A modular architecture for mobile robots equipped with continuous-discrete observers

Gian Marco Vinco, Philipp Braun, Luca Zaccarian

A modular architecture for mobile robots equipped with continuous-discrete observers

Gian Marco Vinco[1], Philipp Braun[2], Luca Zaccarian[1,3]

***Abstract*—We illustrate the design and development of a modular hardware/software system for multiple unicycle-like mobile robots localized via a set of camera modules. We describe the architecture and calibration of the hardware/software setup and then discuss two continuous-discrete observation laws for the distributed estimation of the robot positions. We show that a suitable model exploiting the onboard IMU measurements of the robots, enables obtaining an estimation error that is a cascade of two linear systems, for which we can show global exponential convergence to zero. The results are illustrated by our experimental tests.**

***Index Terms*—Nonlinear estimation, Hybrid Systems, Controller Hardware/Software, Autonomous mobile robots.**

I. Introduction

Control and navigation of unmanned vehicles such as mobile robots or drones experiences a continuing research interest not only in military applications. For example, on an experimental scale, drones have been used to deliver medical equipment [1], and fleets of mobile robots may replace humans and conveyor belts in warehouses [2]. For such control applications reliable models to estimate the states of the system are necessary, and common approaches usually combine visual measurements with measurements from inertial measurement units (IMUs) in the observer design (see, e.g., [3]–[6]).

In this context, measurements are usually available at different time scales. IMU data are essentially continuous measurements while visual measurements are obtained at a significantly slower rate and potentially not periodically. Solely relying on IMU measurements does not provide observability of the position dynamics. Conversely, even if observability holds when using the visual measurements only, due to the generally slow rate of these measurements, a reliable state estimation is difficult to guarantee if the IMU data is ignored. Since IMUs are cheap as compared with cameras, it is not surprising that common approaches fuse data from cameras and IMUs. As examples, in [6] a multirate moving horizon estimator for mobile robots is discussed, while the approaches in [4], [5] rely on (extended) Kalman filters.

In this paper we follow this stream by developing a low cost control architecture for mobile robots consisting of *camera client units* and *Makeblock mBot robot units*, both equipped with *Raspberry Pis*. The overall setup and the acquisition system are described in Section II. The camera client units provide delayed sporadic measurements of the position and the orientation of the robots and the robots are equipped with gyroscopes and accelerometers providing, respectively, angular velocity and linear acceleration measurements, which can be considered as continuous signals in the context of this paper.

In Section III two models and corresponding observers for the Makeblock mBot robot units are derived. For the observer design we use the hybrid systems formalism of [7], acknowledging the peculiar nature of continuous-time systems with discrete-time measurements. The first observer described in Section III is a standard continuous-discrete extended Kalman filter [8, Sec. 13.2.2], [9, Sec. 5.3.3] applied to the unicycle dynamics [10, Sec. 2.3.1], which only uses the camera measurements and ignores the IMU data. The second set of dynamics is derived based on the hybrid time observer developed in [11], [12] but could have alternatively been written using an extended Kalman filter. In contrast to the first observer, the second observer combines the discrete-time position measurements with the velocity and acceleration IMU measurements. Initial results on the performance of the control architecture are given in Section IV. The control architecture provides ample opportunities for extensions, which are reported in Section V, in our conclusions.

II. Mobile robot control architecture

The setup, represented in Fig. 1, consists of a centralized communication system, including a *main server* in the role of a hub for the data exchange and a scalable number of *Camera Client Units* and *Robot Client Units*. Each camera client unit measures the position and orientation of the robots in its field of view, with a specific acquisition frame rate, and

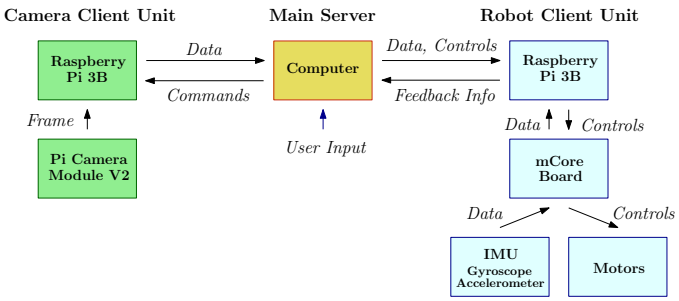

Fig. 1. Main components of the mobile robot control architecture and their interconnections.

The research presented in this paper was conducted while P. Braun was with the University of Newcastle, Australia, and while G. M. Vinco was visiting the University of Newcastle, Australia.

[1]Dept. of Industrial Engineering, University of Trento, Trento 38123, Italy.
[2]Research School of Electrical, Energy, and Materials Engineering, Australian National University, Canberra, Australia (e-mail: philipp.braun@anu.edu.au).
[3] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France (e-mail: luca.zaccarian@laas.fr).

sends the results to the main server. The robot client units are equipped with an onboard microprocessor on a Raspberry Pi card. Additionally, the robot client units include onboard IMU sensors comprising gyroscopes and accelerometers. The communication is based on the TCP/IP protocol and designed as a master-slave model, managed by the *main server*. The system is modular and scalable because each unit works as a complete standalone system, which allows for multiple camera and/or robot units. We describe below the units architecture and the actuators/sensors calibration.

A. Camera and robot client units

1) Camera units: Each camera unit consists of a Raspberry Camera module V2 serially connected to a dedicated Raspberry Pi 3B+ microprocessor. The connection is established through the I2C serial port of the microprocessor. The camera acquisition parameters (such as resolution, frame rate and brightness) are set during the initialization process of the unit. After the initialization, the TCP/IP communication with the *main server* is established through the generation of a specific socket channel. Each camera unit uses the channel to send the robot localization data to the server. The robot localization is performed as a marker identification process through the OpenCV Computer Vision library.

Each robot is provided with a specific pair of color markers (a pointer and a center marker – see the bottom-right inset in Fig. 2) enabling position and orientation estimation from the camera image. The marker recognition process is performed in two steps. (1) *Color filtering* is used to identify the presence of the markers at each frame capture. (2) *Morphological transformations* to isolate the pixels corresponding to the (possibly) identified marker and to remove frame background disturbances. With the central marker placed at the mBot center of mass and aligned with the robot longitudinal axis, after some data conversions, the above localization process returns an estimate of the robot position and orientation $[\hat{p}_x(t^*)\ \hat{p}_y(t^*)\ \hat{\phi}(t^*)]^\top \in \mathbb{R}^3$ in the (x, y) plane. The localization process takes about 0.4 s, i.e., a constant delay of $\Delta = 0.4$ seconds occurs between the measurement (snapshot) time t^* and the time $t^* + \Delta$ when the measurement is available at the robot unit. Finally, while the camera frame rate was set to 40 Hz, most of the produced frames were disregarded due to the above mentioned camera processing time Δ.

Remark 1: The quality of the marker identification strictly relies on the accuracy of the initial camera calibration. During the initialization, it is necessary to define the colors of the center marker, that the camera is supposed to identify through the color filter. Each color is defined as a triple of values in the HSV colorspace. The calibration is performed manually by the user which can adjust the three values for each marker. Additionally, the calibration can be remotely adjusted at any time by sending specific control commands from the *main server* to the camera unit. Since this calibration is a fairly standard process, we do not discuss it in Section II-B. Additional details can be found in [13, §1.1]. ∘

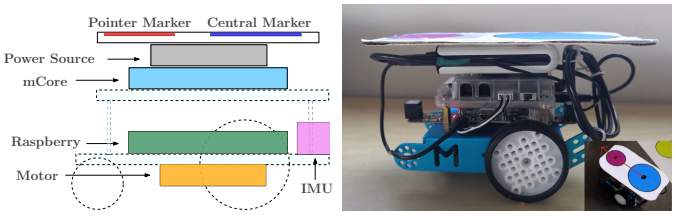

Fig. 2. Components of each mobile robot (left) and a snapshot (right).

2) Robot units: Each robot unit consists of a Makeblock mBot unicycle comprising an onboard mCore microcontroller and a pair of motors (see Fig. 2). The off-the-shelf device is augmented with a Raspberry Pi 3B+ serially connected to the mCore. This allows us to connect the robot unit with the *main server* through a TCP/IP socket channel. Thanks to a dedicated Simulink software package for Raspberry Pi boards, it is possible to develop Simulink models and deploy them on the Raspberry microprocessor. These models run as standalone processes in the background of the main robot script and can be used to run decentralized observers and controllers.

Each mobile robot unit is sensorized with a 3-axis gyroscope and a 3-axis accelerometer, embedded in the Makeblock MPU6050 IMU unit. The IMU provides measurements of the robot forward and lateral accelerations $a = [a_{\text{fw}}\, a_{\text{lat}}]^\top \in \mathbb{R}^2$, as well as measurements of the angular velocity $\omega \in \mathbb{R}$ (with respect to the z-axis) with a maximum rate of 200Hz. (The three remaining measurements from the gyroscope and the accelerometer are ignored in this paper.) The IMU is directly connected to a I2C serial port of the mCore microcontroller.

Fig. 3 visualizes the software architecture of each mobile robot unit. The architecture comprises a sequence of layers. At the lowest programming level, the main Python process manages the data communication and information exchange. Concurrently, an Arduino based script defines the main process executed by the mCore microcontroller, handling the actuation section delivering the motors setpoint and receiving the sensors acquisition. Finally, the Simulink model, running as a standalone process, implements observation and control laws. The three layers communicate in a continuous TCP/IP communication network, exchanging data and commands.

B. Motors and sensors calibration

Before the IMU data can be used in an observer design the sensors need to be calibrated. Similarly, a calibration of

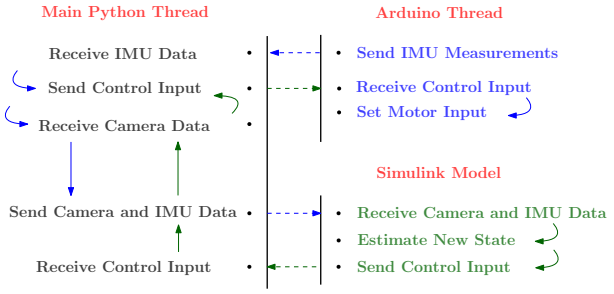

Fig. 3. Software architecture of each mobile robot unit.

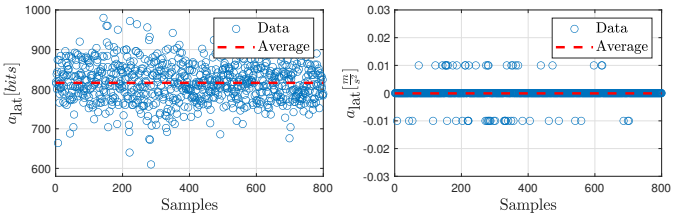

Fig. 4. Lateral accelerometer data a_{lat}. Raw measurements (left) and calibrated measurements (right). Similar plots are obtained for the forward acceleration a_{fw}.

the left and right motors is necessary to identify the static map between the setpoint request and the resulting wheels velocity.

1) IMU calibration: The calibration process is included in the main Arduino script of the mCore unit and is performed during the initialization task. For 5 s, the calibration function acquires the acceleration $a = [a_{\text{fw}}\ a_{\text{lat}}]^\top$ and angular velocity ω measurements, while the robot is at rest. In this process, nonzero measurements are identified as bias, also comprising the effect of gravity. During subsequent experiments, the average of the data acquired in the calibration process is used as a zero reference. Fig. 4 shows an example of the acceleration measurements acquired during the calibration process, including the internal analog to digital conversion performed by the embedded 16 bit analog-to-digital converter.

2) Motor calibration: The setpoint f_ℓ and f_r delivered to the left and right motors of each mobile robot unit range in the interval $[-250, 250]$. A series of experiments has been performed to estimate the characteristic curve associating the setpoint f_i, $i \in \{\ell, r\}$ to the actual left and right wheels velocities v_ℓ and v_r. In these experiments, one motor input is set to zero, while the other motor is driven by an increasing setpoint f_i, ramping up from -250 to 250, increasing by 10 units every 20 s, then the total experiment duration lasts $20\frac{500}{10} = 1000$ s, that is, roughly 15 minutes. Under the assumption of a uniform circular motion, during each steady-state velocity step, the forward velocity corresponds to $v_{\text{fw}} = \omega d$, where ω is the angular velocity measured by the IMU and d is the curvature radius of the circular motion, i.e., easily computed in this setting as the distance between the wheels. The obtained left "ℓ" and right "r" calibration curves are shown in Fig. 5 and exhibit a peculiar deadzone nonlinearity. They can be mathematically represented as

$$v_i(f_i) = \begin{cases} \alpha_i f_i - \beta_i \operatorname{sign}(f_i), & \text{if } |f_i| \geq 50, \\ 0, & \text{if } |f_i| \leq 50, \end{cases} \qquad i \in \{\ell, r\},$$

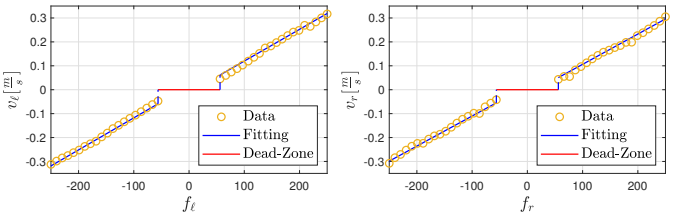

Fig. 5. Characteristic curves corresponding to the left and right motor obtained through the motor calibration.

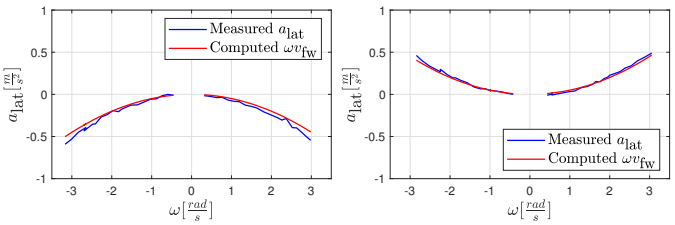

Fig. 6. Lateral acceleration for a uniform circular motion and relation between a_{lat} and ω, based on the measured data (for the left and the right wheels).

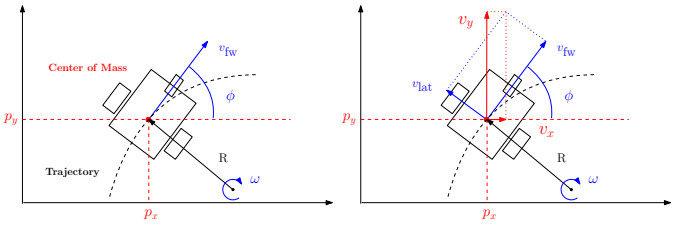

Fig. 7. Unicycle coordinates in (1): relation between velocities (v_x, v_y) in the global frame and velocities $(v_{\text{fw}}, v_{\text{lat}})$ in the body (right).

with the identified scalars $\alpha_r = \alpha_\ell = 0.0013$, $\beta_r = 0.0223$ and $\beta_\ell = 0.0182$.

During the experimented uniform circular motions, the lateral acceleration satisfies the relation $a_{\text{lat}} = \omega^2 d$, which is used in Fig. 6 to validate the assumption $v_{\text{fw}} = \omega d$. In particular, the expected relation $a_{\text{lat}} = \omega v_{\text{fw}} = \omega^2 d$ is compared with the measured accelerometer and gyroscope data and the result justifies the assumption.

III. Mobile Robot Dynamics & Observer Design

We present here two unicycle models and the corresponding estimators. The first (simpler) model only uses the camera measurements whereas the second, more sophisticated model, also uses the IMU measurements for the state estimation.

A. Standard Unicycle & Kalman Filter

The *unicycle*, a standard model describing mobile robot dynamics (see, e.g., [14, Ex. 4.3.16] and [10, Sec. 2.3.1]), well represents the dynamics of an mBot in terms of its position $(p_x, p_y) \in \mathbb{R}^2$ and orientation $\phi \in [0, 2\pi)$, namely

$$\dot{x} = f(x, u) + \mu = \begin{bmatrix} \nu_{\text{ref}} \cos(\phi) \\ \nu_{\text{ref}} \sin(\phi) \\ \omega_{\text{ref}} \end{bmatrix} + \mu, \tag{1}$$

where $x = [p_x\ p_y\ \phi]^\top$ is the unknown state of the robot and $\mu \in \mathbb{R}^3$ is an unknown process noise. As visualized in Fig. 7, input $u = \left[\begin{smallmatrix} \nu_{\text{ref}} \\ \omega_{\text{ref}} \end{smallmatrix}\right] \in \mathbb{R}^2$ comprises the *forward velocity* ν_{ref} and the *angular velocity* ω_{ref}, which are here approximated to coincide with the reference signals delivered to the actuators. More precisely, since the inputs applicable to the mBot are the left and right motor velocities $v = [v_\ell\ v_r]^\top$ (see Section II-B), we clarify the relation between v and u as

$$u = \tfrac{1}{2d} \left[\begin{smallmatrix} d & d \\ -1 & 1 \end{smallmatrix}\right] v \quad \Longleftrightarrow \quad v = \tfrac{1}{2} \left[\begin{smallmatrix} 2 & -d \\ 2 & d \end{smallmatrix}\right] u, \tag{2}$$

with d being again the distance between the two wheels. Since there is a one-to-one correspondence between u and v, we will

focus on u in the following. As outlined in Section II-A1, at discrete time instants $t_k \in \mathbb{R}_{\geq 0}$ the state measurements

$$y(t_k) = x(t_k - \Delta) + \eta(k) \tag{3}$$

are obtained from the camera units (which measure both position and orientation), where $\Delta = 0.4$ s is the processing time discussed in Section II-A1. The discrete input $\eta \in \mathbb{R}^3$ represents unknown measurement noise depending on the precision of the camera-based localization. Since multiple cameras could co-exist, and each robot could be in the field of view of multiple cameras, we consider that, for a specific robot, two successive camera measurements arrive at time instants satisfying $t_{k+1} - t_k \in [T_m, T_M] = [0.1, 1]$ seconds.

With these definitions, an estimate of the state $x(t)$ can be obtained through the continuous-discrete (or hybrid) extended Kalman filter [8, Sec. 13.2.2]. In particular, using the hybrid systems formalism of [7] we may define the overall state of the Kalman filter as $\Theta = (\hat{x}, \hat{\chi}, P, \tau) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^{3\times 3} \times \mathbb{R}_{\geq 0}$ and introduce the flow and jump sets, respectively $\mathcal{C} = \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^{3\times 3} \times [0, T_M]$, $\mathcal{D} = \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^{3\times 3} \times [T_m, T_M]$, where $\mathcal{D}$ encodes the property $t_{k+1} - t_k \in [T_m, T_M]$ commented above about the intersample time between two camera measurement instants. Then, the continuous-discrete extended Kalman filter for the mobile robot can be written as

$$\begin{cases} \dot{\hat{x}} = f(\hat{x}, u(t)) \\ \dot{\hat{\chi}} = f(\hat{\chi}, u(t-\Delta)) \\ \dot{P} = A(t)P + PA(t)^\top + Q \\ \dot{\tau} = 1, \end{cases} \quad \Theta \in \mathcal{C}, \tag{4}$$

$$\begin{cases} \hat{x}^+ = \hat{\chi} + (I-K)(y - \hat{\chi}) \\ \qquad +\varphi(t, t-\Delta, \hat{\chi} + (I-K)(y-\hat{\chi}), u_{[t-\Delta,t]}(\cdot)) \\ \hat{\chi}^+ = \hat{\chi} + (I-K)(y-\hat{\chi}) \\ P^+ = KPK^\top + (I-K)R(I-K)^\top \\ \tau^+ = 0, \end{cases} \quad \Theta \in \mathcal{D},$$

$$K = I - P(P+R)^{-1}, \quad A(t) = \tfrac{\partial f}{\partial x}(\hat{\chi}, u(t-\Delta)),$$

where K is the Kalman gain, $\varphi(t, t-\Delta, \xi_0)$ is the unique solution of the initial value problem: $\dot{\xi} = f(\xi, u(\tau))$, $\xi(t-\Delta) = \xi_0$, evaluated at time t, which is computed by running a numerical simulation at each measurement time t_k. The constant matrices R and Q denote the covariance matrices of the process disturbance μ and the measurement noise η.

Following the formalism of [7], the solution Θ of (4) is parametrized by the ordinary time t and the (discrete) count j of the number of measurements received so far. In particular, at hybrid time (t, j), state $\hat{x}(t, j)$ provides an estimate of the plant state $x(t)$, while state $\hat{\chi}(t, j)$ provides an estimate of state $x(t-\Delta)$, where Δ is the camera processing time. During the flow evolution (when no measurement is available), the two states run in parallel, driven by Δ-shifted versions of the input u. Instead, at each measurement time (t_j, j), the continuous-discrete Kalman update is first applied to $\hat{\chi}^+$ and then the selection $\hat{x}^+ = \hat{\chi}^+ + \varphi(t, t-\Delta, \hat{\chi}^+, u)$ propagates forward the initial value $\hat{\chi}^+$ via the prediction term φ. The Kalman gain K and the dynamics of P simplify significantly as compared to the standard results in [8, Sec. 13.2.2], because the measurement y in (3) provide the whole state.

The simple nonlinear model (1) and the ensuing observer (4) perhaps represents the most intuitive approach to our sampled-data estimation problem. However, it is hard to prove the convergence to zero of the estimation error $x - \hat{x}$, even in the ideal case of $\mu = 0$ and $\eta = 0$. This problem is resolved by the more sophisticated model derived in the next section.

B. Extended dynamics & observer design

We derive here a second model that allows using the continuous IMU measurements.[1] In this model, the estimation error becomes a nonlinear cascade of two linear systems (one for the position and one for the orientation) and we can rely on input-to-state stability properties [15] to deduce uniform convergence properties following the approach of [11].

For obtaining a cascade form, we separate the dynamics into the rotational and translational subcomponents. For the rotational part, we use the state $x_\phi = [\phi \ \omega \ d_\phi]^\top$, where ϕ denotes the robot orientation, ω denotes the angular velocity and d_ϕ models external (unmeasured) biases affecting the angular dynamics $\dot{\phi} = \omega + d_\phi$, satisfying $\dot{d}_\phi = 0$. Moreover, we model by $\dot{\omega} = -\lambda(\omega - \omega_{\text{ref}})$ taking into account a reasonable time constant $\frac{1}{\lambda}$ between the commanded angular velocity $\omega_{\text{ref}} = \frac{1}{d}(-v_\ell + v_r)$ introduced in (1), (2), and the actual angular velocity ω. Experimental tests led to the identified value $\lambda = 9.85$. Summarizing, we obtain the model

$$\dot{x}_\phi = A_\phi x_\phi + B_\phi \omega_{\text{ref}} + \mu_\phi, \qquad y^c_\phi = C^c_\phi x_\phi + \eta^c_\phi, \tag{5}$$

with μ_ϕ, η^c_ϕ being process disturbance and measurement noise,

$$A_\phi = \begin{bmatrix} 0 & 1 & 1 \\ 0 & -\lambda_\phi & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B_\phi = \begin{bmatrix} 0 \\ \lambda_\phi \\ 0 \end{bmatrix}, \quad (C^c_\phi)^\top = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix},$$

and where y^c_ϕ is the velocity output obtained from the gyro.

An appealing feature of the angular dynamics (5) is its linearity in the state x_ϕ. To obtain a similar linear relation for the translational dynamics, we first characterize the relation between the velocities (v_x, v_y) in the x- and y-directions of the global frame, and the forward and lateral velocities $(v_{\text{fw}}, v_{\text{lat}})$ in the body frame visualized in Fig. 7:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = R(\phi) \begin{bmatrix} v_{\text{fw}} \\ v_{\text{lat}} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix} \begin{bmatrix} v_{\text{fw}} \\ v_{\text{lat}} \end{bmatrix} \tag{6}$$

where $R(\hat{\phi})$ denotes the standard rotation matrix.

Differentiating both sides of (6), and recalling that a rotation matrix satisfies $\dot{R}(\phi) = JR(\phi)\dot{\phi} = R(\phi)J\dot{\phi}$, where $J = \left[\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right]$, we obtain

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = R(\phi) \left(J\dot{\phi} \begin{bmatrix} v_{\text{fw}} \\ 0 \end{bmatrix} + \begin{bmatrix} \dot{v}_{\text{fw}} \\ 0 \end{bmatrix} \right) = R(\phi) \begin{bmatrix} \dot{v}_{\text{fw}} \\ \dot{\phi}\, v_{\text{fw}} \end{bmatrix}, \tag{7}$$

and the zeros at the second entry of the vectors reported in the middle are motivated by the fact that the robot kinematics constrains the lateral velocity v_{lat} and its derivative $\dot{v}_{\text{lat}}$ to be both zero at all times.

[1] The IMU measurements are acquired significantly faster than the camera measurements and thus are denoted as "continuous" in this paper.

In the rightmost vector of (7) we recognize the forward acceleration $\dot{v}_{\text{fw}} = a_{\text{fw}}$ and the lateral acceleration $a_{\text{lat}} = \dot{\phi} v_{\text{fw}}$, this last relation being experimentally confirmed by the experiments in Fig. 6. Thus, with $a = [a_{\text{fw}}\ a_{\text{lat}}]^\top$, it holds that $\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = R(\phi)a$ and $\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \end{bmatrix}$. Defining the translational state $x_T = [p_x\ p_y\ v_x\ v_y]^\top$ we may then write the model

$$\dot{x}_T = A_T x + B_T(\phi)a + \mu_T, \tag{8}$$

where $\mu_T \in \mathbb{R}^4$ defines unmodeled process noise and with $\mathcal{O}, \mathcal{I} \in \mathbb{R}^{2\times 2}$ denoting the zero matrix and the identity matrix, respectively, A_T and $B_T(\phi)$ are defined as

$$A_T = \begin{bmatrix} \mathcal{O} & \mathcal{I} \\ \mathcal{O} & \mathcal{O} \end{bmatrix}, \quad B_T(\phi) = \begin{bmatrix} \mathcal{O} \\ R(\phi) \end{bmatrix}. \tag{9}$$

The overall model (5), (8) exhibits the desired cascaded structure, where ϕ is interpreted as an input of (8).

Without camera measurements, an asymptotic observer cannot be built for x_ϕ, x_T, due to the lack of alternative position measurements. Nevertheless, the continuous output y_ϕ^c in (5) can be used to design a Luenberger gain [16, §16.5] $L_\phi = [0\ \ell_2\ \ell_3]^\top$, $\ell_2, \ell_3 \in \mathbb{R}$ such that the closed-loop matrix $A_\phi^{\text{cl}} = A_\phi - L_\phi C_\phi^c$ has two eigenvalues in the left half-plane and one eigenvalue equal to zero (the one that is not observable for pair (A_ϕ, C_ϕ^c)). After this preliminary continuous-time output injection term, we may then consider the sporadic discrete-time camera-based outputs

$$y_\phi^d(t_k) = C_\phi^d x_\phi(t_k - \Delta) + \eta_\phi^d(k), \quad C_\phi^d = \begin{bmatrix} 1\ 0\ 0 \end{bmatrix}, \tag{10}$$

$$y_T^d(t_k) = C_T x_T(t_k - \Delta) + \eta_T^d(k), \quad C_T = \begin{bmatrix} 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0 \end{bmatrix}, \tag{11}$$

with η_ϕ^d and η_T^d being discrete-time measurement noise.

While we cannot characterize observability in terms of the continuous-time matrices A_ϕ^{cl} and A_T and the discrete-time output matrices C_ϕ^d and C_T, it is quite intuitive that the sampled-data measurements (10), (11) provide enough information for estimating the state of (5), (8). Rather than using a Kalman filter, we choose to follow the solution in [11] that provides an immediate proof of convergence to zero of the error dynamics. A similar result could be concluded from the results in [17] or [18], for example, when using a Kalman filter architecture.

An important advantage stemming from the linearity of (5) and (8) (regardless of the nonlinear coupling in the cascade) is that we do not need to introduce the double state $\hat{x}$ and $\hat{\chi}$ used in (4), nor do we need to run the online simulation required to compute the term φ at each sampled measurement. More specifically, proceeding as in (4), we define an overall state of the estimator as $\Gamma = [\hat{x}_T^\top\ \hat{x}_\phi^\top\ \tau]^\top$ and define the new flow and jump sets $\mathcal{C} = \mathbb{R}^7 \times [0, T_M]$ and $\mathcal{D} = \mathbb{R}^7 \times [T_m, T_M]$, which capture, just as in (4), the sporadic nature of the measurements. Then the observer flow dynamics corresponds to the models derived in (5), (8) augmented with the continuous output injection from the continuous gyroscope measurement y_ϕ^c:

$$\begin{cases} \dot{\hat{x}}_T = A_T \hat{x}_T + B_T(\hat{\phi})a \\ \dot{\hat{x}}_\phi = A_\phi^{\text{cl}} \hat{x}_\phi + L_\phi y_\phi^c + B_\phi \omega_{\text{ref}} \\ \dot{\tau} = 1, \end{cases} \qquad \Gamma \in \mathcal{C}. \tag{12a}$$

The ensuing flow dynamics of the orientation estimation error $\tilde{\phi} = \phi - \hat{\phi}$ is then determined from (5), (12a) as $\dot{\tilde{\phi}} = (A_\phi - L_\phi C_\phi^c)\tilde{\phi} + \mu_\phi - L_\phi \eta_\phi^c = A_\phi^{\text{cl}} \tilde{\phi} + \mu_\phi - L_\phi \eta_\phi^c$, which is fully linear. As a consequence the camera processing time delay can be taken into account as follows, with the notation $\hat{y}_\phi^d = C_\phi^d \hat{x}_\phi$:

$$\begin{aligned} y_\phi^d(t_k) - \hat{y}_\phi^d(t_k - \Delta) &= C_\phi^d (x_\phi(t_k - \Delta) - \hat{x}_\phi(t_k - \Delta)) + \eta_\phi^d(k) \\ &= C_\phi^d \mathrm{e}^{-A_\phi^{\text{cl}} \Delta} \tilde{x}_\phi(t_k) + \bar{\eta}_\phi^d(k) = \bar{C}_\phi^d \tilde{x}_\phi(t_k) + \bar{\eta}_\phi^d(k), \end{aligned}$$

where $\bar{C}_\phi^d = C_\phi^d \mathrm{e}^{-A_\phi^{\text{cl}} \Delta}$ is a transformed constant sampled-data output matrix and $\bar{\eta}_\phi^d$ is a transformed measurement noise depending on η_ϕ^d, μ_ϕ and η_ϕ^c. For the translational error dynamics $\tilde{x}_T = x_T - \hat{x}_T$, considering $\hat{\phi} = \phi$ (this is possible because we exploit the cascaded structure of the interconnection and the fact that B_T is uniformly bounded) parallel calculations lead to $y_T^d(t_k) - \hat{y}_T^d(t_k - \Delta) = \bar{C}_T \tilde{x}_T(t_k) + \bar{\eta}_T^d(k)$, where η_T^d is a transformed noise input and $\hat{y}_T^d = C_T \hat{x}_T$, $\bar{C}_T = C_T \mathrm{e}^{-A_T \Delta}$.

Based on the above description, we may follow twice the construction of [11, Algorithm 1] focusing on the virtual output matrices $\bar{C}_\phi^d$ and $\bar{C}_T$ to design, respectively, time-varying observer gains $K_T(\cdot) : [T_m, T_M] \to \mathbb{R}^{4\times 2}$ and $K_\phi(\cdot) : [T_m, T_M] \to \mathbb{R}^{3\times 1}$ obtained from [11, Eq. (17)], providing the observer jump dynamics

$$\begin{cases} \hat{x}_T^+ = \hat{x}_T + K_T(\tau)(y_T - \hat{y}_T^d(t - \Delta)) \\ \hat{x}_\phi^+ = \hat{x}_\phi + K_\phi(\tau)(y_\phi^d - \hat{y}_\phi^d(t - \Delta)) \\ \tau^+ = 0, \end{cases} \qquad \Gamma \in \mathcal{D}, \tag{12b}$$

which evidently provides an easier implementation, only requiring memory variables to store the past values of the predicted outputs $\hat{y}_T^d$ and $\hat{y}_\phi^d$. In addition, the following result can be proven using the derivations reported above and the main result of [11]. The proof is omitted due to space constraints.

Theorem 1: In the noise free case, the continuous-discrete observer (12) provides asymptotic convergence to zero of the estimation error $\begin{bmatrix} x_T - \hat{x}_T \\ x_\phi - \hat{x}_\phi \end{bmatrix}$.

IV. Experimental results

We assess here the performance of the observers introduced in Section III, through experiments on a robot performing circular motions. In particular, Fig. 8 shows experiments using the Kalman filter described in Section III-A while Fig. 9 reports on the observer of Section III-B. For the observer (4), Fig. 8 shows the first two components of the state estimates $\hat{x}$ together with the sampled camera measurements y in (3), represented in the (x, y) plane. On the top left, the first five seconds of the experiment are shown and on the top right the time window $t \in [14, 18]$ seconds is visualized. The time evolution of the estimated state $\hat{x}$, together with the camera measurements are shown in the bottom figures. The camera measurements are shifted by Δ so that they are correctly aligned with the estimates. Similarly, for the observer (12), the top Fig. 9 shows the position estimates $\hat{x}_T$, together with the camera measurements y_T^d in (11) at the beginning (left) and at the end of the experiment. The lower plots show the time histories of the same variables with the measurements shifted by Δ, to correctly align them with $\hat{x}_T$ and $\hat{x}_\phi$.

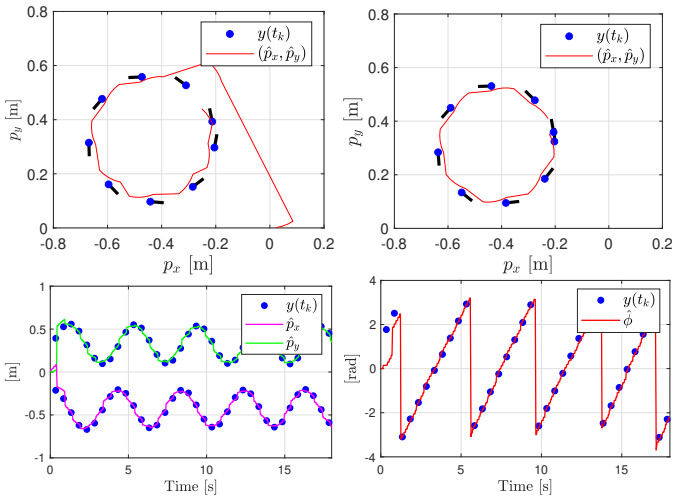

Fig. 8. The top figures show the estimates $\hat{x}$ of observer (4), together with the camera measurements y in (3), at the beginning of the experiment (left) and the end of the experiment (right). The bottom figures show the time histories of the same position and orientation estimates.

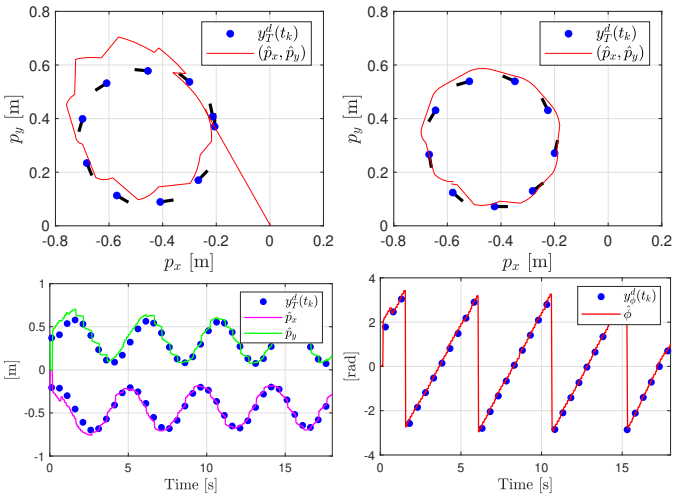

Fig. 9. The top figures show the estimated state $\hat{x}_T$ obtained from observer (12), together with the camera measurements y_T^d in (11), at the beginning of the experiment (left) and the end of the experiment (right). The bottom figures show the time histories of the position and orientation estimates.

In the experiments, the circular motion is obtained through a constant input u corresponding to $\nu_{\text{ref}} = 0.225[\frac{\text{m}}{\text{s}}]$ and $\omega_{\text{ref}} = 1.5[\frac{rad}{s}]$. The ordinary differential equations involved in the observer dynamics are solved numerically using an explicit Euler method with a sampling time of 0.025 s. From the IMU measurements, the average values over time windows of length 0.1s are used. The camera measurements are received with an intersample time of about 0.5 s.

From the experimental results we observe that, after an initial transient, both observers achieve comparable results and seem to return a reliable estimate of the robot state. We point out that the camera frames are subject to measurement errors, which cannot be quantified in the current setup, and thus they are not capturing the true state of the robot. Hence, a performance evaluation using the mismatch between the camera measurements and state estimates is not leading to a useful unit describing the performance of an observer. Performing the experiments using a high precision ground truth motion capture system will allow us to evaluate both, the accuracy of the camera acquisition process and the performance of the observers, and is left for future work. The performance analysis will be combined with a tuning process of the parameters involved in the observer design.

V. Conclusion

We developed a modular hardware/software system for mobile robot localization, combining camera modules, mBots and onboard IMU sensors. For this setup, we have developed a continuous-discrete state observer combining continuous IMU measurements with sporadic non-periodic camera measurements. In future work the performance of the state observer will be addressed in detail using a MoCap system, the IMU measurements frequency will be increased, the model will be refined and the setting will be validated for more sophisticated closed-loop trajectories.

References

[1] M. Hampson. Drone delivers human kidney: The organ was flown several kilometers by a drone without incurring damage. *IEEE Spectrum*, 56(1):7–9, 2019.

[2] E. Guizzo. Three engineers, hundreds of robots, one warehouse. *IEEE Spectrum*, 45(7):26–34, 2008.

[3] B. Barshan and H. F. Durrant-Whyte. Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11(3):328–342, 1995.

[4] F. M. Mirzaei and S. I. Roumeliotis. A Kalman filter-based algorithm for IMU-camera calibration: Observability analysis and performance evaluation. *IEEE Transactions on Robotics*, 24(5):1143–1156, 2008.

[5] L. Li, Y. Liu, T. Jiang, K. Wang, and M. Fang. Adaptive trajectory tracking of nonholonomic mobile robots using vision-based position and velocity estimation. *IEEE Transactions on Cybernetics*, 48(2):571–582, 2018.

[6] A. Liu, W. Zhang, M. Z. Q. Chen, and L. Yu. Moving horizon estimation for mobile robots with multirate sampling. *IEEE Transactions on Industrial Electronics*, 64(2):1457–1467, 2017.

[7] R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid dynamical systems: modeling stability, and robustness*. Princeton University Press, 2012.

[8] D. Simon. *Optimal State Estimation: Kalman, H_∞, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[9] F. L . Lewis, L. Xie, and D. Popa. *Optimal and Robust Rstimation: With an Introduction to Stochastic Control Theory*. CRC Press, 2008.

[10] S. G. Tzafestas. *Introduction to Mobile Robot Control*. Elsevier, 2013.

[11] A. Sferlazza, S. Tarbouriech, and L. Zaccarian. Time-varying sampled-data observer with asynchronous measurements. *IEEE Transactions on Automatic Control*, 64(2):869–876, 2019.

[12] F. Alonge, F. D'Ippolito, G. Garraffa, and A. Sferlazza. A hybrid observer for localization of mobile vehicles with asynchronous measurements. *Asian Journal of Control*, 21(4):1506–1521, 2019.

[13] G. M. Vinco. Coordination of mobile robots: architecture for hybrid observer and controller design. Master's thesis, Univ. of Trento, 2020.

[14] E. D. Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems*. Springer Science & Business Media, 1998.

[15] E. D. Sontag. Remarks on stabilization and input-to-state stability. In *Proc. of the 28th IEEE Conference on Decision and Control*, 1989.

[16] J. P. Hespanha. *Linear Systems Theory*. Princeton Univ. Press, 2018.

[17] B. D. O. Anderson and J. B. Moore. Detectability and stabilizability of time-varying discrete-time linear systems. *SIAM Journal on Control and Optimization*, 19(1):20–32, 1981.

[18] A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.