IMU and Cable Encoder Data Fusion for In-Pipe Mobile Robot Localization

Andreu Corominas Murtra, Josep M. Mirats Tur
CETaqua, Water Technological Center. Barcelona, Catalunya, Spain
www.cetaqua.com, acorominas/jmirats@cetaqua.com

***Abstract*—Inner pipe inspection of sewer networks is a hard and tedious task, due to the nature of the environment, which is narrow, dark, wet and dirty. So, mobile robots can play an important role to solve condition assessment of such huge civil infrastructures, resulting in a clear benefit for citizens. One of the fundamental tasks that a mobile robot should solve is localization, but in such environments GPS signal is completely denied, so alternative methods have to be developed. Visual odometry and visual SLAM are promising techniques to be applied in such environments, but they require a populated set of visual feature tracks, which is a requirement that can not be fulfilled in such environments in a continuous way. With the aim of designing robust and reliable robot systems, this paper proposes and evaluates a complementary approach to localize a mobile robot, which is based on sensor data fusion of an inertial measurement unit and of a cable encoder, which measures the length of an unfolded cable, from the starting point of operations up to the tethered robot. Data fusion is based on optimization of a set of windowed states given the sensor measurements in that window. The paper details theoretical basis, practical implementation issues and results obtained in testing pipe scenarios.**

I. Introduction

Inner pipe inspection of sewer networks is a hard and tedious task, due to the nature of the environment, which is narrow, dark, wet and dirty. Mobile robots can play an important role to solve condition assessment of such huge civil infrastructures, resulting in a clear benefit for citizens. For instance, only at the city of Barcelona, sewer network has an overall length of $5000\ Km$, an underground asset that has grown over the last centuries, built of different materials and sizes. Inspection robots have a big market to be exploited in such scenarios, but fundamental robot capabilities have to be robustly solved to definitively introduce them as reliable and market-ready solutions.

One of these fundamental tasks that a mobile robot should solve is localization, taking into account that in such environments GPS signal is completely denied. To overcome this limitation, alternative methods have to be developed. Visual odometry [12], [9], [5] and visual SLAM [2], [7], [11] are promising techniques to be applied in such environments. Adding an Inertial Measurement Unit (IMU) to the sensor set-up provides extra robustness, while it overcomes scale observability lack of monocular vision approaches [1], [10], [8]. However, such vision based methods require a populated set of visual feature tracks to robustly estimate robot positions. This requirement is not allways fulfilled in inner pipe scenarios, due to areas of poor pipe surface texture, to sudden water flows that difficults image processing steps, or to high illumination reflections produced by on-board light source.

This paper proposes and evaluates a complementary approach to localize a mobile robot, which is based on data fusion of two sensor devices: an IMU and a cable encoder, the later measuring the length of an unfolded cable, from the starting point of operations up to the robot platform, which is tethered for safety, communications and energy reasons. This sensor set-up has the great advantage of its availability, since it is independent of environment situations which usually cause failures in image processing. The presented method is based on the Graph SLAM approach [13], [1], [4], so it can be easily exported as a complementary method to add robustness to an inertial/vision based Graph SLAM estimation. Optimization is carried out within an interval of consecutive platform states, called a window state, given the sensor measurements in that interval. Optimization tries to find out the most likely window state, given sensor measurements, where likelihood is treated in a probabilistic way. In terms of a cost function, such optimization searches the window state point that minimizes a cost function, which is expressed as a sum of several constraints, imposed by measurements. The paper details theoretical basis, practical implementation issues and results obtained in testing pipe scenarios with a wheeled platform.

II. State Definitions and Graph Model

We consider the following *full platform state* at iteration t:

$$\bar{\mathbf{x}}^t = \left[\mathbf{p}^t\ \mathbf{v}^t\ \mathbf{q}^t\right], \tag{1}$$

which is actually a column vector where $\mathbf{p}^t = \left[p_x^t\ p_y^t\ p_z^t\right]^T$ is the position of the robot, $\mathbf{v}^t = \left[v_x^t\ v_y^t\ v_z^t\right]^T$ is the velocity and $\mathbf{q}^t = [q_0^t\ q_1^t\ q_2^t\ q_3^t]^T$ is the quaternion representation of the vehicle orientation, being q_0^t the real part of the quaternion. All three vectors are expressed with respect to a given reference frame, which can be the starting point of operations.

Position part, $\mathbf{p}^t$, which is the final goal variable to be estimated, will be computed by integrating the optimized linear velocities, out of the minimization loop, since neither IMU nor cable encoder measure the position component of the state $\bar{\mathbf{x}}^t$ (they measure derivatives or indirect values). Therefore we redefine the platform state in a dimensional-reduced way, much more efficient to be used by an optimization procedure as it will be explained on section III. The so called *platform state* is:

$$\mathbf{x}^t = \left[v_x^t\ v_y^t\ v_z^t\ q_1^t\ q_2^t\ q_3^t\right]^T. \tag{2}$$

Only velocity components and imaginary part of the quaternion have been kept, since they are the independent values to be estimated directly. Real part of the quaternion can be computed by forcing $|\mathbf{q}^t| = 1$.

The trajectory run by the robot is modelled as a graph linking consecutive nodes, where each node represents a platform state at a given iteration t, and each link sets a constraint imposed by sensor data between two of such nodes. Figure 1 shows the *raw graph* issued from this model, composed by $N+1$ platform states.

Since IMU data is provided at much higher rate than cable encoder measurements, localization will be computed for a subset of those platform states depicted on figure 1, so a new version of the graph, called *reduced graph*, is thought which only incorporates such selected platform states, also called keyframes. Figure 2 shows the reduced graph, where all nodes are linked by a cable encoder measurement and by an integration of a subset of consecutive IMU measurements. Based on this reduced graph we will formally define a window as a subset of N_w+1 consecutive platform states (i.e. N_w+1 consecutive keyframes).

So at this point three iteration index can be formally defined. Assuming that IMU is the highest rate sensor, the first one is the index running on IMU iterations, $t \in [0, N]$. The second index is that of those iterations marking keyframes, $i \in [0, N_K]$, and the third index runs over a *window* state, $j \in [0, N_w]$. A *window state* is defined as:

$$\mathbf{s}^i = \begin{bmatrix} \mathbf{x}^i & \mathbf{x}^{i+1} & \dots & \mathbf{x}^{i+j} & \dots & \mathbf{x}^{i+N_w} \end{bmatrix}, \tag{3}$$

which is also a column vector concatenating N_w+1 platform states, from $\mathbf{x}^i$ to $\mathbf{x}^{i+N_w}$.

Please note that t, i and j indexes are integer values. Each of them has a corresponding time stamp value labelled, respectively, as τ_t, τ_i and τ_{i+j}, allowing proper ordering operations between them.

III. State Optimization

State estimation will be computed for each window over the window state space defined by $\mathbf{s}^i$. However, we assume the first state of each window, $\mathbf{x}^i$, as a starting point, so it is *anchored* and optimization will not modify it. The cost function to be minimized is:

$$F(\mathbf{s}^i) = F_I(\mathbf{s}^i) + F_C(\mathbf{s}^i) + F_W(\mathbf{s}^i) \tag{4}$$

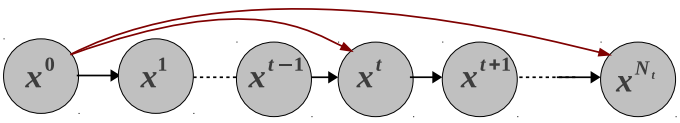

Fig. 1. Raw graph created when fusing IMU and cable constraints. Black arrows are IMU measurements and red ones represent cable length measurements from the starting motion point.

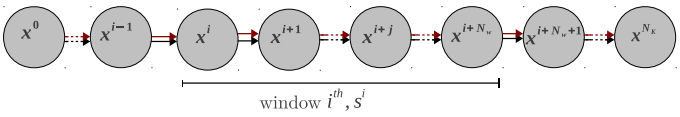

Fig. 2. Reduced graph with N_K+1 keyframe platform states. Each pair of neighboring nodes are linked by two constraints: (1) Black, IMU constraints imposed by integrating several consecutive inertial measurements, and (2) Red, cable length constraints imposed as the difference of two consecutive cable encoder measurements.

where the first term $F_I(\mathbf{s}^i)$ is the cost function associated to IMU measurements, the second one $F_C(\mathbf{s}^i)$ is that associated to cable encoder data and the third one, $F_W(\mathbf{s}^i)$ is that imposing the wheeled vehicle constraints. These three function costs are detailed in the following section III. Please note that $F(\mathbf{s}^i)$ does not have an explicit expression, but given a set of sensor measurements, and given a point $\mathbf{s}^i$, we can compute its value (pointwise evaluation), so numerical techniques will be employed to compute the required minimization. The goal for each optimization is to find the $\mathbf{s}^i_o$ point, which minimizes the function $F(\mathbf{s}^i)$:

$$\mathbf{s}^i_o = argmin F(\mathbf{s}^i) \tag{5}$$

A. IMU Constraint

An IMU device provides high rate (up to 1 KHz) inertial measurements. At iteration t, the IMU measurement is:

$$\boldsymbol{\iota}^t = \begin{bmatrix} \alpha^t_x & \alpha^t_y & \alpha^t_z & \omega^t_x & \omega^t_y & \omega^t_z \end{bmatrix}^T, \tag{6}$$

where $\boldsymbol{\alpha}^t = \begin{bmatrix} \alpha^t_x & \alpha^t_y & \alpha^t_z \end{bmatrix}^T$ is the body acceleration vector and $\boldsymbol{\omega}^t = \begin{bmatrix} \omega^t_x & \omega^t_y & \omega^t_z \end{bmatrix}^T$ is the rotational rate vector, both measurements reported with respect to the current vehicle coordinate frame. With these measurements, a set of constraints can be imposed to the pose graph, but these constraints are not formulated on such measurement space. Alternatively, consecutive measurements are integrated to compute the so called *integrated* platform states, labelled as $\mathbf{x}_\int$. Therefore, for the i^{th} window, the following vector is defined:

$$\boldsymbol{\delta}^i_I = \left[(\mathbf{x}^{i+1}_\int - \mathbf{x}^{i+1})^T \ \dots (\mathbf{x}^{i+N_w}_\int - \mathbf{x}^{i+N_w})^T \right]^T, \tag{7}$$

where terms $\mathbf{x}^{i+j}, \forall j \in [1, N_w]$ are provided by the evaluation point $\mathbf{s}^i$, while $\mathbf{x}^{i+j}_\int, \forall j \in [1, N_w]$ are computed with $\mathbf{x}^i$ initial (anchored) point and IMU measurements $\{\boldsymbol{\iota}^t | \tau_t \in [\tau_i, \tau_{i+N_w}]\}$. The cost function to be minimized associated to IMU measurements is:

$$F_I(\mathbf{s}^i) = (\boldsymbol{\delta}^i_I)^T (\mathbf{C}^i_I)^{-1} \boldsymbol{\delta}^i_I, \tag{8}$$

so at this point we have to provide a computer routine that, given measurements during the window period, and given an evaluation point $\mathbf{s}^i$, is able to output $F_I(\mathbf{s}^i)$. Therefore, we have to detail computation of $\mathbf{x}^{i+j}_\int$, and also matrix $\mathbf{C}^i_I$, which parameterizes the uncertainty of IMU measurements, thus $(\mathbf{C}^i_I)^{-1}$ weights the overall IMU constraint.

Within the i^{th} window, the first integrated state, $\mathbf{x}^i_\int$, is provided by the first (anchored) platform state of the evaluation point $\mathbf{s}^i$, and the following integrated platform states, $\mathbf{x}^t_\int$ are computed with the following equations:

$$\begin{aligned} \mathbf{v}^t_\int &= \mathbf{v}^{t-1}_\int + \mathbf{R}^{t-1}(\boldsymbol{\alpha}^t - \mathbf{b}_\alpha)\Delta^t - \mathbf{g}^0 \Delta^t \ , \\ \mathbf{q}^t_\int &= \mathbf{W}^t \mathbf{q}^{t-1}_\int \ , \\ \mathbf{q}^t_\int &\leftarrow \frac{\mathbf{q}^t_\int}{|\mathbf{q}^t_\int|} \ \text{(quaternion normalization)} \ , \end{aligned} \tag{9}$$

where Δ^t is the time elapsed between time stamps τ_{t-1} and τ_t, $\mathbf{R}^{t-1} = \mathbf{R}(\mathbf{q}^{t-1}_\int)$ is the rotation matrix associated to the platform orientation at $t-1$, $\mathbf{g}^0$ is the gravity vector, and $\mathbf{b}_\alpha$ and $\mathbf{b}_\omega$ are the accelerometer and gyro bias respectively. Gravity, and bias are estimated initially as it is explained in

subsection III-D. Recall that from quaternion representation, rotation matrix is:

$$\mathbf{R}(\mathbf{q}) = 2 \begin{bmatrix} \frac{1}{2} - q_2^2 - q_3^2 & q_1 q_2 - q_0 q_3 & q_0 q_2 + q_1 q_3 \\ q_0 q_3 + q_1 q_2 & \frac{1}{2} - q_1^2 - q_3^2 & q_2 q_3 - q_0 q_1 \\ q_1 q_3 - q_0 q_2 & q_0 q_1 + q_2 q_3 & \frac{1}{2} - q_1^2 - q_2^2 \end{bmatrix}, \tag{10}$$

and matrix $\mathbf{W}^t$ is the small angle approximation of the measured rotational increment $\boldsymbol{\omega}^t \Delta^t$ (typically $\Delta^t < 10^{-2}\ s$):

$$\mathbf{W}^t = \frac{\Delta^t}{2} \begin{bmatrix} 2/\Delta^t & -(w_x^t - b_{\omega_x}) & -(w_y^t - b_{\omega_y}) & -(w_z^t - b_{\omega_z}) \\ (w_x^t - b_{\omega_x}) & 2/\Delta^t & (w_z^t - b_{\omega_z}) & -(w_y^t - b_{\omega_y}) \\ (w_y^t - b_{\omega_y}) & -(w_z^t - b_{\omega_z}) & 2/\Delta^t & (w_x^t - b_{\omega_x}) \\ (w_z^t - b_{\omega_z}) & (w_y^t - b_{\omega_y}) & -(w_x^t - b_{\omega_x}) & 2/\Delta^t \end{bmatrix}. \tag{11}$$

On the other hand, matrix $\mathbf{C}_I^i$ will weight the constraint, based on the estimated uncertainty, which depends on the model and on the noisy measurements. So matrix $\mathbf{C}_I^i$ will be of the form:

$$\mathbf{C}_I^i = \begin{bmatrix} \mathbf{C}_\int^{i+1} & & & \\ & \mathbf{C}_\int^{i+2} & & \\ & & \ddots & \\ & & & \mathbf{C}_\int^{i+N_w} \end{bmatrix}, \tag{12}$$

where

$$\mathbf{C}_\int^t = \mathbf{J}_1 \mathbf{C}_\int^{t-1} \mathbf{J}_1^T + \mathbf{J}_2 \mathbf{Q}_I \mathbf{J}_2^T, \tag{13}$$

with Jacobian $\mathbf{J}_1$:

$$\mathbf{J}_1 = \frac{\partial \mathbf{x}_\int^t}{\partial \mathbf{x}_\int^{t-1}} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{J}_{1,vq} \\ \mathbf{0}_3 & \mathbf{J}_{1,qq} \end{bmatrix}, \tag{14}$$

Jacobian $\mathbf{J}_2$:

$$\mathbf{J}_2 = \frac{\partial \mathbf{x}_\int^t}{\partial \boldsymbol{\iota}^t} = \begin{bmatrix} \mathbf{J}_{2,v\alpha} & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{J}_{2,q\omega} \end{bmatrix}, \tag{15}$$

and matrix $\mathbf{Q}_I$ is the IMU device noise matrix:

$$\mathbf{Q}_I = \begin{bmatrix} \sigma_\alpha^2 \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \sigma_\omega^2 \mathbf{I}_3 \end{bmatrix}. \tag{16}$$

Jacobian matrices $\mathbf{J}_{1,vq}$, $\mathbf{J}_{1,qq}$, $\mathbf{J}_{2,v\alpha}$ and $\mathbf{J}_{2,q\omega}$ are partial derivatives of $\mathbf{v}_\int^t$ and $\mathbf{q}_\int^t$ with respect to $\mathbf{q}_\int^{t-1}$, $\boldsymbol{\alpha}^t$ and $\boldsymbol{\omega}^t$.

Algorithm 1 summarizes the procedure to compute the IMU part of the cost function. Instead of accumulating a large δ_I vector and $\mathbf{C}_I$ matrix, the algorithm updates the cost by computing the contribution at each j node as a mahalanobis distance between $\mathbf{x}_\int$ and $\mathbf{x}^{i+j}$. This is equivalent to equation 8 since $\mathbf{C}_I$ matrix is a block-built matrix, but avoids computation of the inverse of a large matrix.

B. Cable Length Constraint

During the time period of the i^{th}window, measurements acquired by the cable encoder are the set $\{\lambda^{i+j} | \tau_{i+j} \in [\tau_i, \tau i + N_w]\}$, so keyframes are directly choosen when new cable encoder data is available. Cable encoder measures the total unfolded length of the cable tethering the robot. Within a window, we define the following difference:

$$\ell^i = \lambda^{i+N_w} - \lambda^i, \tag{17}$$

Algorithm 1 IMU Cost Function

INPUTS
. WINDOW STATE POINT: $\mathbf{s}^i$
. SENSOR DATA: $\{\iota^t, \forall t | \tau_t \in [\tau_i, \tau_{i+N_w}]\}$
. KEYFRAME TIMESTAMPS: $\{\tau_{i+j}, \forall j = 1 \dots N_w\}$
. PARAMETERS: $\mathbf{g}^0, \mathbf{b}_\alpha, \mathbf{b}_\omega, \sigma_\alpha, \sigma_\omega$
OUTPUT: $F_I(\mathbf{s}^i)$

```
x_∫ = x^i //anchored platform state, x^i = s^i[1:6]
C_∫ = [0_6]
j = 1
F_I = 0
while τ_t < τ_{i+N_w} do
    x_∫, C_∫ ← imuIntegration(x_∫, C_∫, ι^t, g^0, b_α, b_ω)
    t++
    if τ_t > τ_{i+j} then
        F_I += (x_∫ − x^{i+j})^T C_∫^{-1} (x_∫ − x^{i+j})
        j++
    end if
end while
return F_I
```

which is the difference between measured cable lengths at the end and start points of the window. Given an evaluation point $\mathbf{s}^i$, an expected length difference, $\hat{\ell}^i$, can be computed as the sum of the forward vehicle increments within the window, being $\mathbf{u}_F$ the normal forward vector with respect to the vehicle frame:

$$\hat{\ell}^i = \sum_{j=0}^{N_w - 1} \mathbf{u}_F (\mathbf{R}^{i+j})^{-1} \mathbf{v}^{i+j} \Delta_{i+j}^{i+j+1}, \tag{18}$$

which implicitly assumes that cable deployment follows the inner pipe geometry, and primary force that pulls the cable deployment is due to forward vehicle motion. Δ_{i+j}^{i+j+1} is the ellapsed time between consecutive keyframes $i+j$ and $i+j+1$.

Given cable measurements and the evaluation point $\mathbf{s}^i$, the difference between expected, $\hat{\ell}^i$, and measured, ℓ^i, is:

$$\delta_C^i(\mathbf{s}^i, \lambda^i, \lambda^{i+N_w}) = \hat{\ell}^i(\mathbf{s}^i) - \ell^i(\lambda^i, \lambda^{i+N_w}). \tag{19}$$

At this point, we could follow the same approach described by IMU function cost, but we realise that, within window intervals, noise of cable measurement differences was mainly due to resolution error, since encoder measures up to $1cm$ cable length variations. This fact inspires us an alternative probabilistic model rather than a Gaussian one. The proposed approach aims to model that, within the encoder resolution range, measurements should be very likely, but out of this resolution interval, probability should fall down rapidly. In terms of a cost function, $F_C(\mathbf{s}^i)$, the proposed model results at zero cost for values of δ_C^i inside the encoder resolution range, but the cost grows up when δ_C^i is out of this resolution tolerance interval. This model is summarized with the following cost function:

$$F_C(\mathbf{s}^i) = F_C(\delta_C^i(\mathbf{s}^i)) = \begin{bmatrix} 0, & |\delta_C^i| < \gamma_C \\ (\delta_C^i)^2 / \sigma_C^2 \end{bmatrix} \tag{20}$$

Other noise sources could be the non-straight deployment of cable length, but since the proposed model computes differences within the starting point of the window and the end

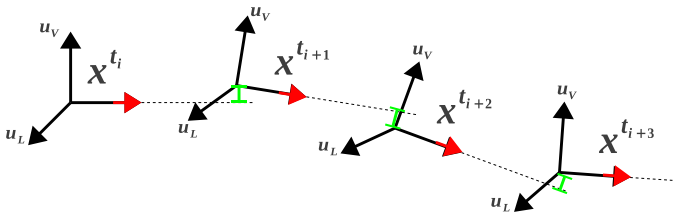

Fig. 3. Small green segments are parts of accumulated vertical displacement over a window of $N_w = 3$. Lateral displacement follows the same idea.

one (equation 17), main cable curvatures are cancelled since they affect both cable length measurements λ^i and λ^{i+N_w}.

C. Wheel Constraint

Wheel constraint can be applied under the assumption that the platform is wheeled, and slippage and jumping are relatively low. It assumes the heuristics that lateral and vertical displacements of the vehicle are small, so it bounds these motions. For a window state point $\mathbf{s}^i$, lateral and vertical displacements, δ^i_L and δ^i_V, can be computed as:

$$\delta^i_{L,V}(\mathbf{s}^i) = \sum_{j=0}^{N_w-1} \mathbf{u}_{L,V}(\mathbf{R}^{i+j})^{-1}\mathbf{v}^{i+j}\Delta^{i+j+1}_{i+j}, \qquad (21)$$

where $\mathbf{u}_{L,V}$ are the lateral and vertical axis of the vehicle coordinate frame. Figure 3 shows the vertical displacements in a window of $N_w = 3$.

The proposed model allows, up to a certain threshold, non-forward displacements (lateral or vertical). So the probabilistic model is neither Gaussian as it wasn't for the cable constraint discussed in subsection III-B. We use a similar approach, which in terms of cost function is modelled as equation 20, but with parameters γ_L, σ_L and γ_V, σ_V.

Algorith 2 summarizes how cable and wheel constraints are computed.

Algorithm 2 CABLE & WHEEL Cost Function

INPUTS
. WINDOW STATE POINT: $\mathbf{s}^i$
. SENSOR DATA: $\{\lambda^{i+j} | \tau_{i+j} \in [\tau_i, \tau_{i+N_w}]\}$
. KEYFRAME TIMESTAMPS: $\{\tau_{i+j}, \forall j = 1 \dots N_w\}$
. PARAMETERS: $\sigma_C, \gamma_C, \sigma_L, \gamma_L, \sigma_V, \gamma_V$
OUTPUT: $F_C(\mathbf{s}^i), F_W(\mathbf{s}^i)$

$\ell^i = \lambda^{i+N_w} - \lambda^i$
$\hat{\ell}^i = 0$
for $j = 0; j < N_w; j++$ **do**
  $\mathbf{d} = (\mathbf{R}^{i+j})^{-1}\mathbf{v}^{i+j}\Delta^{i+j+1}_{i+j}$
  $\hat{\ell}^i += \mathbf{u}_F \cdot \mathbf{d}$
  $\delta^i_L += \mathbf{u}_L \cdot \mathbf{d}$
  $\delta^i_V += \mathbf{u}_V \cdot \mathbf{d}$
end for
$\delta^i_C = \hat{\ell}^i - \ell^i$
$F_C = F_C(\delta^i_C, \sigma_C, \gamma_C)$ //equation 20
$F_L = F_L(\delta^i_L, \sigma_L, \gamma_L)$ //equation 20
$F_V = F_V(\delta^i_V, \sigma_V, \gamma_V)$ //equation 20
$F_W = F_L + F_V$
return F_C, F_W

D. IMU Initialization

IMU device is mainly affected by two kinds of noise processes. One of them is considered as a Gaussian process, which is taken into account by weighting the IMU constraint by the inverse of a covariance matrix (equation 12). But a second noise process is of low dynamics, and it can be modelled as a measurement bias for accelerometers, $\mathbf{b}_\alpha$ and another bias for the gyros, $\mathbf{b}_\omega$. Moreover, accelerometers also measure the ubiquitous gravity vector, $\mathbf{g}^0$, that is compensated in ground applications with the force experimented by the floor. So three vectors have to be estimated initially, in order to use IMU data properly, $\mathbf{g}^0$, $\mathbf{b}_\alpha$ and $\mathbf{b}_\omega$, since our approach does not observe such quantities online. Such initalization assumes that the platform is completely stopped while running the optimization loop, forcing the integrated velocities to be 0 and the orientation quaternion to be $[1\ 0\ 0\ 0]^T$. So the state to be optimized is the 9-dimensional vector of $\left[\mathbf{g}^0\ \mathbf{b}_\alpha\ \mathbf{b}_\omega\right]$. Initial guess for the gravity is the mean acceleration during a short period (for instance, one second), and initial bias are set to zero vectors. Subsection V-B reports results on runnig such initialization with the actual platform used for the experiments.

E. Initial Guess

When calling a minimization routine, an initial guess has to be provided, labelled as $\tilde{\mathbf{s}}^i$, which is a window state point from which the optimization will start computations. For i^{th} window, available measurements are α^t, ω^t, Δ^t, $\forall t | \tau_t \in [\tau_i, \tau_{i+N_w}]$ and λ^{i+j}, $\forall j | \tau_{i+j} \in [\tau_i, \tau_{i+N_w}]$. Moreover we know also the past estimate history, $\{\mathbf{x}^0 \dots \mathbf{x}^{i-1} \dots \mathbf{x}^{i+N_w-1}\}$. So the initial guess for the optimizer, provided as an initial window state point is:

$$\tilde{\mathbf{s}}^i = \{\mathbf{s}^{i-1} \setminus \mathbf{x}^{i-1}, \mathbf{x}^{i+N_w}_f\}, \qquad (22)$$

which indicates that optimization result of the last window is used as an initial guess for the current i^{th} window, poping front $\mathbf{x}^{i-1}$ and pushing back an IMU prediction for the last platform state, $\mathbf{x}^{i+N_w}_f$.

At the beginning, the first platform state point is set to zero, $\tilde{\mathbf{x}}^0 = [0\ 0\ 0\ 0\ 0\ 0\ ]^T$, which assumes a stopped platform before operations and sets the localization coordinate reference frame at the initial point of platform motion.

IV. Implementation Details

A software prototype has been implemented in C++ under Ubuntu 12.04, by using the algebra and optimization classes of dlib library [3]. This section details three practical issues: timeline, skipping cable data and threshold value.

a) Timeline: Since IMU and encoder devices provide data asynchronously and at different rates, timeline concepts are critical, and iteration indexes are based on them. Figure 4 shows the continuous timeline and the different involved indexes running over it. As suggested in figure 4, the first platform state of the window is called *anchored*, since their components are not part of the optimization space, but it is required as a necessary starting point to integrate IMU data (see subsection III-A), as well as to compute expected forward displacements (see subsection III-B).

b) Skipping Cable Data: Cable encoder measures, at $10Hz$, the length of the unfolded cable tethering the robot, with a $1cm$ resolution error. In the other hand, IMU is a much higher sensitive device that measures inertial data at $100Hz$, so a big difference in the nature of both measurements exists. Since robot speeds are usually below $0.5m/s$, it has been observed that skipping (ignoring) several consecutive cable data points results in better optimized trajectories, computed faster, probably because of cable data needs *amplitude* to have an important corrective effect over IMU data. Otherwise, within a short window, cable data falls under the resolution noise level. So a parameter N_s is set, indicating how many consecutive cable data points are periodically skipped from the raw cable data stream. Values for N_s will be typically between 0 and 3. Higher values of N_s could cause large extrapolation errors when computing expected forward increments (see equation 18).

c) Threshold Values: In subsections III-B and III-C a non-Gaussian probabilistic model has been proposed, depending on a threshold. For cable length cost function, F_C (see equation 20), the threshold is γ_C, which is set to $0.5cm$, so algorithm allows absolute differences between expected and measured cable increments up to $0.5cm$ within a window. Beyond γ_C, cost function behaves as a mahalanobis distance with σ_C set to $1cm$. For wheel constraint case, threshold $\gamma_L = \gamma_V = 1cm$, so the maximum slippage/flying/falling tolerated by the model will be up to $1cm$ inside a window, and $\sigma_L = \sigma_V = 1mm$.

V. Experimental Results

A. Data Set & Ground Truth

On 6th and 7th July 2012, at JT headquarters [6], in Lindau, Germany, several data acquisition sessions were carried out, with the aim of having a data set of a mobile/tethered platform (figure 5) running inside a measured pipe, while collecting data from an IMU device and a cable encoder, both data streams stored with timestamps provided by the same machine.

The data set consists on several forward and backward trajectories, with some trials at constant velocity and other trials changing speeds, but always running inside a measured pipe, pictured on the right side of figure 6. A scaled and

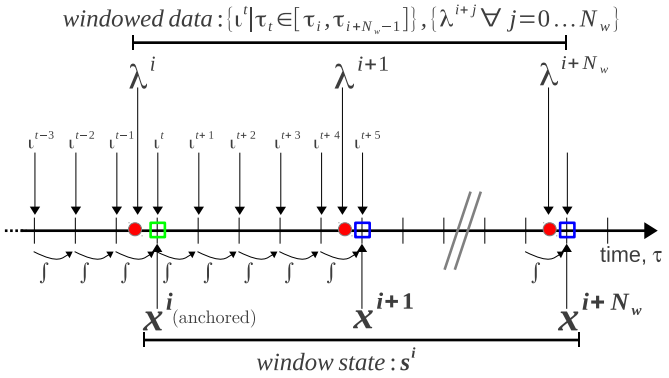

Fig. 4. Different iteration indexes running over timeline. Red points on the timeline mark cable data timestamp and small black ticks mark IMU timestamp. Sensor data used to optimize i^{th} window is grouped on top, and platform states forming the window state are grouped below.

Fig. 5. JT wheeled platform used to collect data.

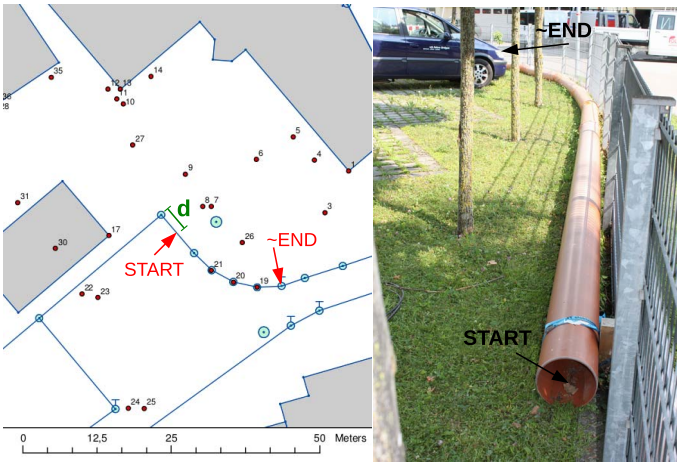

Fig. 6. On the left, a georeferenced map of the JT garden area, where the pipe run by the robot is deployed. On the right, a picture of that pipe. Distance between a reference corner and the start of the pipe is marked with a d, and was measured to $3.1m$.

georeferenced map, shown on the left side of figure 6, was used to extract a ground truth of such pipe, so estimated trajectories can be compared with the reference pipe layout. Tractor velocities were always below $0.4m/s$.

Even if this ground truth data is metrically precise, it does not represent exactly the true trajectory done by the robot. Instead, it represents the geometry of the pipe layout where the experiments were performed.

B. Gravity and Bias Estimation

As discussed in subsection III-D, IMU integration requires to initialize the gravity vector, as well as accelerometer and gyroscope bias. Before each test, we used a sequence of $15s$ duration of IMU data, with the platform completely stopped to compute these three vectors. For illustrative purposes, the following are the values of the IMU bias for the first trial test:

$$\mathbf{b}_\alpha = [0.0018887 \quad -0.0017985 \quad -0.0031113]^T \ m/s^2$$
$$\mathbf{b}_\omega = [-0.0000181 \quad 0.0000641 \quad 0.0000574]^T \ rad/s$$

Gravity was also estimated in the same loop. For the case of the same trial as above, estimated gravity initialization was:

$$\mathbf{g}^0 = [-0.6064705 \quad 9.7851897 \quad -0.1997489]^T \ m/s^2 \quad (23)$$

C. Trajectory Estimation

This section provides results of the estimated trajectory run by the robot inside the pipe shown in figure 6. Several executions of the optimization were run, modifying two parameters in order to tune them for adequate values. The parameters to

be tunned were N_s and N_w. N_s is the number of cable data skip points, being $N_s = 0$ if all points are taken, so 0 points are skipped. The other parameter to investigate, N_w, is the window length, being $N_w = 1$ the reduced case where only one platform state is optimized. Both parameters have effects on the computation time required to execute the estimation loop. Current implementation runs on a standard laptop PC (intel core i5), keeping real-time peformance when $N_w < 10$. Figure 7 plots results obtained running four executions, setting $N_w = 7, 15$ and $N_s = 0, 1$.

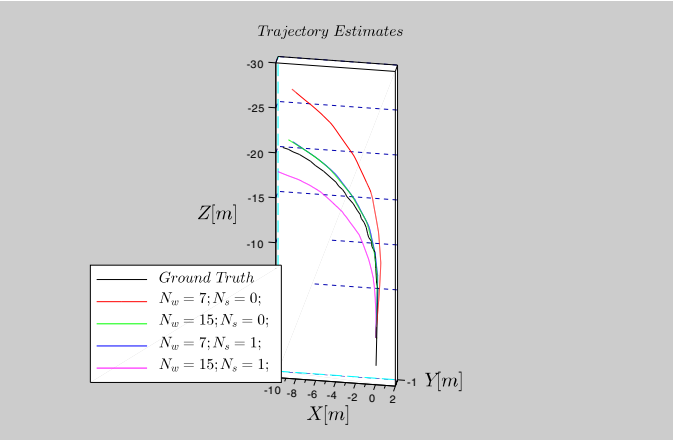

Fig. 7. Estimated trajectories with different values for N_w and N_s compared with ground truth pipe profile (black).

We were also interested to identify the effect of imposing each constraint separately, so next results compare trajectory estimates (figure 8) for four cases: (1) only IMU constraint, (2)IMU and cable constraints, (3) IMU and wheel constraints and (3) IMU,cable and wheel constraints. All executions were done by setting $N_w = 7$ and $N_s = 1$. We see how, only when all three constraints are jointly considered, estimates are close to ground truth. In the case where cable constraint is not considered, forward vehicle velocity has no bound to grow up, so an amplified trajectory arises (blue plot in figure 8).

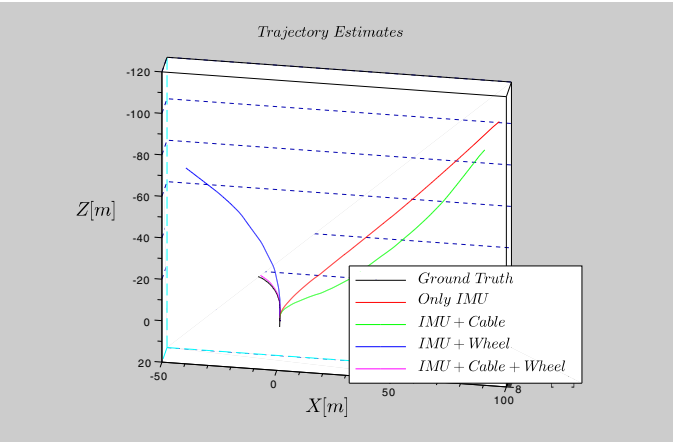

Fig. 8. Estimated trajectories considering different constraint sets. For all cases, $N_w = 7$ and $N_s = 1$.

VI. Conclusion

This paper proposes a complementary technique for visual/inertial localization of a mobile robot intended to be used in sewer pipe network scenarios. The approach uses sensor data from an IMU and from a cable encoder, the later measuring the length of an unfolded cable linking the mobile robot with its starting point of operations. Data fusion is based on the Graph SLAM framework and also requires to add extra constraints provided by assuming low slippage and jumping of the platform, which are reasonable assumptions for wheeled inspection platforms operating at low speeds. Optimization is preformed through a set of consecutive platform states. The reported results show that the approach reduces considerably the IMU drift, so it is well suited to be considered as a complementary technique to solve situations where visual feature tracking could fail, due to environment situations, such as low texture areas, hard light reflections or sudden water flows or drops.

Acknowledgment

This work is carried out under the European FP7-SME PipeGuard project with agreement number 286580. Authors would also thank the engineering support received by JT-elektronik and Inspiralia, which are also partners of the project.

References

[1] M. Agrawal and K. Konolige. FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping. *IEEE Transactions on Robotics*, 24(5), October 2008.

[2] A.J. Davison. Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proceedingg of the International Conference on Computer Vision (ICCV)*, 2003.

[3] dLib. dlib c++ library website. dlib.net.

[4] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2:31–43, 2010.

[5] P. Hansen, H. Alismail, P. Rander, and B. Browning. Monocular visual odometry for robot localization in LNG pipes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China. May, 2011.

[6] JT-elektronik. Jt-elektronik website. www.jt-elektronik.de/.

[7] M.I. A. Lourakis and A.A. Argyros. SBA: A Software Package for Generic Sparse Bundle Adjustment. *ACM Trans. Math. Software*, 36(1):1–30, 2009.

[8] T. Lupton and S. Sukkarieh. Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions. *IEEE Transactions on Robotics*, 28(1), 2012.

[9] M. Maimone, Y. Cheng, and L. Matthies. Two Years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, 3(24):169–186, 2007.

[10] A. Martinelli. Vision and IMU Data Fusion: Closed-Form Solutions for Attitude, Speed, Absolute Scale, and Bias Determination. *IEEE Transactions on Robotics*, February 2012.

[11] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. Rslam: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, pages 1 – 17, June 2010.

[12] D. Nistér, O. Naroditsky, and J. Bergen. Visual Odometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[13] S. Thrun and M. Montemerlo. The GraphSLAM Algorithm With Applications to Large-Scale Mapping of Urban Structures. *International Journal of Robotics Research*, 25:403–430, 2006.