

# Pronto: a Multi-Sensor State Estimator for Legged Robots in Real World Scenarios

Marco Camurri<sup>1,\*</sup>, Milad Ramezani<sup>1</sup>, Simona Nobili<sup>1,2</sup> and Maurice Fallon<sup>1</sup>

<sup>1</sup>*Dynamic Robot Systems, Oxford Robotics Institute, Department of Engineering Science, University of Oxford, Oxford, UK*

<sup>2</sup>*University of Edinburgh, Edinburgh, UK*

Correspondence\*:

M. Camurri  
mcamurri@robots.ox.ac.uk

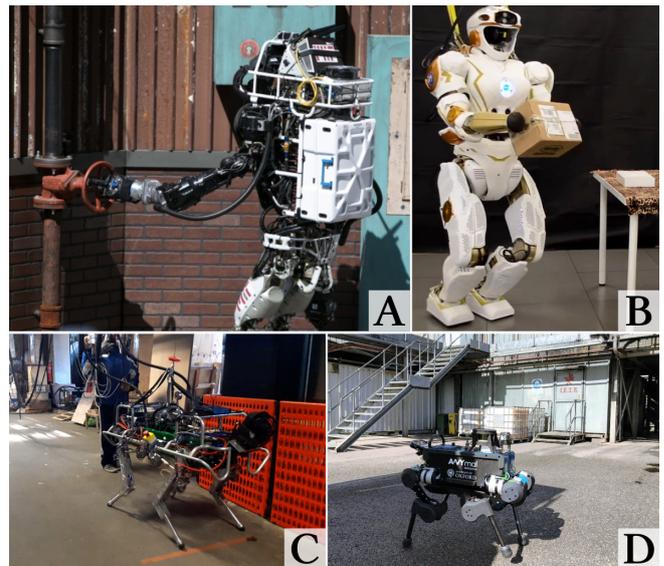
## ABSTRACT

In this paper we present a modular and flexible state estimation framework for legged robots operating in real world scenarios, where environmental conditions such as occlusions, low light, rough terrain and dynamic obstacles can severely impair estimation performance. At the core of the proposed estimation system, called Pronto, is an Extended Kalman Filter (EKF) which fuses IMU and Leg Odometry sensing for pose and velocity estimation. We also show how Pronto can integrate pose corrections from visual and LIDAR and odometry to correct pose drift in a loosely coupled manner. This allows it to have a real-time proprioceptive estimation thread running at high frequency (250–1000 Hz) for use in the control loop while taking advantage of occasional (and often delayed) low frequency (1–15 Hz) updates from exteroceptive sources, such as cameras and LIDARs. To demonstrate the robustness and versatility of the approach, we have tested it on a variety of legged platforms, including two humanoid robots (the Boston Dynamics Atlas and NASA Valkyrie) and two dynamic quadruped robots (IIT HyQ and ANYbotics ANYmal) for more than 2 h of total runtime and 1.37 km of distance traveled. The tests were conducted in a number of different field scenarios which the conditions described above. The algorithms presented in this paper are made available to the research community as open-source ROS packages.

## 1 INTRODUCTION

Legged robotics is rapidly transitioning from research laboratories into the real world, as demonstrated by the recent introduction of several commercial quadruped platforms.

To be truly useful, legged robots must be able to reliably and rapidly navigate across rough terrain



**Figure 1.** The Atlas (A) and Valkyrie (B) humanoid robots; The HyQ (C) and ANYmal (D) dynamic quadruped robots. Sources: MIT, University of Edinburgh, Istituto Italiano di Tecnologia (IIT), University of Oxford.

and be stable in the presence of disturbances such as slips or pushes. They must also be able to perceive and manipulate the environment whilst avoiding collisions with obstacles and people.

None of these tasks can be accomplished without the ability to accurately and robustly estimate the pose and velocity of the robot (i.e., its state) in real-time using only onboard sensors and computers. The robot's state is used to plan and track body trajectories, to balance and recover from external disturbances, as well as to map the environment and navigate through it.

To achieve a satisfactory level of accuracy, proprioceptive and exteroceptive sensor fusion is necessary, exposing the problem of synchronization and latency between the different signals coming from each sensor.

Migrating from the controlled environment of a laboratory to the real operating conditions of industrial applications (e.g., oil rig platform inspection or mine exploration) makes the task even more challenging, as it requires extra effort to robustify the estimation algorithm against unknown situations and long periods of continuous operation without human intervention.

In this paper, we demonstrate how inertial, kinematic, stereo vision and LIDAR sensing can be combined to produce a low-latency and high-frequency state estimate which can be directly used to control state-of-the-art humanoids and dynamic quadrupeds. In turn, this estimate can be used to build accurate maps of the robot’s immediate environment and to enable navigational autonomy and manipulation.

Compared with prior research, this contribution is the first to provide an open source implementation of a fully integrated state estimation system performing sensor fusion of IMU, kinematics, stereo vision and LIDAR on four different legged platforms: the NASA Valkyrie and Boston Dynamics Atlas humanoids; and the IIT HyQ and ANYbotics ANYmal quadrupeds (Fig. 1). Another key achievement in comparison with the state-of-the-art is the demonstration of the algorithm in closed loop with the controller.

## 1.1 Contribution

This paper combines previous works focused on the individual platforms including state estimation on Atlas in Fallon et al. (2014); an extension to the HyQ quadruped and incorporation of vision in Nobili et al. (2017a); and further evaluation of LIDAR localization in Nobili et al. (2017b).

The paper provides a complete and coherent overview of the method with *a*) a comprehensive and updated survey on state estimation methods for legged robots; *b*) additional experimental results on the ANYmal quadruped platform; *c*) a more detailed description of the overall estimation method and architecture.

Furthermore, we release the Pronto state estimator, the FOVIS Visual Odometry, and the AICP LIDAR odometry modules as open-source ROS packages for the research community.

## 2 RELATED WORK

The literature on state estimation for legged robots can be classified according to several criteria: the type of sensors used (proprioceptive, exteroceptive, or both); output frequency (at control rate, e.g.

400 Hz, or camera/LIDAR rate, e.g. 10 Hz); state definition (pose, velocity, joint states, contact points, etc.); presence of loop closures (odometry vs. SLAM); the degree of marginalization of past states (from filtering to full batch optimization). Finally, if there is fusion of proprioceptive and exteroceptive signals, this can be performed in a *loosely* or *tightly* coupled manner.

In this section we divide the related work in three main categories: *proprioceptive state estimation*, which includes filtering methods to fuse only the high frequency signals such as IMU and kinematics; *multi-sensor filtering*, which cover filtering methods with proprioceptive and exteroceptive sensor fusion; *multi-sensor smoothing*, which typically involve fusion of Visual Odometry (VO), IMU and kinematics in a tightly coupled manner within probabilistic graphical models frameworks such as factor graphs.

### 2.1 Proprioceptive State Estimation

Nearly all modern legged robots are equipped with IMUs, encoders and force/torque sensors. Since these devices provide low dimensional signals at high frequencies (250–1000 Hz), they are the first to be fused for a smooth (although drifting) state estimate, useful for control purposes. Since real-time safety is paramount for controllers, most methods are based on algorithms such as the Kalman filter (Section 2.1.1) or lightweight optimization methods (Section 2.1.2).

#### 2.1.1 Kalman Filtering

Bloesch et al. (2012) were the first to propose an EKF-based state estimator method that did not depend on a specific type of gait or number of legs. The filter used IMU signals (linear acceleration and angular velocity) as inputs to be integrated for the process model. The state included pose and velocity of the robot, as well as IMU biases and feet contact locations. In this way, they could define leg odometry measurements from forward kinematics of the feet in stable contact with the ground. Their work was implemented on the StarLETH robot and tested on short indoor experiments. Shortly after, Rotella et al. (2014) adapted the same method to humanoid platforms by including the ankle joint and the feet orientations in the state vector.

An important aspect of humanoid robot state estimation is the distance between the Center of Mass (CoM) and the feet, which is larger than quadruped platforms. In humanoids, the flexibility of the links is therefore not negligible and can lead to falls when the CoM is incorrectly estimated to be inside the relatively small support polygon

given by the robot's footprints. Xinjilefu et al. (2015) explicitly estimated the CoM offset using an inverted pendulum model to infer modeling error and/or unexpected external forces. In contrast, the approach of Koolen et al. (2016) modeled the elasticity of their robot's leg joints to better distribute error.

The above methods integrated the kinematics as position constraints. An alternative approach is to use differential kinematics in addition to forward kinematics to create linear velocity measurements, which are then integrated within the filter to get consistent position estimates. Bloesch et al. (2013) applied this approach again on the StarLETH quadruped robot. Since angular velocity from the IMU appeared on both the inertial process model and the measurement update, the authors proposed the Unscented Kalman Filter (UKF) instead of an EKF to better handle the correlation between the joint and gyroscope noises.

Fallon et al. (2014) used the same elasticity model of Koolen et al. (2016) and integrated leg odometry as velocity measurements on the Atlas robot. Since the EKF models the measurements as Gaussian, nonlinearities such as slippages or impacts are not captured by the filter noise model. Therefore, special care was taken to ignore invalid contacts by classifying the output from the contact sensor in the feet.

When feet sensors are unavailable, the contact feet are detected by thresholding the Ground Reaction Forces (GRF), which are estimated from the joint torques. Camurri et al. (2017) proposed a method that evaluates GRF discontinuities to discard invalid leg odometry velocity measurements on the HyQ quadruped robot. To better detect the feet in contact, they also proposed a logistic regression method to learn the optimal GRF threshold on different gaits. A different approach, based on Hidden Markov Model (HMM), was adopted by Jenelten et al. (2019) for slip recovery on the ANYmal robot. In this case, the probability of contact for each leg was determined from dynamics and differential kinematics.

### 2.1.2 Optimization

Kalman-based filtering have been preferred over more sophisticated methods because of its simplicity and low computational expense. However, recent technological progress have made optimization-based methods feasible to use. These methods can overcome some limitations such as the need to define a process model, even when unfit for the application. Indeed, the widely adopted EKF inertial process model approximates the robot to a

ballistic missile, while optimization methods could incorporate the floating base dynamics equations of motion instead.

Xinjilefu et al. (2014) formulated the state estimation of the Atlas robot as a Quadratic Programming (QP) problem. The cost function was defined as the weighted sum of two quadratic terms: the modeling error and the measurement error, where the former is derived from the floating base dynamics equation of motion while the latter is derived from encoders, force/torque sensors and IMU measurements. The optimization variable was composed by: the generalized (i.e., joint and base link) velocities, generalized forces and the modeling error itself. Note that the base link pose was not part of the state and was estimated separately with an EKF. Tests on the Atlas robot have shown significant improvements in the behavior of the feedback controller with this estimation method.

A more unified optimization-based solution was proposed by Bloesch et al. (2018). In their approach, they eliminated the process model and made each measurement dependent on both the current and the previous state of the system. Intuitively, this is similar to an incremental smoothing method with a window of size two. The approach was able to integrate the dynamic equations of motion to estimate the linear and angular acceleration of the robot body in addition to what sensed by the IMU, providing extra redundancy. If a process model is available, it could still be incorporated as a pseudo measurement, allowing the form of an EKF to be retained if required.

## 2.2 Multisensor Filtering

Chilian et al. (2011) were among the first to discuss stereo, inertial and kinematic fusion on a legged robot. They used a six-legged crawling robot measuring just 35 cm across – yet combining all the required sensing on board.

Similarly, Ahn et al. (2012) addressed the 3D pose estimation of the humanoid robot Roboray, using an EKF-based SLAM technique. Their motion estimation pipeline contains a visual-inertial-kinematic odometry module and visual SLAM module. The kinematic and visual odometry are used to update the IMU measurements within an EKF filter. These constitute the input of the visual SLAM algorithm which performed loop closures and decrease the drift.

Hornung et al. (2010) applied Monte Carlo localization (MCL), a Bayes filtering approach which recursively estimates the posterior, to estimate the 6 DoF pose of the Nao humanoid robot.

By fusing the measurements of a 2D LIDAR with a motion model, they estimated the pose of the robot’s torso including while climbing a miniature staircase.

Ma et al. (2016) proposed an error-state Kalman filter fusing a tactical grade inertial measurement unit with stereo visual odometry to produce a pose estimate for navigation tasks, such as path planning. The robot’s kinematic sensing was only used when visual odometry failed. Their approach was focused on pose estimation and was not used within the robot’s closed loop controller. Their extensive evaluation (over thousands of meters) achieved 1 % error per distance travelled.

In contrast to the above mentioned methods, we aim to estimate both pose and velocity of the robot with multisensor fusion, and use this estimate online inside the control loop. This is motivated by the fact that, for highly dynamic motions, the drift rate of proprioceptive estimators is unacceptable and requires integration of other exteroceptive signals.

The estimator used in this work is based on a loosely-coupled EKF, an approach that has been previously applied to Micro Aerial Vehicles (MAVs), e.g., Lynen et al. (2013) Shen et al. (2014).

### 2.3 Multisensor Smoothing

Smoothing methods are well established in the MAVs community for tightly coupled visual-inertial navigation, partly due to the relatively low complexity of these machines (e.g., fewer degrees of freedom). The main advantage of smoothing is the ability to jointly use all (or part) of the past history of measurements to reduce the uncertainty around the full robot’s trajectory.

In the recent years, promising works have been released to apply these techniques to legged machines. Hartley et al. (2018b) proposed the first attempt to fuse leg odometry and IMU in a factor graph on the Cassie bipedal robot. They extended the state with the feet contact locations and defined two new factors to incorporate forward kinematics and impose a zero velocity constraint on the contact points of a foot. These were then combined with the preintegrated IMU factor from Forster et al. (2017a). Hartley et al. (2018a) extended this work to include additional pose measurements from the SVO Visual Odometry system (Forster et al. (2017b)). Both works were demonstrated on Cassie in controlled environments for a short period of time (<5 min).

Wisth et al. (2019) proposed a tightly coupled visual-inertial-legged system based on the iSAM2 solver running on the ANYmal robot. The method extracts Kanade-Lucas features from the stereo

camera on a RealSense D435 camera and optimizes them as the landmarks in the graph. Leg odometry was integrated as relative pose factors obtained from the internal state estimator running on the robot, Bloesch et al. (2018). The method was demonstrated on extensive outdoor experiments in urban and industrial scenarios where dynamic occludants and textureless areas were present in the scene.

All the above works were based on the assumption of a stationary point of contact. This assumption is violated every time there are slippages or deformations of the leg and/or the ground. Contact detection methods can help to reject sporadic slippage or deformation events. However, when these occur regularly, they need to be modelled.

Wisth et al. (2019) proposed a factor graph method that models contact nonlinearities as bias term of the linear velocity measurements from leg odometry. This can reduce the inconsistency between leg and visual odometry and provide a more robust pose and velocity estimate.

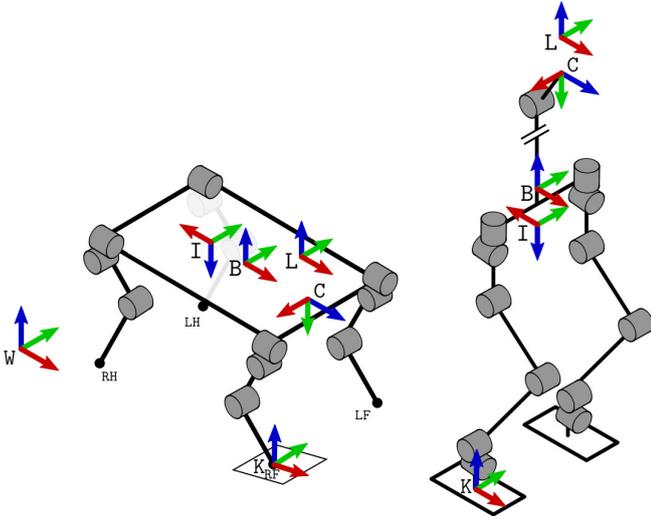
## 3 PROBLEM STATEMENT

We wish to track the pose and velocity of an articulated floating base robot with two or more legs and equipped with an onboard IMU, joint sensing of position and torque, cameras, and LIDARs. In this paper, we will focus on the Atlas and Valkyrie 28-DoF humanoids and on the HyQ and ANYmal 12-DoF quadrupeds. The robots of the same type share the same kinematic tree, with differences only in the links length and sensor locations.

### 3.1 Frames and Definitions

#### 3.1.1 Coordinate Frames

In Figure 2 we illustrate the reference frames relevant to our estimation problem. The inertial frame  $W$  and the base frame  $B$  are rigidly attached to the ground and the robot’s floating base, respectively. The frames located at the sensor origins are also rigidly attached to the floating base, namely: the camera optical frame  $C$ , the IMU frame  $I$ , the LIDAR frame  $L$ . The relative locations of these frames are known by design or can be retrieved with calibration procedures such as the ones described in Furgale et al. (2013); Reinke et al. (2019). One or more temporal contact frames  $K$  are created when a foot comes into contact with the ground.



**Figure 2.** Reference frame conventions for typical quadruped and humanoid robots (with a simplified upper torso structure). The world frame  $W$  is fixed to earth, while the base frame  $B$ , the camera’s optical frame  $C$ , and the IMU frame,  $I$  are rigidly attached to the robot’s chassis or head. When a foot touches the ground (e.g., the Right Front, RF), a contact frame  $K$  (perpendicular to the ground) is defined.

### 3.1.2 Notation

In the remainder of the paper, we adopt the following conventions: the robot position  $\mathbf{p} = {}_W\mathbf{p}_{WB} \in \mathbb{R}^3$  and orientation  $\mathbf{R} = \mathbf{R}_{WB} \in SO(3)$  are from world to base and expressed in world coordinates; the robot velocities  $\mathbf{v} = {}_B\mathbf{v}_{WB}$ ,  $\boldsymbol{\omega} = {}_B\boldsymbol{\omega}_{WB} \in \mathbb{R}^3$  are from world to base expressed in base coordinates; the IMU biases  ${}_I\mathbf{b}^a$ ,  ${}_I\mathbf{b}^\omega \in \mathbb{R}^3$  are expressed in IMU coordinates. A time dependent vector quantity  $\mathbf{a}$  computed at time  $t_k$  is shortened as  $\mathbf{a}_k = \mathbf{a}(t_k)$ .

### 3.2 State Definition

The robot state is defined as the vector combining position, orientation, linear velocity, and IMU biases. The angular velocity does not appear as it is assumed to be directly measured by the IMU once properly bias compensated. The state at time  $t_k$  is:

$$\mathbf{x}_k = [\mathbf{p}_k \quad \mathbf{R}_k \quad \mathbf{v}_k \quad \mathbf{b}_k^a \quad \mathbf{b}_k^\omega]^\top \quad (1)$$

The orientation uncertainty is tracked by the exponential coordinates of the perturbation rotation vector, as described in Bry et al. (2012).

### 3.3 Requirements

To effectively track base and feet trajectories, the state estimate should have negligible drift at least

over the course of one planning cycle. Modern footstep planners typically replan every 1–5 s. Low latency velocity estimates (including transduction and data transmission) are also fundamental for the feedback loop of a controller.

Low drift or drift-free state estimates are also required for navigation tasks (such as mapping and global path planning) as basic building blocks for many autonomous systems.

With these considerations in mind, we define the following requirements for a state estimator designed to run on legged robots in field operations:

- low pose drift on short range (e.g., 10 m);
- reliability in real semi-structured environments (i.e., does not diverge);
- signal smoothness for safe use in a control loop.

## 4 METHOD DESCRIPTION

Our approach adapts the core EKF filter described in Bry et al. (2012), with velocity corrections added by Fallon et al. (2014) for humanoid kinematics and then extended to quadruped kinematics in Camurri et al. (2017) (see Section 4.2). Additional pose corrections from visual odometry and LIDAR registration are described in Sections 4.4 and 4.5.

The goal of the EKF is to estimate the mean  $\boldsymbol{\mu}$  and covariance  $\boldsymbol{\Sigma}$  of the Gaussian distribution over the state,  $\mathbf{x}_k$ , given: the previous state  $\mathbf{x}_{k-1}$ , the current control input  $\mathbf{u}_k$ , and the current measurement  $\mathbf{z}_k$ . The state is first predicted using the nonlinear discrete transition function  $\mathbf{f}(\cdot)$  and then corrected by the observation function  $\mathbf{h}(\cdot)$ . Both functions are corrupted by zero-mean Gaussian process noise  $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$  and measurement noise  $\boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_k)$ :

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_k, \mathbf{w}_k) \quad (2)$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \boldsymbol{\eta}_k) \quad (3)$$

The mean and covariance are propagated in the standard manner:

$$\boldsymbol{\mu}_k^- = \mathbf{f}(\boldsymbol{\mu}_{k-1}, \mathbf{u}_k, \mathbf{0}) \quad (4)$$

$$\boldsymbol{\Sigma}_k^- = \mathbf{A}_{k-1} \boldsymbol{\Sigma}_{k-1} \mathbf{A}_{k-1}^\top + \mathbf{W}_{k-1} \mathbf{Q} \mathbf{W}_{k-1}^\top \quad (5)$$

where the minus superscript indicates that the quantity is evaluated before the measurement update takes place. For details on the derivation of the partial derivatives of the transition function,  $\mathbf{A}$  and  $\mathbf{W}$ , please refer to Bry et al. (2012).

The measurements are also integrated in a standard EKF manner. For instance, a velocity

measurement  $\tilde{\mathbf{v}}_k \in \mathbb{R}^3$  with covariance matrix  $\mathbf{P}_k^v \in \mathbb{R}^{3 \times 3}$  would be integrated as follows:

$$\mathbf{z}_k = \tilde{\mathbf{v}}_k \quad (6)$$

$$\mathbf{K}_k = \Sigma_k^- \mathbf{H}^\top (\mathbf{H} \Sigma_k^- \mathbf{H}^\top + \mathbf{P}_k^v)^{-1} \quad (7)$$

$$\boldsymbol{\mu}_k = \boldsymbol{\mu}_k^- + \mathbf{K}_k (\mathbf{z}_k - \mathbf{H} \boldsymbol{\mu}_k^-) \quad (8)$$

$$\Sigma_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \Sigma_k^- \quad (9)$$

where:  $\mathbf{K}_k \in \mathbb{R}^{15 \times 3}$  is the Kalman gain and  $\mathbf{H} \in \mathbb{R}^{3 \times 15}$  is the Jacobian of the observation function, which in the specific case above acts as a selector matrix for the linear velocity substate.

#### 4.1 Inertial Process Model

The acceleration (in the presence of gravity) and angular velocity are sensed by the IMU at high frequencies in the range 0.4–1 kHz. These are affected by bias and zero-mean Gaussian noise:

$${}_{\mathbf{I}}\tilde{\boldsymbol{\omega}}_{\mathbf{WI}} = {}_{\mathbf{I}}\boldsymbol{\omega}_{\mathbf{WI}} + \mathbf{b}^\omega + \boldsymbol{\eta}_\omega \quad (10)$$

$${}_{\mathbf{I}}\tilde{\mathbf{a}}_{\mathbf{WI}} = {}_{\mathbf{I}}\mathbf{a}_{\mathbf{WI}} + \mathbf{b}^a + \boldsymbol{\eta}_a \quad (11)$$

These quantities are transformed into the base frame and used as inputs to the process model:

$$\mathbf{u} = \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathbf{IB}} ({}_{\mathbf{I}}\tilde{\boldsymbol{\omega}}_{\mathbf{WI}} - \mathbf{b}_\omega - \boldsymbol{\eta}_\omega) \\ \mathbf{R}_{\mathbf{IB}} ({}_{\mathbf{I}}\tilde{\mathbf{a}}_{\mathbf{WI}} - \mathbf{b}_a - \boldsymbol{\eta}_a) \end{bmatrix} \quad (12)$$

where  $\mathbf{R}_{\mathbf{IB}}$  is the rotational part of the rigid transform between the IMU and base frames. Note that we ignore the effects of angular acceleration and centripetal force (see Diebel (2006)) and assume the IMU is close enough to the robot's base to make them negligible.

The process equations are:

$$\dot{\mathbf{p}} = \mathbf{R}\mathbf{v} \quad (13)$$

$$\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\omega}^\wedge \quad (14)$$

$$\dot{\mathbf{v}} = -\boldsymbol{\omega}^\wedge \mathbf{v} + \mathbf{R}^\top \mathbf{g} + \mathbf{a} \quad (15)$$

$$\dot{\mathbf{b}}_a = \boldsymbol{\eta}_b^a \quad (16)$$

$$\dot{\mathbf{b}}_\omega = \boldsymbol{\eta}_b^\omega \quad (17)$$

where  $\boldsymbol{\eta}_b^\omega$ ,  $\boldsymbol{\eta}_b^a$  are bias random walk noises.

Given Equations 13–17, we can predict the mean of the state  $\mathbf{x}_k$  by simple integration over the period

$$\Delta t = t_k - t_{k-1}:$$

$$\mathbf{u}_k = \begin{bmatrix} \boldsymbol{\omega}_k \\ \mathbf{a}_k \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{\mathbf{IB}} (\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_{k-1}^\omega) \\ \mathbf{R}_{\mathbf{IB}} (\tilde{\mathbf{a}}_k - \mathbf{b}_{k-1}^a) \end{bmatrix} \quad (18)$$

$$\boldsymbol{\mu}_k^- = \mathbf{f}(\boldsymbol{\mu}_{k-1}, \mathbf{u}_k, \mathbf{0}) = \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{v}_{k-1} \\ \mathbf{0} \\ \mathbf{b}_{k-1}^a \\ \mathbf{b}_{k-1}^\omega \end{bmatrix} +$$

$$\begin{bmatrix} \mathbf{v}_{k-1} \Delta t \\ (-\tilde{\boldsymbol{\omega}}_k^\wedge \mathbf{v}_{k-1} + (\mathbf{R}_{k-1})^\top \mathbf{g} + \mathbf{a}_k) \Delta t \\ \mathbf{R}_{k-1} \exp(\boldsymbol{\omega}_k^\wedge \Delta t) \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \quad (19)$$

Note that the attitude is integrated separately using the exponential map between the Lie group of rotations and its Lie algebra at the identity (see Forster et al. (2017a)).

The prior covariance  $\Sigma_k^-$  is also computed by Euler integration of the partial derivatives of the process equation, as detailed in Bry et al. (2015).

Having propagated the filter, measurements from other sensors can be used to correct the state vector. In the following sections, we derive measurements and their covariance matrix from leg, visual and LIDAR odometry.

#### 4.2 Leg Odometry

Leg odometry estimates the incremental motion of the floating base of a legged robot from the forward kinematics of the legs in stable contact with the ground. This measurement can be formulated as either a relative pose or a velocity measurement. In our system, we formulate linear velocity measurements.

In the following sections, we derive this measurement specifically for humanoids and quadrupeds.

##### 4.2.1 Humanoids

We adopt the contact classification and velocity measurement strategies from Fallon et al. (2014).

##### 4.2.1.1 Contact Classification

Humanoid robots are typically equipped with force/torque sensors at the feet, from which the contact state can be inferred by thresholding the measured normal force. Torsional friction is assumed to be high enough for there to be no foot rotation.

We use a Schmitt trigger to classify contact forces sensed by the robot’s 3-axis foot force-torque sensors and to detect how likely a foot is to be in contact. For simplicity, only one foot is detected as in contact during a double support phase and used a simple state machine to decides which foot is more reliable.

We also classify other events in the gait cycle, such as striking contact (as a 20–30 N positive and increasing discontinuity lasting more than 5 ms) and breaking contact (negative force discontinuity below a threshold) Because these events create unrealistic measurements, the EKF integrates them with higher measurement covariance.

Finally, we found that in some cases it is necessary to use the state of the controller to decide which contact points were in stable contact. For example, when climbing stairs the toe of the trailing foot pushes the robot upward but is not in stationary contact (a “toe off” event). In that case, we use information from the controller to assign the leading foot to be the primary fixed foot.

#### 4.2.1.2 Measurements

Once the primary fixed foot is established, a velocity measurement is created by differentiation of the base position across the interval  $\Delta t = t_k - t_{k-1}$ . The foot contact locations at times  $t_{k-1}$  and  $t_k$  are defined as the composition of the base position in world coordinates and the foot position in base coordinates:

$${}^w\mathbf{p}_{\text{WK}}(t_{k-1}) = \mathbf{p}_{k-1} + \mathbf{R}_{k-1}\text{fk}(\mathbf{q}_{k-1}) \quad (20)$$

$${}^w\mathbf{p}_{\text{WK}}(t_k) = \mathbf{p}_k + \mathbf{R}_k\text{fk}(\mathbf{q}_k) \quad (21)$$

$${}^w\mathbf{p}_{\text{WK}}(t_k) = {}^w\mathbf{p}_{\text{WK}}(t_{k-1}) \quad (22)$$

where  $\text{fk}(\cdot)$  is the forward kinematic function that returns the foot location in base coordinates and  $\tilde{\mathbf{q}}$  are the measured joint positions.

Since the contact location in world coordinates does not change over the interval (see Equation 22), the difference in position  $\mathbf{p}_k - \mathbf{p}_{k-1}$  can be inferred from the forward kinematics only, by subtracting and rearranging Equations 20–21. Finally, the discrete differentiation is then simply computed by dividing  $\mathbf{p}_k - \mathbf{p}_{k-1}$  by the time interval (Equation 23).

$$\begin{aligned} \tilde{\mathbf{v}}_k &= \frac{\mathbf{p}_k - \mathbf{p}_{k-1}}{\Delta t} + \boldsymbol{\eta}^v = \\ &= \frac{\mathbf{R}_{k-1}\text{fk}(\tilde{\mathbf{q}}_{k-1}) - \mathbf{R}_k\text{fk}(\tilde{\mathbf{q}}_k)}{\Delta t} + \boldsymbol{\eta}^v \end{aligned} \quad (23)$$

$$\mathbf{z}_k = \tilde{\mathbf{v}}_k \quad (24)$$

where the covariance matrix  $\mathbf{P}_k^v = \mathbf{P}^v$  is defined from fixed values (empirically found) which are increased when special events (striking contact, breaking contact) occur.

#### 4.2.2 Quadrupeds

Quadruped robots are typically equipped with high precision joint encoders from which low noise joint velocity measurements can be derived. However, a major challenge is achieving accurate contact estimation since field-ready quadrupeds are not typically equipped with direct contact sensors as they easily break during operation.

##### 4.2.2.1 Contact Classification

Quadruped robot feet are usually approximated to be point-like and then assumed to exert only pure forces onto the ground. These forces can be estimated for each individual leg  $\ell \in \{\text{LF, RF, LH, RH}\}$  using the base acceleration  $\dot{\boldsymbol{\omega}}, \dot{\mathbf{v}}$  and torques  $\boldsymbol{\tau}$ :

$$\mathbf{f}^\ell = - \left( \mathbf{J}(\mathbf{q})^\top \right)^\dagger \left( \boldsymbol{\tau} - \mathbf{h}_q - \mathbf{F}^\top \begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\mathbf{v}} \end{bmatrix} \right) \quad (25)$$

where:  $\mathbf{J}(\cdot)$  is the foot Jacobian,  $\mathbf{h}_q$  are the Coriolis effects, and  $\mathbf{F}$  is the matrix of spatial forces required at the floating base to support unit accelerations about each joint variable (see Featherstone (2008)).

Let  $f_k^\ell \in \mathbb{R}$  be the vertical component of  $\mathbf{f}^\ell \in \mathbb{R}^3$  at time  $t_k$ . Thus, we model the probability for a particular foot to be in *firm, static and stable* contact with the following Sigmoid function:

$$\mathcal{P}_k^\ell(s_k^\ell = 1 | \mathbf{f}_k^\ell) = \frac{1}{1 + \exp(-\beta_1 f_k^\ell - \beta_0)} \quad (26)$$

where  $s_k^\ell \in \mathbb{B}$  is a binary value that indicates contact/no-contact for foot  $\ell$  at time  $t_k$ . We learn the Sigmoid parameters  $\beta_0$  and  $\beta_1$  using a logistic classifier, as described in Camurri et al. (2017).

For each leg, we determine a (binary) contact state  $s_k^\ell = 1$  if  $\mathcal{P}_k^\ell > 0.5$  and  $s_k^\ell = 0$  otherwise.

##### 4.2.2.2 Measurements

Having determined the set of legs in contact, for a given leg  $\ell$  the robot’s linear velocity can be computed as follows:

$${}^B\mathbf{v}_{\text{WB}} = -{}^B\mathbf{v}_{\text{BK}} - {}^B\boldsymbol{\omega}_{\text{WB}} \times {}^B\mathbf{p}_{\text{BK}} \quad (27)$$

From the sensed joint positions and velocities  $\tilde{\mathbf{q}}, \dot{\tilde{\mathbf{q}}}$  and their additive noises  $\boldsymbol{\eta}^q, \boldsymbol{\eta}^{\dot{q}}$ , we can rewrite

Eq. (27) as a linear velocity measurement of the robot's base, computed using the leg  $\ell$ :

$$\tilde{v}_\ell = -J(\tilde{\mathbf{q}} - \boldsymbol{\eta}^q) \cdot (\tilde{\dot{\mathbf{q}}} - \boldsymbol{\eta}^{\dot{q}}) - \boldsymbol{\omega} \times \text{fk}(\tilde{\mathbf{q}} - \boldsymbol{\eta}^q) \quad (28)$$

where  $\text{fk}(\cdot)$  and  $J(\cdot)$  are the forward kinematics function and its Jacobian, respectively.

As in Bloesch et al. (2013), we collect all the effects of noise into one additive term  $\boldsymbol{\eta}^v$ :

$$\tilde{\mathbf{v}}_k^\ell = -J(\tilde{\mathbf{q}}_k) \tilde{\dot{\mathbf{q}}}_k - \boldsymbol{\omega} \times \text{fk}(\tilde{\mathbf{q}}_k) + \boldsymbol{\eta}^v \quad (29)$$

Since multiple legs can be in contact simultaneously, we define the velocity measurement as a weighted average using the set of legs in contact, where weights are using the contact probabilities  $\mathcal{P}_k^\ell$  from Equation 26:

$$\tilde{\mathbf{v}}_k = \frac{\sum \mathcal{P}_k^\ell \tilde{\mathbf{v}}_k^\ell}{\sum \mathcal{P}_k^\ell} + \boldsymbol{\eta}^v \quad \forall \ell \mid s^\ell \neq 0 \quad (30)$$

$$\mathbf{z} = \tilde{\mathbf{v}}_k \quad (31)$$

The adaptive covariance  $\mathbf{P}_k^v$  is associated with the velocity measurement and accounts for harsh impact forces (up to 600 N for a 90 kg robot trotting). These forces can severely undermine the estimation performance, because compression of the legs or the ground causes wrong kinematic measurements which translate into velocity and position errors.

The covariance is computed as the combination of a fix term (from encoder noise datasheet), the inter-leg velocity covariance  $\mathbf{D}_k$  and a term that is proportional to force discontinuities (that are caused by impacts). For convenience, let  $c_k$  be the total number of detected contact legs at time  $t_k$ . The inter-leg covariance is defined as the covariance matrix of the velocity contributions from the contact legs:

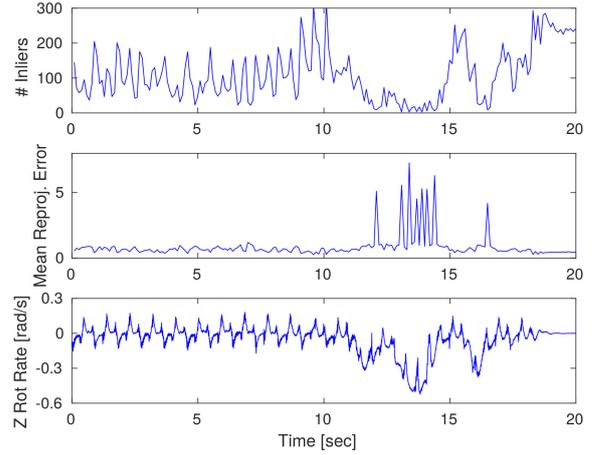
$$\begin{aligned} \mathbf{D}_k &= \frac{1}{c_k} \sum (\tilde{\mathbf{v}}_k - \tilde{\mathbf{v}}_k^\ell)(\tilde{\mathbf{v}}_k - \tilde{\mathbf{v}}_k^\ell)^\top \\ &\simeq \begin{bmatrix} \sigma_x^2 & 0 & 0 \\ 0 & \sigma_y^2 & 0 \\ 0 & 0 & \sigma_z^2 \end{bmatrix} = \Lambda(\sigma_x^2, \sigma_y^2, \sigma_z^2) \end{aligned} \quad (32)$$

The force discontinuity is defined as the average normal force absolute difference per each leg:

$$\Delta f = \frac{1}{c_k} \sum_{\forall \ell} |f_k^\ell - f_{k-1}^\ell| \quad (33)$$

From Equations 32–33, the final covariance for the velocity measurement is:

$$\mathbf{P}_k^v = \mathbf{P}_0^v + \left[ \frac{1}{2} \left( \Lambda(\sigma_x, \sigma_y, \sigma_z) + \mathbf{I}_3 \frac{\Delta f}{\alpha} \right) \right]^2 \quad (34)$$



**Figure 3.** Visual odometry performance during a trotting sequence on HyQ: the robot first trots forward at 0.3 m/s and then turns in place sharply over a 5 s period. During the initial trotting phase, VO performance is satisfactory. However, image blur causes the number of inliers to fall and mean re-projection error to spike. During this part of the experiment, no VO measurements are incorporated into the main motion estimate.

where  $\alpha$  is a constant normalization factor, empirically determined.

### 4.3 Zero Velocity Bias Estimation

The yaw drift due to bias evolution can be significant over long periods of time. Yaw error is also the dominant source of drift in any state estimator or SLAM system. For this reason, in Ma et al. (2016) zero velocity updates were used to measure rotation rate bias estimates.

In our system, we continually check for periods where the robot is stationary using the joint velocities and GRF.

When the robot is stationary for at least 400 ms, the gyro bias is updated to the average angular velocity recorded during the stationary period:

$$\tilde{\mathbf{b}}_k^\omega = \frac{1}{t_k - t_i} \sum_{j=i}^k \tilde{\boldsymbol{\omega}}_j \quad (35)$$

$$\mathbf{z}_k = \tilde{\mathbf{b}}_k^\omega \quad (36)$$

where  $t_k - t_i > 400$  ms.

Since the bias is generally a very small quantity (i.e.,  $\boldsymbol{\omega} \ll \mathbf{b}^\omega$ ), the covariance associated to the measurement can be typically set to very small values without affecting the control system of the robot.

## 4.4 Visual Odometry

Visual Odometry estimates the pose of the robot by tracking features on camera images. The VO estimate frequency is typically in the range of 10–30 Hz, which corresponds to the camera frame rate.

When used in combination with LIDAR odometry, the benefit of VO is twofold. First, it makes the overall estimated trajectory smoother if compared with a inertial-kinematic-LIDAR only system, as it reduces the drift rate in between two LIDAR updates. Second, the reduced drift rate also helps the LIDAR registration itself, as the sparsity of the LIDAR scans requires the accumulation of scans over time before performing the registration. Therefore, a smaller drift rate during the accumulation produces higher quality point clouds to be registered.

Our visual odometry pipeline is based on the FOVIS algorithm by Huang et al. (2011). The measurements are loosely integrated into the filter as relative pose measurements between frames. This would allow the use of other VO algorithms, such as ORB-SLAM (Mur-Artal et al. (2015)), SVO (Forster et al. (2017b)) or VINS-Mono (Qin et al. (2018)), to name a few. FOVIS was chosen because of its computational efficiency.

The only input to FOVIS is a sequence of stereo image pairs. It tracks FAST features in a key-frame approach to estimate incremental camera motion. Given two keyframes at times  $t_i, t_j$ , we denote the estimated relative motion of the camera between these two times as  ${}_{c_i}\tilde{\mathbf{T}}_{c_i c_j} = \tilde{\mathbf{T}}_{c_i j}$ . Using the known camera-to-base frame transformation,  ${}_{\mathbf{B}}\mathbf{T}_{\mathbf{BC}}$ , this can be expressed at the corresponding estimate of the motion of the base frame as:

$$\tilde{\mathbf{T}}_{\mathbf{B}ij} = {}_{\mathbf{B}}\mathbf{T}_{\mathbf{BC}} \tilde{\mathbf{T}}_{c_i j} ({}_{\mathbf{B}}\mathbf{T}_{\mathbf{BC}})^{-1} \quad (37)$$

We integrate the VO estimate for a time window  $t_j - t_i$  which is typically 2–3 s. When used in combination with the LIDAR module, we then form a position measurement in the world frame, as follows:

$${}_{\mathbf{w}}\tilde{\mathbf{T}}_{\mathbf{WB}}(t_j) = {}_{\mathbf{w}}\mathbf{T}_{\mathbf{WB}}(i) \tilde{\mathbf{T}}_{\mathbf{B}ij} \quad (38)$$

$$\tilde{\mathbf{p}}_j = \text{trans} \left( {}_{\mathbf{w}}\tilde{\mathbf{T}}_{\mathbf{WB}}(t_j) \right) \quad (39)$$

$$z_j = \tilde{\mathbf{p}}_j \quad (40)$$

where the pose of the robot at time  $t_i$  is taken from filter's history of states. Note that we choose to use only the translational part of Equation 38 for the EKF filter update, as yaw corrections from the LIDAR are more accurate and sufficiently frequent.

Without the LIDAR module, the VO update can also include rotational components (typically only yaw, since roll and pitch are observable from the IMU).

Note that the update could be delayed in time (i.e.,  $t_j < t_k$ ), therefore the filter will re-apply the chain of measurements from time  $t_j$  to  $t_k$ , as explained in Section 5.1.

The covariance matrix for the measurement was manually set to fixed values. However, when the FOVIS algorithm reports failure, the measurement is discarded. The algorithm reports failure in three cases: 1) when the number of inlier features being tracked is below a threshold (10 in our case); 2) when the solution of the optimization is degenerate; 3) when the reprojection error is higher than a threshold (1.5 pixels in our case).

An example of failure is provided by Figure 3, where at time 12 s the number of inliers drops below the threshold (top plot) and the reprojection error increases significantly (medium plot) due to an abrupt robot rotation which caused motion blur (bottom plot).

## 4.5 LIDAR Odometry

Our LIDAR odometry is based on the Auto-tuned Iterative Closest Point (AICP) algorithm by Nobili et al. (2017b), which improves the ICP implementation from Pomerleau et al. (2013) by making it more robust against significant changes in overlap between the clouds to be registered.

The rotating Hokuyo LIDAR sensor inside the Multisense SL (mounted on Atlas, Valkyrie and HyQ), as well as the Velodyne VLP-16 (mounted on ANYmal) produce very sparse point clouds which cannot be used directly for scan to scan registration.

Therefore, we accumulate consecutive measurements from the sensor as a *reference* point cloud. The filter's state is used as the source of robot poses during the accumulation. We assume that the pose drift during the accumulation is small enough not to create significantly distorted reference point clouds. In this context, the VO module is important, as it keeps the drift bounded during the accumulation period.

Once a sufficiently dense reference cloud is obtained, a sequence of *reading* point clouds are accumulated and registered against the reference for motion estimation. The result of the registration constitutes an additional relative pose measurement for the EKF.

#### 4.5.1 Reference Update

Using the first accumulated point cloud as the reference and registering the forthcoming clouds as reading is effective only in confined scenes. When the robot travels far away from its initial location, this method is intractable due to the decreasing overlap between the source and the reading clouds, eventually resulting in ICP failure.

To guarantee the sufficient overlap between the reference and the reading clouds, we update the reference clouds whenever the overlap drops below a safety threshold. In long range missions, such as the one described in Section 7.5, we conveniently forced a reference update after the robot has traveled 5 m from its initial location. This way, the drift is effectively bounded while having sufficient overlap for data association.

#### 4.5.2 Pre-filtering

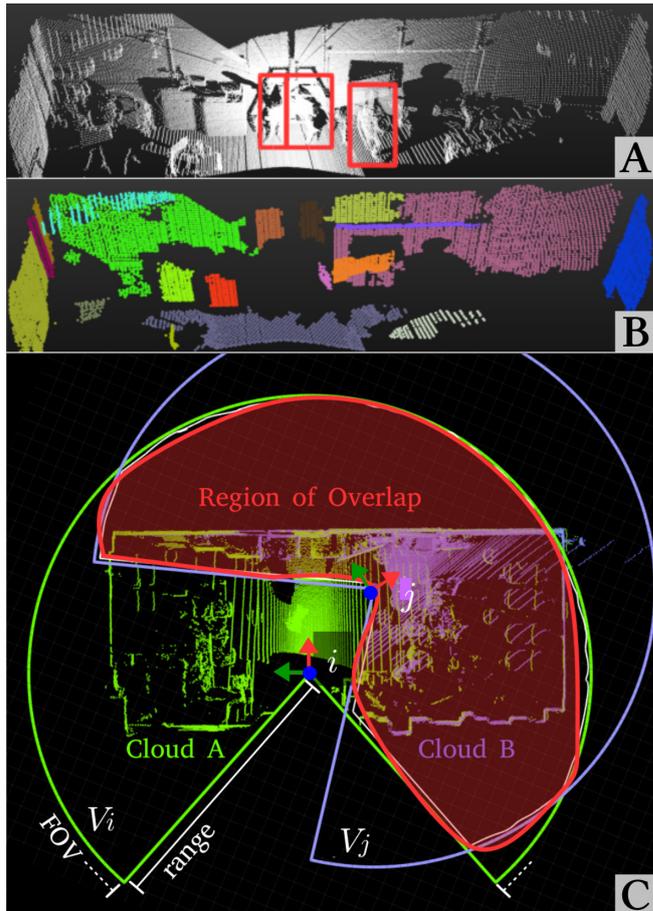
According to Segal et al. (2009), point-to-plane registration has proven to have superior performance than point-to-point. Therefore, we extract planar macro-features (e.g., walls, doors, ceilings) and implicitly discard all other entities (including dynamic obstacles). We also apply a voxel filter with a leaf size of 8 cm to uniformly downsample the clouds. This step is necessary to equalize the contribution from all the points during the optimization process, as point clouds are denser in the proximity of the sensor.

For planar surface extraction, we adopt a region growing strategy: a patch which is larger than a specific area (e.g., 30 cm  $\times$  30 cm) is accepted for further process. An example of pre-filtering output is shown in Figures 4A and 4B.

#### 4.5.3 Auto-tuned ICP

Most ICP solutions assume a constant overlap between reference and reading clouds. However, when partial occlusion occurs (e.g., during passage through a narrow door), this assumption is violated and the massive concentration of points on the occludant (e.g., the walls besides the door) can cause wrong correspondences.

In contrast, we continuously estimate the amount of overlap between the point clouds and automatically tune the ICP inlier ratio for robust registration. The overlap parameter is proportional to the true positive correspondences (i.e., the higher is  $\Omega$ , the larger the number of true positive matches and vice versa). In the following subsection, we briefly describe how  $\Omega$  is computed. More details can be found in Nobili et al. (2017b).



**Figure 4.** Pre-filtering and Outlier filtering. (A) raw point cloud from Valkyrie’s dataset, people are outlined in red. (B) after pre-filtering. People and small irrelevant features have been filtered-out. (C) region of overlap as the result of the analysis of the two volumes of the point clouds.

#### 4.5.4 Overlap Filter

The overlap parameter  $\Omega$  is computed in a point-wise fashion (Figure 4C). Let  ${}_wA, {}_wB$  be the reference and reading point clouds acquired at times  $t_i, t_j$  and whose points have been expressed in world coordinates by using a prior from the EKF. Each cloud is confined into the volumes  $V_i, V_j$  by the sensor Field-of-View (FoV). The intersection of the two volumes defines an overlap region (red in figure). If  $S_i$  and  $S_j$  are the points of  $A_i$  and  $B_j$  belonging to the overlap region, we can define the overlap parameter  $\Omega$  as:

$$\Omega = \frac{|S_i|}{|A|} \cdot \frac{|S_j|}{|B|} \quad (41)$$

where  $|\cdot|$  indicates the number of points in the cloud.

We use the overlap parameter from Eq. (41) to dynamically set the inlier ratio of the ICP algorithm. If  $0.2 < \Omega < 0.7$  we set the inlier ratio to  $\Omega$ . If  $\Omega$  is below 0.2, the inlier ratio is set to 0.2, as it is the minimum required for ICP registration. Finally, if  $\Omega$  exceeds 0.7, the inlier ratio is bounded to 0.7 to avoid overestimation.

We follow three heuristics to determine if an alignment is successful. First, the mean residual point-wise error should be smaller than the threshold  $\alpha$ :

$$MSE = \frac{1}{n} \sum_{i=1}^n r_i < \alpha \quad (42)$$

where  $r_1, \dots, r_n$  are the residual distances between the accepted matching points in the input clouds. Second, the median of the residual distribution,  $Q(50)$ , should be smaller than the threshold  $\alpha$ :

$$Q(50) < \alpha \quad (43)$$

Third, the quantile corresponding to the overlap measure should be also smaller than  $\alpha$ :

$$Q(\Omega) < \alpha \quad (44)$$

The first two conditions are commonly used metrics of robustness, while the third automatically adapts to the degree of point cloud overlap. The parameter  $\alpha$  was set to 0.01 m during our experiments.

### 4.5.5 Measurements

Once the two clouds  $A, B$  have been successfully registered, the estimate  $\tilde{\mathbf{T}}_{B_i j}$  of the robot's base relative pose between time  $t_i$  and  $t_j$  is available, similarly to Equation 37. Thus, the measurement is incorporated in the same way, but including rotation:

$${}_{\mathbf{w}}\tilde{\mathbf{T}}_{\mathbf{w}B}(t_j) = {}_{\mathbf{w}}\mathbf{T}_{\mathbf{w}B}(i) \tilde{\mathbf{T}}_{B_i j} \quad (45)$$

$$\tilde{\mathbf{p}}_j = \text{trans} \left( {}_{\mathbf{w}}\tilde{\mathbf{T}}_{\mathbf{w}B}(t_j) \right) \quad (46)$$

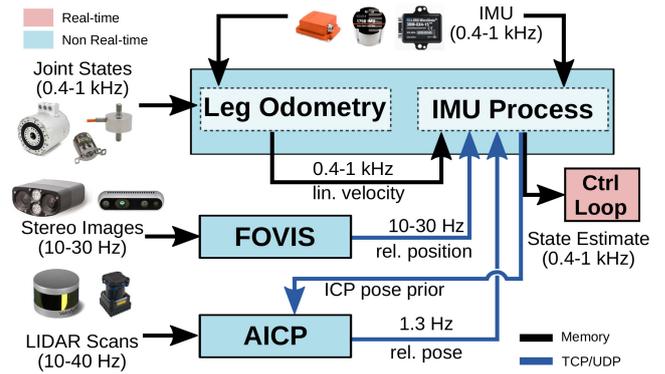
$$\tilde{\mathbf{R}}_j = \angle \left( {}_{\mathbf{w}}\tilde{\mathbf{T}}_{\mathbf{w}B}(t_j) \right) \quad (47)$$

$$\mathbf{z}_j = [\tilde{\mathbf{p}}_j \quad \tilde{\mathbf{R}}_j] \quad (48)$$

where again the absolute pose of the robot at time  $t_i$  is taken from the filter and the covariance matrix is set to fixed values. Note the time index is  $j$  as typically the measure is delayed (i.e.,  $t_j < t_k$ ).

## 5 IMPLEMENTATION

A block diagram of our system is presented in Figure 5. Even though the Pronto modules can all



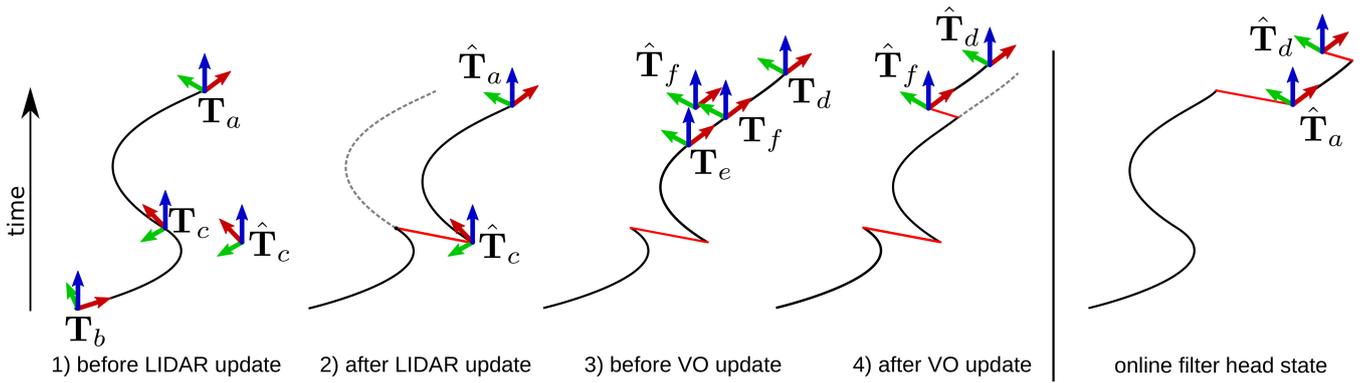
**Figure 5.** Block diagram of our system: the IMU process model and leg odometry run on the control computer in a real time process (in pink) on the control computer, while the other modules run on separate computers in the user space (light blue). These modules output filter measurements as ROS messages which are exchanged with the real-time domain through native shared memory mechanisms.

be run on a single machine, it is common practice in legged robot design to distribute the computation across two separate computers: a Control PC connected to actuators and proprioceptive sensors running a real-time operating system; and a Vision PC for exteroceptive sensor processing. This design architecture has been adopted for all the robots evaluated in this paper. Its main advantage is that the more critical operations are unaffected by potential delays, failure or overloads caused by the resource intensive data processing from camera and LIDAR.

Therefore, the IMU prediction and leg odometry updates are performed within the same UNIX process running on the Control PC. After every IMU process step, the estimator immediately shares the filter state with the control system via a real time interface based on shared memory. The same estimate is also available on the network for other modules to use (e.g., as prior for ICP).

The FOVIS and AICP processes are run on the Vision PC. Both modules are decoupled from the core estimator, which receives the updates as timestamped messages via a TCP or UDP channel (e.g., ROS messages). This allows Pronto to perform the core IMU/leg odometry, which is more critical, and to incorporate measurements if and when they are available.

On all platforms, the computation is carried out on consumer grade processors (e.g., equivalent to Intel i7 for laptop), with no need for GPU processing.



**Figure 6.** Example illustrating how VO and LIDAR measurements can be incorporated into the filter despite having much higher latency than the IMU process model. In black is the best estimate of the trajectory at that instance, in red are updates introduced by incorporated measurements and dashed gray lines are parts of the trajectory which are recomputed. For clarity, the magnitude of the corrections is exaggerated. Elapsed time is indicated in the upward direction.

## 5.1 Measurement History

The implementation of the filter maintains a history of measurements (with their covariance), filter prior/posterior states, and filter covariances covering a time window of typically 10s. This allows incorporation of asynchronous corrections from VO and LIDAR which have significant latency.

In Figure 6, we explain the concept with a toy example. In black is the best estimate of the current state and history at that moment in time. In red are discontinuities caused by EKF updates (exaggerated for clarity). In dashed gray are portions of the filter history which are overwritten due to a received measurement.

**Event 1:** At time  $t_a$ , the head of the filter points to  $\mathbf{T}_a$ . This state is the result of predictions and measurement integrations available up to time  $t_a$ . At this same time, the filter receives a delayed LIDAR measurement with timestamp  $t_c$  (with  $t_c < t_a$ ). In particular, the measurement involves the relative pose between  $\mathbf{T}_b$  and  $\mathbf{T}_c$  (with  $t_b < t_c$ ). The history consists of a window of measurements, filter states, and filter covariances, ordered by timestamp. Since the window is longer than the time interval  $t_a - t_b$ , the filter head can be moved back to  $\mathbf{T}_c$ , which corresponds to the state recorded at time  $t_c$ .

**Event 2:** The LIDAR measurement is incorporated as an EKF correction resulting in the posterior estimate  $\hat{\mathbf{T}}_c$ . At this point, all the measurements in the history with timestamps after  $t_c$  are re-applied to the filter, as if they had been received after the LIDAR measurement. As a result, the filter head at time  $t_a$  becomes  $\hat{\mathbf{T}}_a$ . The past trajectory

(dashed gray line) is therefore overwritten. The new current state  $\hat{\mathbf{T}}_a$  is the same as it would have been if the LIDAR measurement was received at time  $t_c$  instead of  $t_a$ .

**Event 3:** Over the next period of time, the filter continues to propagate the head of the estimator using the IMU process model and leg odometry. At time  $t_d$ , a new visual odometry measurement is created which measures the relative transformation of the body frame between time  $t_e$  and time  $t_f$ . This measurement is typically received with 150–300 ms of delay.

**Event 4:** We wish to use this information to correct the pose of the robot towards  $\mathbf{T}_f$ , as described in Section 4.4. The key step is that this correction to the filter is carried out using the re-filtered trajectory (mentioned in Event 2). After the correction is applied, the head of the filter becomes  $\hat{\mathbf{T}}_d$  and the estimator continues as normal.

The final sub-figure (on the right) shows the state of the head of the filter over the course of the example. This is the running estimate that would have been available to the controller online.

Note that the proposed framework qualifies as an odometry system, as no loop closures are performed. Therefore, typical corrections from the exteroceptive modules are in the order of few centimeters (*cf.* Figure 9C top right). These discontinuities are small enough to be dealt with by the position controller acting on the robot base with appropriate gains. Bigger discontinuities, such as the ones from a SLAM system, are typically addressed by using two different reference frames

for control and for global path planning (e.g., Wim Meeussen (2010)).

## 5.2 Software Structure

The framework presented in this paper is available to the research community as three open-source repositories:

- `pronto`<sup>1</sup>: library implementations of the EKF inertial process model and the Leg Odometry modules described in Sections 4.1 and 4.2, respectively
- `fovis_ros`<sup>2</sup>: ROS wrapper for the FOVIS algorithm (previously open-source but not ROS compatible)
- `aicp_mapping`<sup>3</sup>: implementation of the AICP algorithm described in Section 4.5.

The first repository is independent from the others and contains all the code necessary to implement a proprioceptive state estimator on a legged robot. To deploy the algorithm on a legged robot of choice, either with or without ROS, the implementation of the forward kinematics API and the creation of a dedicated executable is required. A complete example of a deployment on the ANYmal robot is also provided.

## 6 EXPERIMENTAL PLATFORMS

In the following sections, we describe the relevant characteristics of the experimental platforms used: the Atlas and Valkyrie humanoid robots and the HyQ and ANYmal dynamic quadruped robots. A summary of the main sensors mounted on the robots is provided by Table 1.

### 6.1 Atlas

Atlas (version 5, Figure 1A) is a 195 cm high, 95 kg heavy, 28-DoF hydraulic robot manufactured by Boston Dynamics for the DARPA Robotics Challenge. Each leg has six joints (three hip, one knee, and two ankle joints), the position of which are estimated from the measured travel of their hydraulic actuators using a Linear Variable Differential Transformer (LVDT). Since the accuracy of these devices is limited, the joint velocities are very noisy and therefore not used directly for leg odometry (*cf.* Section 4.2.1). Other measurement nonlinearities such as backlash have

been addressed the same way as Koolen et al. (2016).

Located at the pelvis is a tactical KVH 1750 IMU equipped with Fiber Optic Gyro (FOG) for accurate angular velocity measurements.

The main source of exteroceptive signals is the Carnegie Robotics Multisense SL, a tri-modal ruggedized sensor that includes: a rotating Hokuyo UTM-30LX-EW; a high quality rolling shutter RGB stereo camera with a 7 cm baseline; and a FPGA implementation of the stereo Semi-Global Matching algorithm by Hirschmüller (2008), to provide dense 3D point clouds at nominal camera frequency. All these signals are synchronized in hardware through the FPGA. The laser produces 40 line scans per second with 30 m maximum range — while spinning about the forward-facing axis. Every few seconds, it spins half a revolution and a full 3D point cloud is accumulated with a Field-of-View (FoV) of  $220^\circ \times 180^\circ$ .

### 6.2 Valkyrie

Valkyrie (Figure 1B) is a 1.87 m tall, 129 kg, and 44-DoF (28-DoF without hands) electrically actuated robot developed by NASA for the DARPA Robotics Challenge and space operations (Radford et al. (2015)). As for Atlas, each leg has 6-DoF, with 3-DoF hips, 1-DoF knee and 2-DoF ankles. The hip and knee motors are rotary actuators whose rotation is measured by magnetic encoders and torque by a measuring the spring deflection. The ankle joints are linear with encoders located at the rotation for position and load cells on the shaft for torque measurement, respectively.

Even though the robot is equipped with several cameras for visual servoing, the main exteroceptive sensor considered in this paper is again the Multisense SL. The FoV of the LIDAR is reduced to  $180^\circ \times 120^\circ$  by a plastic cover over the head.

### 6.3 HyQ

HyQ (Figure 1C) is a torque-controlled Hydraulic Quadruped robot (Figure 1) developed by Semini et al. (2011) at the Istituto Italiano di Tecnologia (IIT). The system is 1 m long, and weighs approximately 85 kg. Its 12 revolute joints have a rotational range of  $120^\circ$  and a maximum torque of 160 N m. The 1 kHz sensors are read by a control computer (using a real-time operating system). All other sensors are connected to a perception computer and are passively synchronized with the real-time sensors as described in Olson (2010).

As for Atlas and Valkyrie, the robot's main exteroceptive sensor is the Carnegie Robotics

<sup>1</sup> [github.com/ori-drs/pronto](https://github.com/ori-drs/pronto)

<sup>2</sup> [github.com/ori-drs/fovis\\_ros](https://github.com/ori-drs/fovis_ros)

<sup>3</sup> [github.com/ori-drs/aicp\\_mapping](https://github.com/ori-drs/aicp_mapping)

Sensor	Model	Hz	Specs
<b>ATLAS</b>			
IMU	KVH 1750	333	<i>Init Bias</i> : 0.5°/h   0.5 mg <i>Bias Stab</i> : 0.05°/h   0.05 mg
Stereo Camera	Multisense SL	10	<i>Res</i> : 1024 × 1024 px <i>FoV</i> : 80° × 80° <i>Imager</i> : CMV4000 4MP
LIDAR	Hokuyo UTM-30LX-EW	40	<i>FoV (full rot.)</i> : 220° × 180°
Encoder	N/A	333	<i>Res</i> : <0.0045°
Torque	N/A	333	<i>Res</i> : N/A
<b>VALKYRIE</b>			
IMU	3DM-GX4-25	500	<i>Init Bias</i> : 0.05°/s   2 mg <i>Bias Stab</i> : 10°/h   0.04 mg
Stereo Camera	Multisense SL	10	<i>FoV (full rot.)</i> : 180° × 120°
Encoder	N/A	500	<i>Res</i> : 0.0043°
F/T	ATI Omega85	500	<i>Res</i> : 0.07–0.1 N 0.02–0.03 N m
<b>HyQ</b>			
IMU	KVH 1775	1000	<i>Init Bias</i> : 0.5°/h   0.5 mg <i>Bias Stab</i> : 0.05°/h   0.05 mg
Stereo Camera	Multisense SL	10	See above
Encoder	AEDA3300-BE1	1000	<i>Res</i> : <0.0045°
Force	Burster 8417	1000	<i>Res</i> : <25 N
Torque	N/A	1000	<i>Res</i> : N/A
<b>ANYMAL</b>			
IMU	Xsens MTi-100	400	<i>Init Bias</i> : 0.2°/s   5 mg <i>Bias Stab</i> : 10°/h   15 mg
Stereo Camera	RealSense D435	30	<i>Res</i> : 848 × 480 px <i>FoV</i> : 91.2° × 65.5° <i>Imager</i> : IR global shutter
Encoder	ANYdrive	400	<i>Res</i> : <0.025°
Torque	ANYdrive	400	<i>Res</i> : <0.1 N m

**Table 1.** Sensor specifications divided by robot.

Multisense SL. The stereo camera was configured to capture  $1024 \times 1024$  images at 10 Hz. Figure 10 shows an example of a left camera image and a depth image taken during an experiment — indicating the challenging scenarios we target.

## 6.4 ANYmal

ANYmal (version B, Figure 1D) is a 12-DoF electrically actuated quadruped robot initially designed by Hutter et al. (2016) at ETH Zurich and now manufactured by ANYbotics. It is 80 cm long and weighs 33 kg. Its series elastic actuators can deliver up to 40 N m of torque and provide accurate measurements of the joint position, velocity (internally computed by differentiation) and torque (by spring deflection measurement).

The robot is equipped with a XSens MTi-100 industrial grade IMU, a RealSense D435 camera at the front (for visual stereo odometry and local mapping) and a Velodyne VLP-16 LIDAR on the top (for localization and global mapping).

## 7 EXPERIMENTAL RESULTS

We carried out a series of experiments with the Atlas, Valkyrie, HyQ and ANYmal robots over the course of four years. We present summary results which have a combined time of 2 h and 13 min and 1.37 km of distance traveled, respectively. A summary of the experimental results divided by dataset and robot is available in Table 2.

### 7.1 Evaluation Protocol

We aim to evaluate the estimation performance both quantitatively and qualitatively, with a focus on real-world scenarios and online/real-time execution.

#### 7.1.1 Ground Truth

The experimental results presented in this section have been collected over the span of several years in a variety of different conditions and platforms. For this reason, it was not always possible to generate the ground truth poses from the same source (e.g., motion capture). The last column of Table 2 indicates the experiments where ground truth was available.

For all indoor experiments on HyQ and Valkyrie (lines 2–4, Table 2), we have used a Vicon motion capture system to have millimeter accurate ground truth poses at 100 Hz.

For the HyQ outdoor experiments (line 6 in table), we exploited situations where the robot was completely stationary to accumulate six full sweeps of LIDAR scans from different locations to reconstruct the scene in post-processing via ICP registration. Since the LIDAR was perfectly stationary, the accumulation was performed for at least two full turns (65k points per scan), and the overlap was more than 70 %, we ascribe the accuracy of the reconstruction to the one of the sensor, which is 3 cm for the experimental area evaluated. Then, we have generated a ground truth trajectory by aligning the point cloud data from the onboard LIDAR with the prior map in post-processing. Note that this trajectory is different than the one obtained during online estimation, as there was no prior map involved in this process.

The experiment with ANYmal (line 7, Table 2) have been paired with ground truth from a Leica TS16 laser tracking system, which tracked the robot’s position with millimeter accuracy using a reflective prism on the robot. The data from the laser tracker was then spatio-temporally aligned with the IMU to get ground truth poses via an offline batch optimization, as described in Burri et al. (2016).

Exp. N	Robot	RPE [m]	VO	AICP	OL	CL	T [s]	DT [m]	A [m <sup>2</sup> ]	GT
1	Atlas	$\leq 0.03^*$	-	✓	✓	✓	1236	16	154	-
2	Valkyrie	0.016	-	✓	✓	✓	341	12	78	✓
3	Valkyrie	0.016	-	✓	✓	✓	50	2.5	78	✓
4	HyQ	0.027	✓	✓	✓	✓	1740	400	7.5	✓
5	HyQ	$\leq 0.03^{**}$	✓	✓	✓	✓	1740	400	9	-
6	HyQ	0.033	✓	✓	✓	✓	2640	300	100	✓
7	ANYmal	0.34 0.83	✓ -	✓ -	- -	- -	1996	240	1381	✓

**Table 2.** Summary of the experiments. **Legend:** Exp. N = Experiment number, RPE = Relative Pose Error (translational part, evaluated over 10 m distance, ATE = Absolute Translation Error, VO = Visual Odometry, OL = Online, CL = Control Loop, T = Time, DT = Distance Traveled, A = Area, GT = Ground Truth. \* By evaluation of the ground truth point cloud. \*\* By evaluation of the accuracy in returning to the initial position.

Finally, when ground truth was not available, we have designed the experiments such that the estimation performance could be measured by analyzing the robot's accuracy in returning to its initial position after several forward/backward motions.

### 7.1.2 Pose Estimation Performance

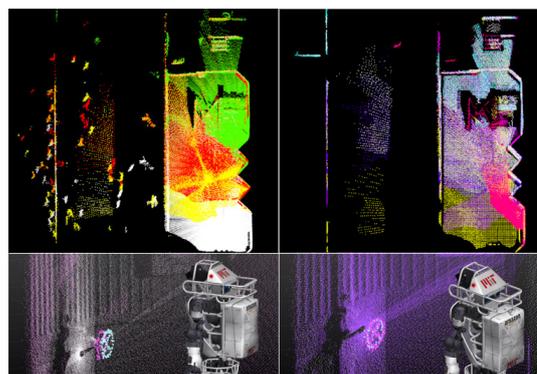
Since our proposed algorithm is an odometry system (i.e., no loop closures are preformed), we base our quantitative analysis on the mean translational component of the Relative Pose Error (RPE), defined by Sturm et al. (2012), over a distance of 10 m. The performance for each experiment is indicated in Table 2.

### 7.1.3 Control Loop Performance

We have evaluated the stability of the algorithm in real conditions by running the estimator in real-time on Valkyrie (to feed footstep planner). On Atlas and HyQ, we also closed the control loop with the estimator. The control loop test implicitly evaluates the quality of the velocity estimates, which are directly used by the locomotion controllers. For the ANYmal the execution was tested offline, but at nominal speed and on consumer grade laptop with comparable performance to the hardware mounted on the robot. In this case, the suitability for control loop was assessed by looking at the signal smoothness.

## 7.2 Atlas Experiments

The Atlas dataset (Table 2, Experiment 1) was collected during a run by the MIT team at the



**Figure 7.** AICP performance on the DRC Finals dataset with Atlas. Top: a top view of the alignment of 206 point clouds during the run — left: raw clouds with people, right: filtered clouds. Bottom left: state estimation without applying correction, valve perceived in different locations by successive clouds. Bottom right: with successful localization, consistent estimate of the affordance.

DARPA Robotics Challenge Finals (Pomona, CA 2015). It consists of 20 min 36 s of continuous operation in a semi-structured environment of 14 m × 11 m, with walls on the right side of the robot and an open-space populated by a crowd on the left. The robot walks through the test scenario along a 16 m path while passing over uneven terrain and manipulating objects (Figure 1A). Accurate maps of the environment were obtained in post-processing.

During the whole competition, Pronto (without AICP) was used to close the control loop. Its

low-drift estimation performance (evaluated to be approximately 1.67% travelled distance in preliminary indoor tests) allowed to successfully traverse uneven and rough terrain, although pauses for re-localization were necessary.

Later on, further offline tests were carried out after the integration of the AICP module. This time, the system performance was qualitatively evaluated from careful observation of the map after the run (Figure 7), where the estimated trajectory is close to error free (approximately 3 cm error for the full run). People have been filtered-out and do not contribute to the alignment (top right). The algorithm is stable and robust enough to compute successful alignments during the entire run (with more than 14 m displacement and overlap decreasing to just 10%), satisfying requirement 2. Under the same conditions, standard ICP algorithms fails after 400 s as they are not accounting for dynamically changing point cloud overlap across the run.

### 7.3 Valkyrie Experiments

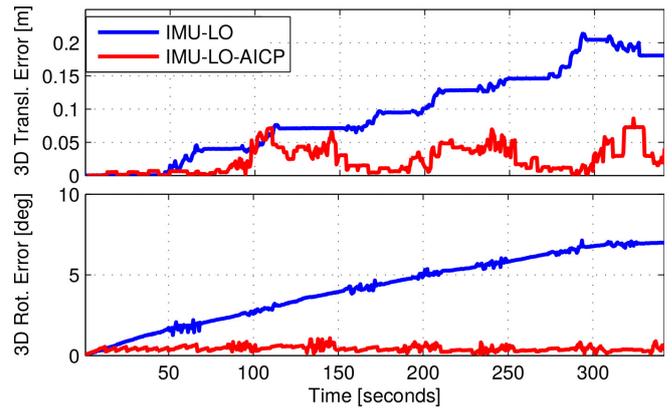
The state estimation framework was tested online on two different tasks: repeated walking on flat ground (Table 2, Experiment 2) and stair climbing (Table 2, Experiment 3).

#### 7.3.1 Repeated Walk to a Target

Valkyrie walked repeatedly forward towards a fixed target identified at the beginning of the run before reversing direction. Over the course of the experiment, the error in translation never exceeds 7.5 cm and is 1.6 cm on average, whereas the estimator without LIDAR has an unbounded drift (Figure 8), mostly dominated by yaw bias (see bottom plot). This satisfies the requirements about expected localization accuracy. Thanks to this localization performance, the robot could reach the goal target and maintain a precise pose estimate during the entire run. In contrast, using the proprioceptive state estimator the robot failed to reach the target due to odometry drift.

#### 7.3.2 Stair Ascend

Valkyrie was placed at 1 m from a staircase. The task was to walk towards it and climb up the steps. Planning was performed only once, at start. Over the course of this 50 s experiment, the median errors in translation and rotation were comparable to Experiment 2. This level of accuracy allowed the robot to safely perform the task without needing to re-plan. In contrast, during the DRC robots typically took a few steps at a time to climb stairs or transverse uneven terrain, pausing periodically to manually re-localize and re-plan. In this context,



**Figure 8.** Translational and rotational error for Experiment 2 (Valkyrie). The blue line shows the kinematic-inertial typical estimation drift while in red is the estimate with the AICP corrections.

our system was demonstrated to enable greater autonomy in task execution.

### 7.4 HyQ Experiments

On HyQ we performed experiments in two different scenarios. First, for Experiment 4, a repetitive trotting motion was carried out in a laboratory environment with a Vicon motion capture system for ground truth. Second, for Experiment 5 and 6, extensive testing was carried out in a poorly lit, industrial area with a featureless concrete floor, as well as test ramps and rock beds (Figure 9B). The environment, the different locomotion gaits (trotting and crawling) and the uneven terrains presented a large number of challenges to our algorithms and demonstrated the importance of using redundant and heterogeneous sensing. The robot's peak velocity when trotting was about 0.5 m/s, which is approximately half of typical human walking speed.

#### 7.4.1 Indoor Repeated Trot to a Target

The robot was commanded to continuously trot forward and backward to reach a fixed target (a particular line in Figure 9A). Robot position and velocity estimates were used by the controller to stabilize the robot motion while tracking the desired position, as described in Barasuol et al. (2013).

Periodically, the operator updated the target so as to command the robot to trot a further 10 cm forward. The experiment continued for a total duration of 29 min. At the end of the run, the robot had covered a total distance of about 400 m and trotted forward and backward 174 times.

In Table 2, we show that the drift is below 3 cm when combining IMU, Legged, Visual and

LIDAR odometry. By comparison, without any exteroceptive signals the drift was more than three times higher. When testing independently the addition of VO or LO, we noticed that incorporating VO reduces the drift rate relative to the base line system, while adding AICP achieves drift-free localization, since the AICP re-localizes against the same fixed map (the room).

To test the performance with uneven terrain and where the reference point cloud has to be updated due to longer paths, a second series of experiments was carried out in a larger environment.

#### 7.4.2 Outdoor Repeated Trot to a Target

An equivalent experiment has been performed within a section of a  $20\text{ m} \times 5\text{ m}$  industrial area surrounded by pallets, walls and air treatment machines. The robot repeated a forward-backward motion covering a  $6\text{ m} \times 1.5\text{ m}$  area towards a target placed at 5 m distance from its starting position (Figure 1C). The robot traveled about 400 m on a 0.5 m/s trotting gait, reaching the target 40 times without any user input at run time.

The results presented in this section show that the fully integrated state estimation system, leveraging IMU, leg odometry, VO and AICP data, produce a very low drift estimate of the robot state. However, no LIDAR reference cloud updates were triggered as the robot did not travel far from its initial location.

In the case of larger explorations, every reference update generates an accumulated error. The magnitude of this error depends on the residual error from the alignment of the new reference to the previous. In the case of HyQ, a reference update happens once every 10–13 m distance covered, depending on occlusions. In the following section, we present statistics from experiments where multiple LIDAR reference cloud updates were made.

#### 7.4.3 Outdoor Industrial Area Exploration

The robot explored the same industrial area described in the previous section. To test the system in different conditions, in some experiments we have added rough terrains and ramps (Figure 9B) with both crawling and trotting gaits at up to 0.5 m/s. Turning in place (as seen in Figure 3) represented an extra challenge for the state estimation system. Lighting conditions varied dramatically during data recording, from bright light to strong shadows and from day to night-time. In some experiments, on-board lighting was used. The dataset is summarized in Table 3 and consists of five runs, for a total duration of 44 min and 300 m traveled.

N	Gait	Duration	Area m <sup>2</sup>	Laser	Ramp
6a	crawl	869 s	20×5, F/B	5 RPM	✓
6b	crawl	675 s	20×5, F	5 RPM	✓
6c	trot	313 s	20×5, F/B	15 RPM	X
6d	trot	330 s	20×5, F/B	10 RPM	X
6e	trot	469 s	7×5, F/B	10 RPM	✓

**Table 3.** Detailed summary of the dataset used for Experiment 6, including log duration, size of arena, type of motion (F/B = forward/backward trajectory), laser spin rate, and terrain features.

No motion capture system was available in this space: to quantitatively evaluate the state estimation performance on the dataset, we built a prior map made up of a collection of 4 carefully aligned point clouds and we estimated drift relative to it.

##### 7.4.3.1 Crawling Gait

In the previous section, we have shown (while trotting) that integrating VO reduces the pose drift rate between the lower frequency AICP corrections. Here, we focus on the importance of using VO in addition to AICP.

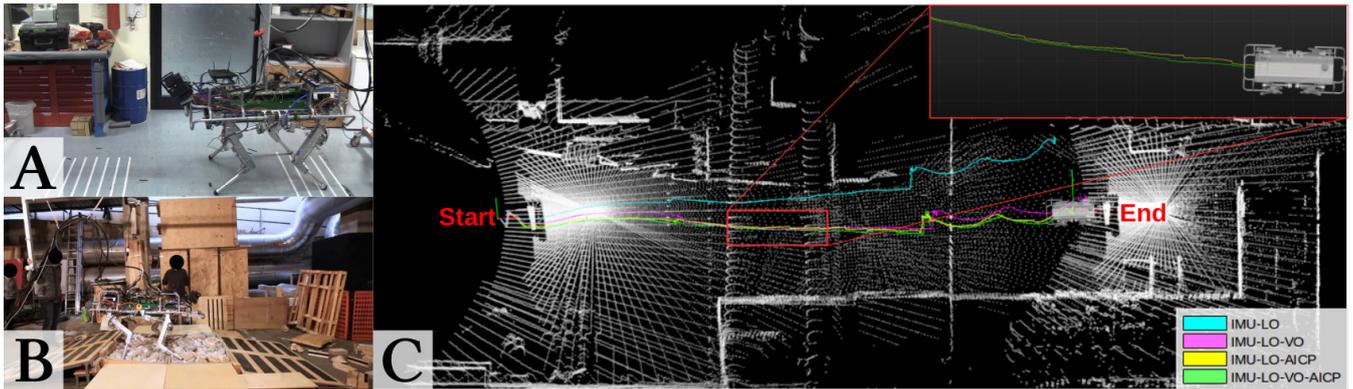
Figure 11 shows the estimated error over the course of Experiment 6a, recorded in the arena of Figure 9. The robot started from pose A, reached B and returned to A. The robot crawled for 40 m and paused to make 3 sharp turns. The experiment was at night and used the on-board LED lights.

During this run, the reference point cloud was updated 4 times. After 860 s, the state estimation performance had not significantly degraded, despite no specific global loop closure being computed.

##### 7.4.3.2 Trotting Gait

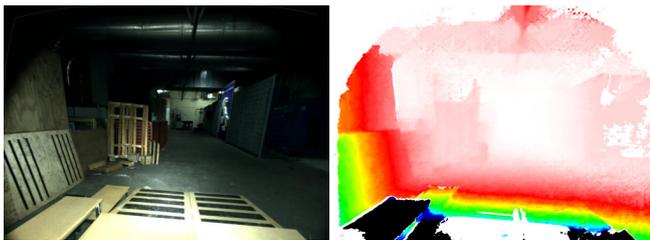
As mentioned previously, trotting is a more dynamic gait with a higher proprioceptive drift rate, which means that the VO could better contribute when combined with AICP. Empirically, this can be seen in the inset plot in Figure 9. In this case, the algorithm with VO produces a smoother trajectory (in green) than without (in yellow). This is important because the robot's base controller uses these estimates to close position and velocity control loops. Discontinuities in the velocity estimate could lead to undesired destabilizing foot forces and controller reactions.

In brief, for Experiments 6c–6e the integration of AICP allowed state estimation with an average 3D median translation error of approximately 4.9 cm. The integration of VO further reduced the median

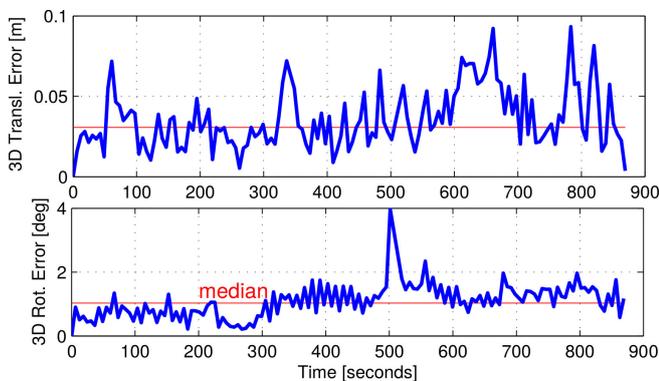


**Figure 9.** (A) Indoor repeatability tests. (B) Outdoor exploration tests in challenging scenarios. (C) Comparison between estimated trajectories of HyQ from Experiment 6d: IMU and Leg Odometry (cyan); IMU, Leg Odometry, VO (magenta); IMU, Leg Odometry (LO), AICP (yellow); IMU, LO, VO, AICP (green). Note the IMU-LO-VO-AICP trajectory is smoother than the combination without VO (inset).

translation error to 3.2 cm (Figure 11). The RPE over 10 m is in line with the indoor experiments.



**Figure 10.** Example of left camera image and depth image produced by the HyQ's stereo camera. This reflects the difficult lighting conditions and challenging structure of the test arena. The scene is illuminated with the sensor's on-board lights.



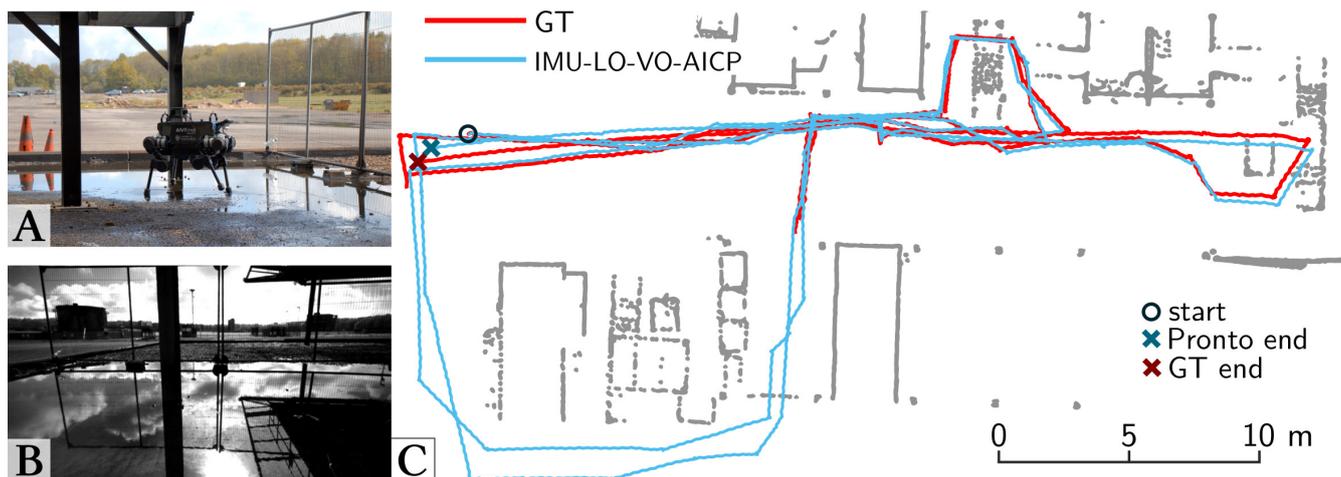
**Figure 11.** Estimated error of the state estimator used in Experiment 6a. The experiment involved the robot crawling for a total of 40 m.

## 7.5 ANYmal Experiments

The ANYmal dataset was collected at the Fire Service College, a 32.5 m × 42.5 m industrial oil rig facility used for firefighter training (Figure 12). The ground truth was collected with a laser tracking system, a Leica TS16, which tracked the robot's position with mm accuracy using a reflective prism on the robot.

The robot started from an open area and was commanded to trot at 0.3 m/s inside the facility, between metal containers and stairs, performing three loops before returning to the initial position, for a total of 240 m distance covered in 33 min. The dataset includes several extra challenges in addition to the ones of the previous section: 1) the area covered is much wider, therefore we had to trigger forced AICP reference updates on a regular basis (i.e., once every 0.5 m traveled); 2) the scene includes open areas where the robot looks at the horizon, where a very limited number of stereo features are available; 3) the scene contains reflections due to water puddles which confuse the visual feature tracking.

The different level of performance compared to previous tests is due to several factors related to the scenario used. In contrast with the previous experiments, the open space and the size of the area covered force triggering of frequent reference updates (more than 40 updates vs. 4 updates in Experiment 6 on HyQ). As no loop closures are performed, in this situation the LIDAR cannot completely eliminate the drift accumulated when new reference updates are triggered. In addition to that, the Velodyne scans are much sparser due to the wider scenario (only a few LIDAR rings are projected onto the ground), making hard to



**Figure 12.** (A) Experiment 7 with the ANYmal robot at the Fire Service College. (B) Onboard camera feed during the experiment. Note the visual odometry challenging conditions due to reflections in the water. (C) Estimated trajectory from Pronto with FOVIS and AICP active (blue) against ground truth (red). The ground truth was not available in the bottom area. Start and end location for both algorithms are highlighted with a circle and a cross, respectively.

constrain the robot position on the  $z$ -axis. We have partially compensated for this problem by augmenting the LIDAR data with a filtered output of a downward facing RealSense D435.

Despite these challenges, the system is able to effectively fuse all the sensors modalities, achieving an RPE of 34 cm over 10 m, which corresponds to 3.4% error. The contribution of the LIDAR localization is particularly evident on the  $z$ -axis where it significantly reduces the characteristic vertical drift caused by leg/ground compression while trotting. This allowed to reduce the RPE by 60% from the baseline algorithm with IMU and Leg Odometry only. After 240 m traveled, the pose estimate is less than 30 cm away from the ground truth (*cf.* the estimate on the  $xy$ -plane in Figure 12C).

## 8 DISCUSSION

In the previous section we demonstrated the ability of our system to overcome a variety of perception challenges, including: low light conditions, motion blur, reflections, dynamic motions, rough terrain. We also showed its versatility by supporting a variety of sensor modalities and four different legged robots.

The simple but effective integration of delayed signals into the time history described in Section 5.1 allowed us to integrate two different odometry sources (Visual and LIDAR) despite their significant delay and different frequencies.

A limitation of the current approach is the lack of measurement update triaging in case of disagreement between different exteroceptive sources. Currently, when an exteroceptive module does not report failure, its confidence on the measurement is only encoded by a fixed covariance matrix. A possible approach is to implement a mechanism that maps the error metrics specific to a module (e.g., VO reprojection error, ICP registration error) into a dynamically changing covariance matrix.

Alternatively, transitioning from loosely to tightly coupled approaches would allow joint optimization over all the measurements, making the estimation more robust against outlier updates. This is on going work.

## 9 CONCLUSION

We have presented a state estimation framework to perform sensor fusion of inertial, kinematic, visual sensing and LIDAR on legged robots, built upon a modular Extended Kalman Filter.

In particular, we indicated how our approach supports dynamic maneuvers and operation in sensor impoverished situations. The reliability of our approach was demonstrated with dynamic gaits and speeds up to 0.5 m/s. A particular technical achievement has been reliably closing the loop with this state estimator in dynamic gaits.

During experiments lasting over two hours, our system was demonstrated to be robust and

continuously accurate, with a RPE less than 35 cm over 10 m traveled for the most challenging scenario and 2–3 cm in smaller areas.

Our current filter marginalizes out previous state variables. In future work we will explore using windowed smoothing to incorporate measurements relative to previous filter states. We are also interested in extending the state with dynamic quantities such as CoM and linear/angular momenta similarly to Xinjilefu et al. (2014).

## CONFLICT OF INTEREST STATEMENT

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## AUTHOR CONTRIBUTIONS

MC wrote the main article, implemented the quadruped leg odometry and the overall filter architecture, and run the experiments on HyQ. MR performed LIDAR localization experiments on the ANYmal robot. SN implemented the LIDAR localization algorithm and performed the localization experiments on Atlas, Valkyrie, and HyQ. MF implemented the bipedal leg odometry, revised and approved the manuscript.

## FUNDING

This research has been conducted as part of the ANYmal research community. It was part funded by the EU H2020 Projects THING and MEMMO, the Innovate UK-funded ORCA Robotics Hub (EP/R026173/1) and a Royal Society University Research Fellowship (Fallon).

## ACKNOWLEDGMENTS

The authors would like to thank the Dynamic Robot Systems Group (University of Oxford), the MIT's DARPA Robotics Challenge team, the University of Edinburgh, and the Dynamic Legged Systems Lab (IIT) for support with experimental trials. The authors also thank the organizers of the RSS and ICRA conferences (2017 edition) for the opportunity to present the authors' prior work.

## REFERENCES

- Ahn, S., Yoon, S., Hyung, S., Kwak, N., and Roh, K. S. (2012). On-board odometry estimation for 3d vision-based slam of humanoid robot. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IEEE)*, 4006–4012

Barasuol, V., Buchli, J., Semini, C., Frigerio, M., De Pieri, E. R., and Caldwell, D. G. (2013). A Reactive Controller Framework for Quadrupedal Locomotion on Challenging Terrain. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)* (Karlsruhe, Germany)

Bloesch, M., Burri, M., Sommer, H., Siegwart, R., and Hutter, M. (2018). The two-state implicit filter recursive estimation for mobile robots. *IEEE Robotics and Automation Letters* 3, 573–580. doi:10.1109/LRA.2017.2776340

Bloesch, M., Gehring, C., Fankhauser, P., Hutter, M., Hoepflinger, M. A., and Siegwart, R. (2013). State estimation for legged robots on unstable and slippery terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 6058–6064. doi:10.1109/IROS.2013.6697236

Bloesch, M., Hutter, M., Hoepflinger, M., Leutenegger, S., Gehring, C., Remy, C. D., et al. (2012). State estimation for legged robots - consistent fusion of leg kinematics and IMU. In *Proc. of Robotics: Science and Systems (RSS)*

Bry, A., Bachrach, A., and Roy, N. (2012). State estimation for aggressive flight in GPS-denied environments using onboard sensing. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 1–8

Bry, A., Richter, C., Bachrach, A., and Roy, N. (2015). Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments. *The International Journal of Robotics Research* 34, 969–1002. doi:10.1177/0278364914558129

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., et al. (2016). The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research* doi: 10.1177/0278364915620033

Camurri, M., Fallon, M., Bazeille, S., Radulescu, A., Barasuol, V., Caldwell, D. G., et al. (2017). Probabilistic Contact Estimation and Impact Detection for State Estimation of Quadruped Robots. *IEEE Robotics and Automation Letters* 2, 1023–1030

Chilian, A., Hirschmüller, H., and Görner, M. (2011). Multi-sensor data fusion for robust pose estimation of a six-legged walking robot. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)* (San Francisco, California)

Diebel, J. (2006). Representing attitude: Euler angles, unit quaternions, and rotation vectors (Stanford University, California)

Fallon, M. F., Antone, M., Roy, N., and Teller, S. (2014). Drift-free humanoid state estimation fusing kinematic, inertial and LIDAR sensing.

- In *IEEE/RSJ Int. Conf. on Humanoid Robots* (Madrid, Spain)
- Featherstone, R. (2008). *Rigid Body Dynamics Algorithms* (Springer)
- Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2017a). On-Manifold Preintegration for Real-Time Visual-Inertial Odometry. *IEEE Transactions on Robotics* 33, 1–21. doi:10.1109/TRO.2016.2597321
- Forster, C., Zhang, Z., Gassner, M., Werlberger, M., and Scaramuzza, D. (2017b). Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics* 33, 249–265. doi:10.1109/TRO.2016.2623335
- Furgale, P., Rehder, J., and Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1280–1286. doi:10.1109/IROS.2013.6696514
- Hartley, R., Jadidi, M. G., Gan, L., Huang, J., Grizzle, J. W., and Eustice, R. M. (2018a). Hybrid contact preintegration for visual-inertial-contact state estimation using factor graphs. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 3783–3790
- Hartley, R., Mangelson, J., Gan, L., Ghaffari Jadidi, M., Walls, J. M., Eustice, R. M., et al. (2018b). Legged robot state-estimation through combined forward kinematic and preintegrated contact factors. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 4422–4429. doi:10.1109/ICRA.2018.8460748
- Hirschmüller, H. (2008). Stereo processing by semi-global matching and mutual information. *IEEE Trans. Pattern Anal. Machine Intell.* 30, 328–341
- Hornung, A., Wurm, K. M., and Bennewitz, M. (2010). Humanoid robot localization in complex indoor environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IEEE)*, 1690–1695
- Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., et al. (2011). Visual odometry and mapping for autonomous flight using an RGB-D camera. *Proceedings of the International Symposium on Robotics Research (ISRR)*
- Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C. D., Tsounis, V., et al. (2016). ANYmal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 38–44. doi:10.1109/IROS.2016.7758092
- Jenelten, F., Hwangbo, J., Tresoldi, F., Bellicoso, C. D., and Hutter, M. (2019). Dynamic locomotion on slippery ground. *IEEE Robotics and Automation Letters* 4, 4170–4176. doi:10.1109/LRA.2019.2931284
- Koolen, T., Bertrand, S., Thomas, G., de Boer, T., Wu, T., Smith, J., et al. (2016). Design of a momentum-based control framework and application to the humanoid robot Atlas. *International Journal of Humanoid Robotics* 13
- Lynen, S., Achtelik, M., Weiss, S., Chli, M., and Siegwart, R. (2013). A robust and modular multi-sensor fusion approach applied to MAV navigation. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (Tokyo, Japan)
- Ma, J., Bajracharya, M., Susca, S., Matthies, L., and Malchano, M. (2016). Real-time pose estimation of a dynamic quadruped in GPS-denied environments for 24-hour operation. *Intl. J. of Robotics Research* 35, 631–653
- Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robotics* 31, 1147–1163
- Nobili, S., Camurri, M., Barasuol, V., Focchi, M., Caldwell, D., Semini, C., et al. (2017a). Heterogeneous sensor fusion for accurate state estimation of dynamic legged robots. In *Robotics: Science and Systems (RSS)*
- Nobili, S., Scona, R., Caravagna, M., and Fallon, M. (2017b). Overlap-based ICP tuning for robust localization of a humanoid robot. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. 4721–4728
- Olson, E. (2010). A passive solution to the sensor synchronization problem. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Taipei, Taiwan)
- Pomerleau, F., Colas, F., Siegwart, R., and Magnenat, S. (2013). Comparing ICP variants on real-world data sets. *Autonomous Robots* 34, 133–148
- Qin, T., Li, P., and Shen, S. (2018). VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics* 34, 1004–1020. doi:10.1109/TRO.2018.2853729
- Radford, N. A., Strawser, P., Hambuchen, K., Mehling, J. S., Verdeyen, W. K., Donnan, A. S., et al. (2015). Valkyrie: NASA's first bipedal humanoid robot. *Journal of Field Robotics* 32, 397–419. doi:10.1002/rob.21560
- Reinke, A., Camurri, M., and Semini, C. (2019). A factor graph approach to multi-camera extrinsic calibration on legged robots. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*. 391–394. doi:10.1109/IRC.2019.00071
- Rotella, N., Bloesch, M., Righetti, L., and Schaal, S. (2014). State estimation for a humanoid robot. In *Proc. of the IEEE/RSJ International Conference*

- on *Intelligent Robots and Systems (IROS)*. 952–958
- Segal, A., Haehnel, D., and Thrun, S. (2009). Generalized-icp. In *Robotics: science and systems* (Seattle, WA), vol. 2, 435
- Semini, C., Tsagarakis, N. G., Guglielmino, E., Focchi, M., Cannella, F., and Caldwell, D. G. (2011). Design of HyQ – A hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 225, 831–849
- Shen, S., Mulgaonkar, Y., Michael, N., and Kumar, V. (2014). Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft mav. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)* (Hong Kong)
- Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A Benchmark for the Evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 573–580. doi:10.1109/IROS.2012.6385773
- Wim Meeussen (2010). *Coordinate Frames for Mobile Platforms*. Tech. Rep. REP – 105. <https://www.ros.org/reps/rep-0105.html>
- Wisth, D., Camurri, M., and Fallon, M. (2019). Preintegrated velocity bias estimation to overcome contact nonlinearities in legged robot odometry. *arXiv*, 1910.09875
- Wisth, D., Camurri, M., and Fallon, M. (2019). Robust legged robot state estimation using factor graph optimization. *IEEE Robotics and Automation Letters* 4, 4507–4514. doi:10.1109/LRA.2019.2933768
- Xinjilefu, X., Feng, S., and Atkeson, C. G. (2014). Dynamic state estimation using quadratic programming. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 989–994. doi:10.1109/IROS.2014.6942679
- Xinjilefu, X., Feng, S., and Atkeson, C. G. (2015). Center of mass estimator for humanoids and its application in modelling error compensation, fall detection and prevention. In *Humanoids, IEEE-RAS International Conference on Humanoid Robots*