion. In [6] the authors used a fusion of vision with publicly available digital map of the vehicle outdoors environment. Other SLAM methods based on mono, stereo, and depth cameras have been proposed in [7]–[9].

In other cases, when the environment can be modified, we can see the localization systems from an infrastructure viewpoint; thus, these systems can fit into three big categories, infrastructure-based, infrastructure-less, and hybrid localization systems. In infrastructure-based localization, the robot and the environment collaborate to perform the positioning, for this to works; the environment must be equipped with sensors, tags, markers or landmarks to facilitate the state estimation. Researchers used many technologies to achieve that, including the usage of *Visible-Light Communication* (VLC) [10]–[12], or exploiting wireless signal properties in radio-frequency based technologies like using the WiFi's *Received Signal Strength Indicator* (RSSI) to estimate the robot position with respect to known WiFi hotspots [13]–[16], or using Zigbee [17]–[19] and *Bluetooth* [20], [21] in a similar way, or using *Ultrawide Band* (UWB) and measure the *Difference Time of Arrival* (DToA) of the signal, then use this information to estimate distances to known UWB anchors [22]–[26]. Other researchers used *dead-reckoning* techniques to estimate the robot pose by integrating displacements in small time intervals; some studies used *Inertial Measurement Units* (IMUs) [27]–[29], others used *Optical Flow Sensors* widely known for being used in computer mice [30]–[33]. However, if used alone, dead-reckoning is known to be error cumulative; therefore, researchers generally use this technique along with other type of sensors like cameras, LiDARs, or radio-frequency sensors.

In [34], Choi proposed a hybrid RFID-Ultrasonic localization system, in this work, the localization is estimated globally using the RFID tags placed on the floor, this estimation gives an absolute position, but with high uncertainties, the algorithm uses ultrasonic sensors to build a local map which is used to reduce the uncertainty and improve accuracy. Other hybrid methods have been proposed in [35], [36].

In this paper, we present *SmartTrolley*; an experimental platform for moving heavy loads in warehouses. This platform, intended to validate multisensor fusion-based indoor localization algorithms, is presented with a LiDAR-based scan matching SLAM implementation with EKF sensor fusion.

We organized this manuscript as follow: Section II describes the *SmartTrolley* mobile platform, the used sensors, actuators, and equipment; Section III provides the kinematic model, the odometry model, and the laser data modeling; Section IV

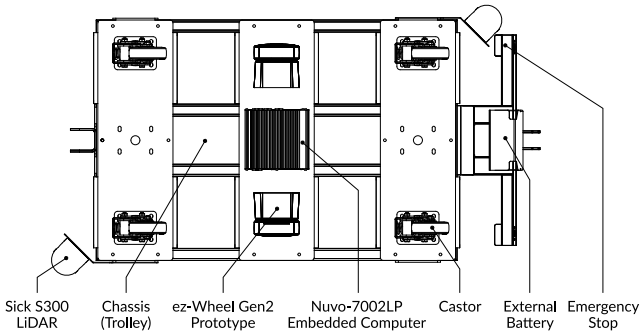

Fig. 1. Bottom view of the main components of our mobile trolley.

describes the model we used to represent the environment, the scan alignment, the pose estimation, and the EKF data fusion; In Section V an experimental validation is presented, and in Section VI we provide a conclusion with an overview of our future perspectives.

II. Experimental Platform

In this section, we present the *SmartTrolley*, a mobile platform for moving heavy loads in warehouses or industrial factories; this experimental platform is used to validate the localization and navigation algorithms.

For the chassis, we used a standard high load trolley of dimensions $1.2m \times 0.8m$. The propulsion is ensured by two self-contained traction wheels named *Gen2* from the *ez-Wheel* company, we mounted two traction wheels in the center of our trolley (as a differential-drive), and one castor at each corner, a bottom view of the platform is illustrated in (Fig. 1). We equipped the platform with two Sick S300 LiDARs and wheels odometers. However, our embedded system includes a large scale of communication technologies to easily add other sensors and implements a sensor fusion framework to fuse these sensors' data for better decision-making in highly dynamic environments.

A. Propulsion motors & wheels

For our platform, we used two wheels of the *ez-Wheel's Gen2*[1] model (Fig. 2); the *Gen2* generates a torque capable of moving around $1500kg$ of charges.

The *ez-Wheel* company develops *plug-and-play electric wheels*, that can be used to make motorizing any mobile platform an easy job. Each wheel is entirely self-contained; the *Gen2* model contains a wheel of diameter $\oslash 150mm$, a motor to wheel gearing (with a reduction factor of $1:14$), a Brushless DC Motor, an incremental encoder of $420\ ticks/rev$, a $100Wh$ Lithium-ion battery with an integrated BMS (*Battery Management System*), an optional brake system, the power electronic circuitry, an *sMC (Safe Motor Control)*[2] module (which ensures the communications, speed regulation and system safety), and an optional *Variscite DART-MX6* Linux powered System-on-Module (SoM) for high-level operations.

All parts are held in a compact housing with access to a Human-Machine Interface (HMI) via standard communication protocols like Ethernet, WiFi, CAN, RS232, and USB.

The *Safe Motor Control* (sMC) module contains an embedded microcontroller that regulates the velocity, it takes the desired speed as an input, and it outputs the incremental ticks counter calculated internally from the integrated rotation encoder.

The *Gen2* is designed with safety in mind; it complies with a set of safety-oriented standards, including the NF EN 61800-5-2 (Adjustable Speed Electrical Power Drive Systems/Part 5-2: Safety requirements), NF EN 61508 (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems), and NF EN 13849 (Safety of Machinery) standards.

We connected the two *Gen2* wheels to the embedded computer via a CAN bus, the communication uses the CANopen protocol.

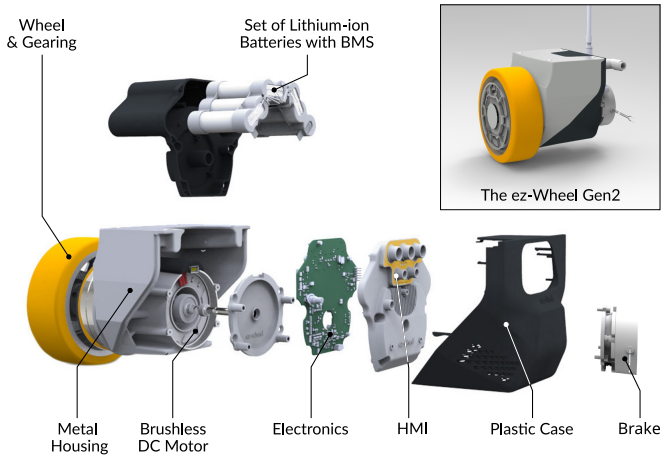

Fig. 2. Internal components of the *ez-Wheel Gen2* self-contained motorized wheel, top right: The assembled *Gen2* wheel.

B. Embedded Computer

We used the *Neousys Nuvo-7002LP* embedded computer (Fig. 3b), with an 8th Generation Intel® Coffee lake Core™ i5 processor, and a 16GB DDR4 2666/2400 SDRAM. By default, this embedded computer supports many IO protocols, including two software-programmable RS-232/422/485 ports that we used them to connect the two *S300* LiDARs via RS-422. To connect our two *Gen2* wheels, we added a CAN bus interface to the initial configuration. We used Ubuntu 18.04 LTS as an operating system and ROS Melodic as middleware.

C. Sensors

1) Proprioceptive sensors: Internal, proprioceptive, or interoception sensors gives us the internal state of the robot, without depending on external observations. In robotics, wheel encoders are the most famous proprioceptive sensors; they measure the rotation angle of the wheel in *ticks*, the encoder resolution is defined by the number of ticks it outputs per

[1]The *ez-Wheel Gen2* prototype we used still in development. The specs, design, and size of the final product might slightly differs the one we used, see: `safetywheeldrive.com` for up-to-date information.

[2]The *sMC (Safe Motor Control)* module is developed by the *ez-Wheel company*, it is certified for usage in safety-critical systems.

(a) *S300 Standard* (b) *Nuvo-7002LP*

Fig. 3. Used equipments, (a) the *S300 Standard* LiDAR, (b) the *Nuvo-7002LP* embedded computer.

one wheel revolution ($tick/rev$). Typically wheel encoders are divided into two types; incremental encoders used generally for speed regulation and absolute encoders used for position regulation.

The motorized wheels we used contains internal incremental wheel encoders; those are used internally to regulate the wheel rotation speed, we used this information to calculate the linear displacements of each wheel, then we used them to calculate the odometry of the robot.

2) Exteroceptive sensors: To perceive the environment, we use two SICK S300 LiDARs (Fig. 3a); each one covers a field of view (FOV) of 270°; we mounted these two LiDARs diagonally to cover 360° (Fig. 1).

The S300 is very popular in industrial applications; it uses the principle of time-of-flight to measure ranges; this model is intended to be used in monitoring hazardous areas indoors [37]. The S300 fulfills many safety-oriented directives and standards, like EN ISO 12100 (Safety of machinery), ISO 11161 (Industrial automation systems), EN ISO 10218-1 (Safety requirements for robots), and ANSI/RIA R15.06 (Safety requirements for Industrial Robots and Robot Systems), a more exhaustive list can be found in [38].

To improve system safety, the S300 can be configured with the *SICK Configuration & Diagnostics Software (CDS)* to define critical space around the robot. In operating time, if an object enters the critical space, the LiDAR sends a signal on its *safety outputs*. These outputs can be linked to brakes or to the embedded computer to handle the safety-critical situation; we used this feature to enhance the safety in dynamic environments at the lowest level.

The *S300 Standard* has a maximum measuring range of $30m$, FOV of 270° with an angular resolution of 0.5°, and offers safety-oriented configurable features, with a protective field range of $3m$ [37].

III. System Modeling

A. Kinematic Model

We model the robot as a differential-drive, the only parameters we can directly control in this case are the left and the right wheels velocities (v_l and v_r respectively). Let $\begin{pmatrix} x & y & \phi \end{pmatrix}^T$ denotes the robot state (the position and orientation in space), L represents the distance between the two traction wheels, and R represents the wheel radius, the robot kinematic model is given by:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} \frac{R}{2}(v_r + v_l)\cos\phi \\ \frac{R}{2}(v_r + v_l)\sin\phi \\ \frac{R}{L}(v_r - v_l) \end{pmatrix} \tag{1}$$

To control our robot, we need a way to specify the linear and angular velocities (v and ω respectively) as inputs; to do this, we can use the Unicycle model:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = \begin{pmatrix} v\cos\phi \\ v\sin\phi \\ \omega \end{pmatrix} \tag{2}$$

From (1) and (2) we can express the output parameters v_l and v_r as a function of input control parameters v and ω:

$$\begin{pmatrix} v_l \\ v_r \end{pmatrix} = \frac{1}{2R}\begin{pmatrix} 2v - \omega L \\ 2v + \omega L \end{pmatrix} \tag{3}$$

B. Odometry

We need to calculate the position by summing traveled distances at small amounts of time, the only information we can get from the encoders is an incremental counter of *ticks*. Let $\Delta tick = (tick_t - tick_{t-1})$ the difference of ticks at a given time iteration t, knowing the wheel radius R and the encoder resolution res, the distance traveled by the robot center $\Delta\mathfrak{D}$ can be calculated using the distances traveled by the left and right wheels (ΔD_l and ΔD_r respectively):

$$\Delta\mathfrak{D} = \frac{\Delta D_l + \Delta D_r}{2} \text{ with } \Delta D_{l|r} = \frac{2\pi R}{res}\Delta tick_{l|r} \tag{4}$$

At time t, the robot odometry $\mathbf{x}_t = \begin{pmatrix} x_t & y_t & \phi_t \end{pmatrix}^T$ can be determined by accumulating the partial odometries $\Delta\mathbf{x}$ which can be calculated from $\Delta\mathfrak{D}$, ΔD_r, and ΔD_l of (4):

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}) = \mathbf{x}_{t-1} + \Delta\mathbf{x} = \begin{pmatrix} x_{t-1} + \Delta\mathfrak{D}\cos\phi_t \\ y_{t-1} + \Delta\mathfrak{D}\sin\phi_t \\ \phi_{t-1} + \frac{\Delta D_r - \Delta D_l}{L} \end{pmatrix} \tag{5}$$

In a differential-drive system, the traction wheels need to be driven with the same velocity profile; this can be challenging due to the variations between the wheels, motors, and floor conditions [39]. Knowing these constraints and adding the wheel slippage problems, the estimated state using only odometers will quickly drift from the real state.

C. Laser scanner

The LiDAR provides a list of ranges (*distances*), which contains 541 elements; each element represents the measured distance at an angle starting at −135° to 135°, with a 0.5° increment. Supposing U is the list of distances indexed with n, we can pass from this ranges list to a list of 2D points in polar coordinates, the point corresponding to the n-th element is u_n such as:

$$u_n = \begin{pmatrix} \rho_n \\ \theta_n \end{pmatrix} = \begin{pmatrix} U_n \\ -135^\circ + n \times 0.5^\circ \end{pmatrix} \text{ with } 0 \leq n < 541 \tag{6}$$

For our algorithm, we need to convert the points from polar (u_n) to cartesian coordinates (p_n) using:

$$p_n = \begin{pmatrix} x_n \\ y_n \end{pmatrix} = \begin{pmatrix} \rho_n \cos\theta_n \\ \rho_n \sin\theta_n \end{pmatrix} \text{ with } 0 \leq n < 541 \tag{7}$$

We represent each scan frame as a collection of 541 2D points.

IV. The SLAM Approach

We equipped the experimental platform with two wheels encoders, and two LiDARs covering 360°, having these sensors, we used an optimization-based, scan matching algorithm named NDT-PSO [40] to align scans from each LiDAR separately. This method is based on the *Normal Distribution Transform* (NDT) scan matching method [3] and the bio-inspired *Particle Swarm Optimization* (PSO) [41].

The proposed system contains an odometer node; two separates NDT-PSO nodes (for each LiDAR) running alongside a fusion node that performs the EKF fusion using the odometry to predict and the NDT-PSO to update.

A. Environment Modeling

We used a *Normal Distribution Transform* [3] based representation to model the environment.

The first step in this representation is to divide the environment to a set of cells of known dimensions ($1m \times 1m$ for example), then the idea is to represent the 2D points contained in each cell "c" as a normal distribution $\mathcal{N}_c(\mu_c, \Sigma_c)$ instead of dealing with them as a point cloud. To achieve this, for a cell "c" we collect all "n" 2D points $p_{i=1\ldots n} = (x_i\ y_i)^T$, and calculate the normal distribution $\mathcal{N}_c(\mu_c, \Sigma_c)$ parameters:

$$\mu_c = \frac{1}{n}\sum_{i=1}^{n} p_i \quad \text{and} \quad \Sigma_c = \frac{1}{n}\sum_{i=1}^{n}(p_i - \mu_c)(p_i - \mu_c)^T \tag{8}$$

Note that for 2D points, the mean $\mu_c \in \mathbb{R}^2$ and the covariance matrix $\Sigma_c \in \mathbb{R}^2 \times \mathbb{R}^2$.

The set of normal distributions ($\mathcal{N}_c$) represents the NDT map; this representation is similar to an occupancy grid [42]. However in NDT, the probability is not set to the whole cell; instead, it associates a probability density function (PDF) to each cell (Fig. 4), the probability of measuring a 2D point sample $\mathbf{p}$ in a cell c is given by the PDF "Π_c" such as:

$$\Pi_c(\mathbf{p}) = \mathcal{C} \exp -\frac{(\mathbf{p} - \mu_c)^T \Sigma_c^{-1} (\mathbf{p} - \mu_c)}{2} \tag{9}$$

In a k-dimensions multivariate normal distribution, the $\mathcal{C}$ constant is usually set to $((2\pi)^{-\frac{k}{2}} \det(\Sigma)^{-\frac{1}{2}})$, however, in the optimization process, we do not need to normalize the probabilities; therefore, $\mathcal{C}$ is set to 1 in the PDF (9).

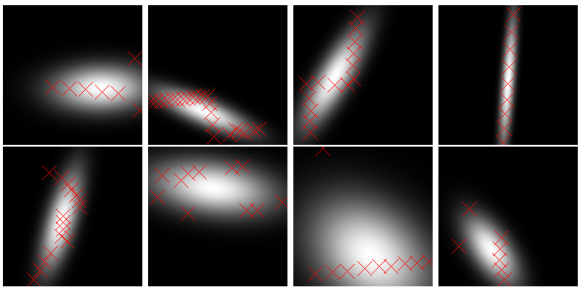

Fig. 4. The NDT representation of some cells, the input points represented as red cross marks (from [3]).

B. Scans Alignment and Pose Estimation

Let $\begin{pmatrix}\Delta x & \Delta y\end{pmatrix}^T$ the translation between two laser scans and $\Delta\phi$ is the orientation change. The spatial mapping $\mathcal{T}(\mathbf{x}_{t-1}, \Delta\mathbf{x}_t)$ between two robot coordinate frames from time $t-1$ to t is:

$$\mathcal{T} : \begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} \cos\phi_t & -\sin\phi_t \\ \sin\phi_t & \cos\phi_t \end{pmatrix} \begin{pmatrix} x_{t-1} \\ y_{t-1} \end{pmatrix} + \begin{pmatrix} \Delta x_t \\ \Delta y_t \end{pmatrix} \tag{10}$$

$$\text{with} \quad \begin{cases} \phi_t = \phi_{t-1} + \Delta\phi_t \\ \mathbf{x}_{t-1} = (x_{t-1}\ y_{t-1}\ \phi_{t-1})^T \text{ and } \Delta\mathbf{x}_t = (\Delta x_t\ \Delta y_t\ \Delta\phi_t)^T \end{cases}$$

Given two scans at times t and $t-1$, the goal of *scan alignment/registration* is to find the transformation parameters Δx_t, Δy_t and $\Delta\phi_t$ between the two scans.

For a scan of N 2D points "$\mathbf{p}_{i=1\ldots N}$", the objective function $\mathbf{\Lambda}$ of the parameter $\Delta\mathbf{x} = (\Delta x\ \Delta y\ \Delta\phi)^T$ is defined as in [3]:

$$\mathbf{\Lambda}(\Delta\mathbf{x}) = -\sum_{i=1}^{N} \Pi_c(\mathcal{T}(\mathbf{p_i}, \Delta\mathbf{x})) \tag{11}$$

In [3], the $\Delta\mathbf{x}$ is optimized using $\mathbf{\Lambda}$ with the Newton method, but in [40] a PSO based optimization is used for better global convergence. PSO starts by randomly initializing a swarm of M particles $P^{i=1\ldots M} = (\Delta x_i, \Delta y_i, \Delta\phi_i)$ in a chosen search space, in our case, the particle is a 3D vector, hence $j = 1\ldots 3$. Using the objective function for each particle P^i, we calculate the particle velocity and position [41] as follow (we removed the i index for simplicity):

$$v_{t+1}^j = w v_t^j + c_1 r_1^j (B_t^j - P_t^j) + c_2 r_2^j (G_t^j - P_t^j) \tag{12}$$

$$x_{t+1}^j = x_t^j + v_{t+1}^j \quad \text{with} \quad j = 1\ldots 3 \tag{13}$$

With w, c_1 and, c_2 are the PSO inertial, cognitive, and social constants; r_1 and r_2 are two positive random numbers in range $[0, 1]$. The B_t is the particle's personal best, and G_t is the global best of the whole swarm. At time $t+1$ and for a particle of index "i", the personal and global bests are chosen as follow:

$$B_{t+1}^i = \begin{cases} P_{t+1}^i & \text{if } \mathbf{\Lambda}(P_{t+1}^i) < \mathbf{\Lambda}(B_t^i) \\ B_t^i & \text{else} \end{cases} \tag{14}$$

$$G_{t+1} = \begin{cases} B_{t+1}^i & \text{if } \mathbf{\Lambda}(B_{t+1}^i) < \mathbf{\Lambda}(G_t) \\ G_t & \text{else} \end{cases} \tag{15}$$

Having all these pieces, the NDT-PSO scan matching SLAM executes as follow:

1) At time t, divide scan $t-1$ into cells (of $1m \times 1m$), and calculate the mean and the covariance for each cell as presented in (8),
2) Convert laser ranges U_k^t to a point cloud p_k^t using (6) and (7).
3) Randomly initialize the particle swarm in a chosen search space.
4) For each particle P^i:
   a) Calculate the particle's velocity and position using (12) and (13).
   b) Map the samples p_k from the coordinate frame of scan t to the scan $t-1$ using $\mathcal{T}(p_k, P^i)$ in (10),
   c) Use the mapped point to determine the personal and global bests (B_t^i, G_t) as in (14) and (15),
   d) Repeat from step (4) until convergence or for a fixed number of iterations.
5) The global best G_t is selected as the change in pose from time $t-1$ to t.

At the end of the iteration, the pose is updated using the global best as follow:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + G_t \tag{16}$$

Our implementation of NDT-PSO uses a temporal window for each cell, implemented as a circular buffer of points inside the cell; this technique limits the number of points used per cell, which makes the memory complexity depends only on the map size and reduces the computational complexity when calculating the normal distribution parameters. Limiting points is very important when operating for a long time; and also, since older points get replaced by new observations, this temporal window allows so far the filtering of moving objects without a proper detection and filtering scheme.

C. Extended Kalman Filter Fusion

Due to small structural imperfections, installing the two laser scanners in the same horizontal plane can be a tricky task, even a little vertical angle difference can cause more remarkable differences in the measured ranges. Without a precise calibration, the front and back LiDARs may see a slightly inclined planes with respect to each other, this can cause some inconsistencies when matching scans from the front LiDAR with those obtained from the back one, and cause higher uncertainties in matched scans.

Separately matching each LiDAR can assure a kind of local consistency between different scans of the same LiDAR. Then, fusing the obtained poses from each LiDAR might give us a better estimate of the pose.

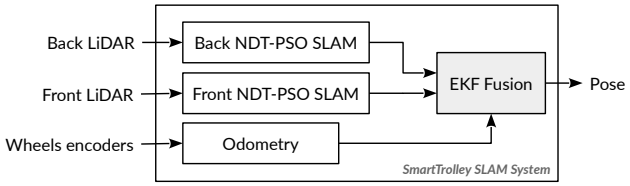

Fig. 5. The *SmartTrolley*'s fusion architecture.

Furthermore, since the two LiDARs are completely independents, using a distributed fusion architecture (Fig. 5) can deliver pose information even if one of the LiDARs fails, which is a required constraint in safety oriented navigation.

We can see that the expression of $\mathbf{x}_t = f(x_{t-1})$ in the odometry model (5) is not linear; thus, an Extended Kalman Filter (EKF) is used for the fusion.

In our preliminary tests, we tried both the EKF and a Particle Filter (PF), we choose an EKF because it is much faster than PF, and because the fusion is performed after NDT-PSO which is based on a stochastic swarm optimization that is similar to a Particle Filter.

1) Prediction Phase: For the prediction step, we used the odometry got from the proprioceptive system of wheel encoders, the prediction of the state $\hat{\mathbf{x}}_{k|k-1}$ and the associated uncertainty $\mathbf{P}_{k|k-1}$ at time k given all measurements up to $k-1$ are:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}) + q_{k-1} \tag{17}$$

$$\mathbf{P}_{k|k-1} = f'(\hat{\mathbf{x}}_{k-1|k-1})\mathbf{P}_{k-1|k-1}f'(\hat{\mathbf{x}}_{k-1|k-1})^T + \mathbf{Q}_{k-1} \tag{18}$$

With $q_{k-1} \sim \mathcal{N}(0, \mathbf{Q}_{k-1})$ is the gaussian process noise, f is the odometry expression defined in (5) and f' it's first Taylor expansion which can be expressed as:

$$f'(\mathbf{x}) = \frac{\partial f}{\partial \mathbf{x}} = \begin{pmatrix} 1 & 0 & -\Delta\mathfrak{D}\sin\phi \\ 0 & 1 & \Delta\mathfrak{D}\cos\phi \\ 0 & 0 & 1 \end{pmatrix} \tag{19}$$

2) Update Phase: In each time iteration, we run NDT-PSO on the front and back LiDARs; each one gives an observation (pose) in the same reference frame, the final step is to fuse these two observations using the predicted state, each time we receive a pose form NDT-PSO we run the prediction and the update with the received observation.

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - h(\hat{\mathbf{x}}_{k|k-1})) \tag{20}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^T \tag{21}$$

With $\mathbf{S}_k$ is the predicted covariance of the measurement $\mathbf{y}_k$, $\mathbf{K}_k$ is the Kalman gain, and $\mathbf{R}_k$ is the covariance of the measurement noise.

Fig. 6. The first prototype of our *SmartTrolley* experimental mobile platform, in the test room.

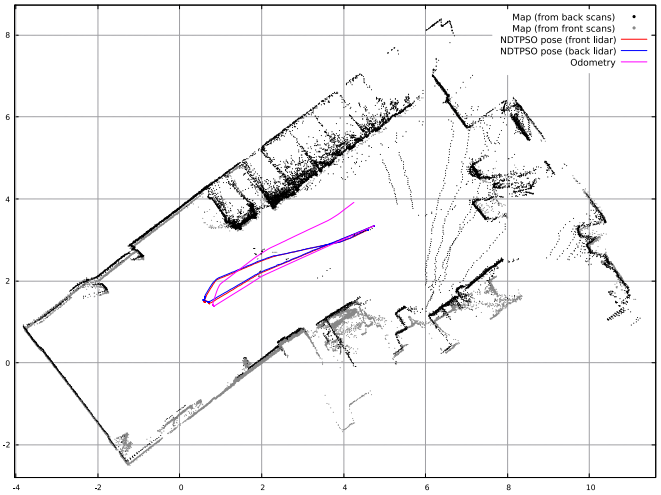

Fig. 7. The NDT-PSO generated maps from the two LiDARs, with associated poses, and the odometry (distance unit is *meters*).

$$\mathbf{S}_k = h'(\hat{\mathbf{x}}_{k|k-1})\mathbf{P}_{k|k-1}h'(\hat{\mathbf{x}}_{k|k-1})^T + \mathbf{R}_k$$
$$\mathbf{K}_k = \mathbf{P}_{k|k-1}h'(\hat{\mathbf{x}}_{k|k-1})^T\mathbf{S}_k^{-1}$$

Note that we do not need an observation model in our case since the poses are already calculated using NDT-PSO, so $h'(\mathbf{x}) = \mathbf{H} \cdot \mathbf{x}$ is used with $\mathbf{H} = I_3$.

V. Experimental Results

In this Section, the *SmartTrolley* platform (Fig. 6) is used to validate our SLAM algorithm, with two NDT-PSO ROS nodes, an odometer node, and the EKF node running in parallel. When running our NDT-PSO implementation for one LiDAR, the algorithm's frequency is around 40Hz while the LiDAR is configured to publish data at a frequency of 8Hz, however, due to the operating system scheduling, when running the two NDT-PSO nodes simultaneously, the overall matching frequency drops to around 9Hz, this can be improved by redesigning the implementation to run the two LiDARs as threads in the same ROS node instead of having two separate processes. The system provides a fused pose at a rate of 8Hz, which is acceptable for real-time execution in our case since the robot operates at low speeds.

We run our experience in a room of $11m$ length and $6.7m$ width in the wider region and $4.25m$ in the narrowest one. We used an Xbox wireless gamepad to move the robot remotely. The *SmartTrolley* started at the position marked with a red star in (Fig. 8), it traveled a distance of $4.5m$ before performing some clockwise and anticlockwise rotations on itself, and then returning in backward direction to nearby its starting position.

The (Fig. 7) shows the odometry trajectory and the two NDT-PSO trajectories generated from the front and back LiDARs with their associated maps. We can notice that at starting, the odometry was close to the pose estimated form the two LiDARs, but over time it gets diverged from the LiDARs' estimated poses, at the end of the experiment, the distance from the actual position of the robot and the odometry was $0.75m$. This effect is well known for the odometry since there is no external correction on it; wheel slippage causes each time a small error which gets cumulated over time, producing a poor estimate of the pose when only using the odometry. We can also see a small mismatching between the two maps; this is due to the structural imperfections discussed previously, this mismatching impacts also the poses associated with each map.

The proposed solution was to use the odometry to predict the state and fuse the two LiDARs poses with an EKF, the (Fig. 8) shows the trajectory after fusion, alongside the input poses and odometry.

Since NDT-PSO initialize the search space randomly at each iteration and uses a stochastic optimization method, the output pose is a little bit noisy; this effect can be seen in (Fig. 8). On the other hand, odometry seems to be much smoother due to its straightforward calculation formulas. Using the odometry in the EKF prediction gives as a smoother output compared to the LiDAR input poses. Prediction using odometry also gives the advantage of using real feedback of the system instead of a kinematic model-based prediction (predicting the robot's movement only from the control input), this is very useful in the case of failure of the kinematic model prediction, like when the motors cannot be regulated to the desired input speed (for example, due to a power issue, or a movement on an inclined floor) or when the robot gets pushed externally.

In (Fig. 8), we can observe that the fused trajectory is somewhere between the two LiDARs trajectories; this is the expected behavior since the EKF uses the same uncertainties for both front and back LiDARs; we can also observe that the fused pose is smoother than LiDAR poses.

The advantage of fusing two poses over merging the scans and performing one SLAM matching is that having two independents input data for the fusion; an outlier pose (due to alignment error in one of the LiDARs) can be corrected so far, the lousy pose will not contribute as much as the good one since it will lay too far from its expected uncertainty.

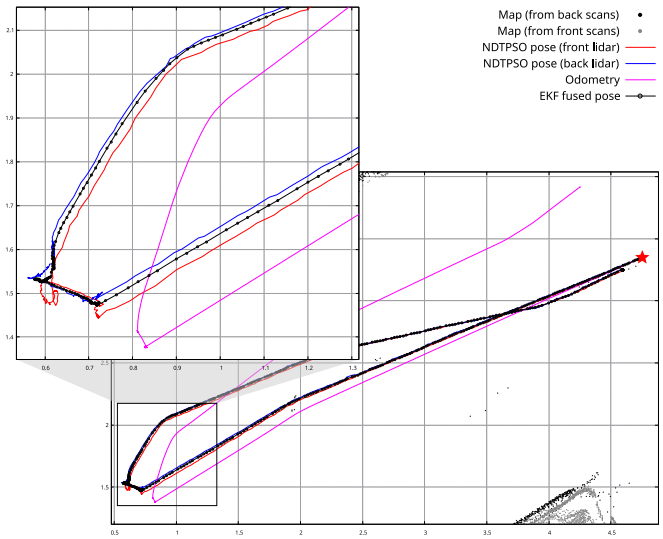

Fig. 8. The EKF fused trajectory, alongside with the input data from odometry and two LiDARs poses (distance unit is *meters*), the starting position is marked with a red star.

Also, merging the two scans needs to have an accurately synchronized LiDARs, when an EKF sensor fusion can deal with asynchronous data.

The table I shows the Mean Absolute Error (MAE), and the Mean Squared Error (MSE) for the trajectory of (Fig. 8), calculated between the fused pose and the odometry, the front LiDAR pose, and the back LiDAR pose. The table gives an idea about each input's contribution to the fused output pose.

TABLE I
MEAN ABSOLUTE ERROR (MAE) AND MEAN SQUARED ERROR (MSE), CALCULATED BETWEEN THE FUSED AND THE INPUT DATA

	MAE $(m\ m\ rad)$	MSE $(m^2\ m^2\ rad^2)$
Odometry	(0.1245 0.1572 0.1341)	(0.0267 0.0640 0.0382)
Front	(0.0161 0.0158 0.0100)	(0.0004 0.0007 0.0004)
Back	(0.0133 0.0103 0.0089)	(0.0003 0.0002 0.0004)

VI. CONCLUSION AND PERSPECTIVES

In this paper, we presented our initial prototype of the *SmartTrolley* experimental platform; we have also presented an EKF-based data fusion scheme for Simultaneous Localization and Mapping (SLAM) using laser scanners. An implementation experimental validation is also provided.

ROS implementations of our SLAM and EKF fusion systems are used to validate the proposed method; our system was able to respond to the real-time constraints of the *SmartTrolley* platform. The results show a smoother pose estimate compared to the noisy LiDARs poses inputs. The distributed fusion architecture we used gave us a robust way for pose estimation, which delivers reasonable estimates even when one of the LiDARs fails, or when an outlier pose is provided by one of the LiDARs as a result of scan matching error.

This prototype uses only LiDARs and wheels encoders. However, the project aims to add a set of low-cost sensors; some for solving the localization problem incrementally (like cameras and ultrasonic sensors); alongside with sensors that can be used to get the absolute localization in the robot's environment (like WiFi, Zigbee, Bluetooth and UWB); and replaces the LiDAR with these sensors. For safety purposes, after removing the LiDARs; a set of infrared ToF ranging sensors will be added around the robot for collision detection and avoidance of obstacles and moving objects.

ACKNOWLEDGMENT

This work has been conducted partially at the *ez-Wheel's Research & Development* as part of the CIFRE doctoral funding program, we would like to thank all our colleges in the company who helped us with this work.

REFERENCES

[1] G. Peng, W. Zheng, Z. Lu, J. Liao, L. Hu, G. Zhang, and D. He, "An Improved AMCL Algorithm Based on Laser Scanning Match in a Complex and Unstructured Environment," *Complexity*, vol. 2018, p. 2327637, Dec. 2018.

[2] K. Lingemann, A. Nüchter, J. Hertzberg, and H. Surmann, "High-speed laser localization for mobile robots," *Robotics and Autonomous Systems*, vol. 51, no. 4, pp. 275–296, Jun. 2005.

[3] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Intelligent Robots and Systems 2003 (IROS 2003)*, vol. 3. IEEE, 2003, pp. 2743–2748.

[4] H. Ye and C. C Zhou, "A new EKF SLAM algorithm of lidar-based AGV fused with bearing information," *TechConnect Briefs*, vol. 4, no. 2018, pp. 32–39, May 2018.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[6] D. N. S. D. Awang Salleh and E. Seignez, "Longitudinal error improvement by visual odometry trajectory trail and road segment matching," *IET Intelligent Transport Systems*, vol. 13, no. 2, pp. 313–322, 2019.

[7] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "PL-SLAM: Real-time monocular visual SLAM with points and lines," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4503–4508.

[8] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.

[9] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[10] E. Torres-Zapata, J. M. Luna-Rivera, R. Perez-Jimenez, V. Guerra, J. Rabadan, J. Rufo, and C. A. Gutierrez, "Implementation of a VLC-based indoor localization system," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 2, p. e3498, 2019.

[11] P. Cherntanomwong and W. Chantharasena, "Indoor localization system using visible light communication," in *2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE)*, Oct. 2015, pp. 480–483.

[12] W. Guan, S. Chen, S. Wen, W. Hou, Z. Tan, and R. Cen, "Indoor Localization System of ROS mobile robot based on Visible Light Communication," *arXiv:2001.01888 [eess]*, Jan. 2020.

[13] C. Chen, Y. Chen, H.-Q. Lai, Y. Han, and K. R. Liu, "High accuracy indoor localization: A WiFi-based approach," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2016, pp. 6245–6249.

[14] P. Bahl and V. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, Mar. 2000, pp. 775–784 vol.2.

[15] J. Biswas and M. Veloso, "WiFi localization and navigation for autonomous indoor mobile robots," in *International Conference on Robotics and Automation*, May 2010.

[16] A. Rai, K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *MobiCom*, 2012.

[17] J. Niu, B. Wang, L. Shu, T. Q. Duong, and Y. Chen, "ZIL: An Energy-Efficient Indoor Localization System Using ZigBee Radio to Detect WiFi Fingerprints," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 7, pp. 1431–1442, Jul. 2015.

[18] T. Alhmiedat, G. Samara, and A. O. A. Salem, "An Indoor Fingerprinting Localization Approach for ZigBee Wireless Sensor Networks," *arXiv:1308.1809 [cs]*, Aug. 2013.

[19] S.-H. Fang, C.-H. Wang, T.-Y. Huang, C.-H. Yang, and Y.-S. Chen, "An Enhanced ZigBee Indoor Positioning System With an Ensemble Approach," *IEEE Communications Letters*, vol. 16, no. 4, pp. 564–567, Apr. 2012.

[20] M. E. Rida, F. Liu, Y. Jadi, A. A. A. Algawhari, and A. Askourih, "Indoor Location Position Based on Bluetooth Signal Strength," in *2015 2nd International Conference on Information Science and Control Engineering*, Apr. 2015, pp. 769–773.

[21] M. Terán, H. Carrillo, and C. Parra, "WLAN-BLE Based Indoor Positioning System using Machine Learning Cloud Services," in *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, Nov. 2018, pp. 1–6.

[22] G. Cheng, "Accurate TOA-Based UWB Localization System in Coal Mine Based on WSN," *Physics Procedia*, vol. 24, pp. 534–540, Jan. 2012.

[23] S. Galler, W. Gerok, J. Schroeder, Kyandoghere Kyamakya, and T. Kaiser, "Combined AOA/TOA UWB localization," in *2007 International Symposium on Communications and Information Technologies*, Oct. 2007, pp. 1049–1053.

[24] A. De Angelis, J. Nilsson, I. Skog, H. Peter, and P. Carbone, "Indoor Positioning by Ultrawide Band Radio Aided Inertial Navigation," *Metrology and Measurement Systems*, no. No 3, pp. 447–460, 2010.

[25] E. García, P. Poudereux, Á. Hernández, J. Ureña, and D. Gualda, "A robust UWB indoor positioning system for highly complex environments," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2015, pp. 3386–3391.

[26] J. Tiemann and C. Wietfeld, "Scalable and precise multi-UAV indoor navigation using TDOA-based UWB localization," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Sep. 2017, pp. 1–7.

[27] H. Guo, M. Uradzinski, H. Yin, and M. Yu, "Indoor positioning based on foot-mounted IMU," *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 63, no. No 3, pp. 629–634, 2015.

[28] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust Visual Inertial Odometry Using a Direct EKF-Based Approach," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. ETH-Zürich, 2015.

[29] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU Dead-Reckoning," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[30] A. Bonarini, M. Matteucci, and M. Restelli, "Dead Reckoning for Mobile Robots Using Two Optical Mice," in *ICINCO 2004, Proceedings of the First International Conference on Informatics in Control, Automation and Robotics, Setúbal, Portugal, August 25-28, 2004*, H. Araújo, A. Vieira, J. Braz, B. Encarnação, and M. Carvalho, Eds. INSTICC Press, 2004, pp. 87–94.

[31] L. Mudrová, J. Faigl, J. Halgašík, and T. Krajník, "Estimation of Mobile Robot Pose from Optical Mouses," in *Research and Education in Robotics - EUROBOT 2010*, ser. Communications in Computer and Information Science. Berlin, Heidelberg: Springer, 2011, pp. 93–107.

[32] Y. Liu, Y. Ou, and W. Han, "Mobile Robot Localization Based on Optical Sensor," in *2019 IEEE International Conference on Real-Time Computing and Robotics (RCAR)*, Aug. 2019, pp. 874–879.

[33] D. Sekimori and F. Miyazaki, "Precise Dead-Reckoning for Mobile Robots using Multiple Optical Mouse Sensors," in *Informatics in Control, Automation and Robotics II*, J. Filipe, J.-L. Ferrier, J. A. Cetto, and M. Carvalho, Eds. Dordrecht: Springer Netherlands, 2007, pp. 145–151.

[34] B. Choi and J. Lee, "Mobile robot localization scheme based on RFID and sonar fusion system," in *2009 IEEE International Symposium on Industrial Electronics*, Jul. 2009, pp. 1035–1040.

[35] Y. Shi, W. Zhang, Z. Yao, M. Li, Z. Liang, Z. Cao, H. Zhang, and Q. Huang, "Design of a Hybrid Indoor Location System Based on Multi-Sensor Fusion for Robot Navigation," *Sensors (Basel, Switzerland)*, vol. 18, no. 10, Oct. 2018.

[36] Y. U. Lee and M. Kavehrad, "Two hybrid positioning system design techniques with lighting LEDs and ad-hoc wireless network," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 4, pp. 1176–1184, Nov. 2012.

[37] "Operating instructions S300 Safety laser scanner, SICK Sensors Intelligence," Feb. 2016.

[38] "Guidelines Safe Machinery: Six steps to a safe machine, SICK Sensors Intelligence," Jul. 2015.

[39] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*, second edition ed. The MIT Press, 2011.

[40] S. Bouraine, A. Bougouffa, and O. Azouaoui, "NDT-PSO, a New NDT based SLAM Approach using Particle Swarm Optimization," in *16th International Conference on Control, Automation, Robotics and Vision (ICARCV) (Accepted)*, 2020.

[41] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95*, vol. 4, Nov. 1995, pp. 1942–1948 vol.4.

[42] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *1985 IEEE International Conference on Robotics and Automation Proceedings*, vol. 2, Mar. 1985, pp. 116–121.