CosySlam: investigating object-level SLAM for detecting locomotion surfaces

César Debeunne, Médéric Fourmy, Yann Labbé, Pierre-Alexandre Léziart, Guilhem Saurel, Joan Solà, Nicolas Mansard

CosySlam: investigating object-level SLAM for detecting locomotion surfaces

César Debeunne[‡†], Médéric Fourmy[†], Yann Labbé[△], Pierre-Alexandre Léziart[†], Guilhem Saurel[†], Joan Solà[†*] and Nicolas Mansard[†+]

***Abstract*— While blindfolded legged locomotion has demonstrated impressive capabilities in the last few years, further progresses are expected from using exteroceptive perception to better adapt the robot behavior to the available surfaces of contact. In this paper, we investigate whether mono cameras are suitable sensors for that aim. We propose to rely on object-level SLAM, fusing RGB images and inertial measurements, to simultaneously estimate the robot balance state (orientation in the gravity field and velocity), the robot position, and the location of candidate contact surfaces. We used CosyPose, a learning-based object pose estimator for which we propose an empirical uncertainty model, as the sole front-end of our visual inertial SLAM. We then combine it with inertial measurements which ideally complete the system observability, although extending the proposed approach would be straightforward (e.g. kinematic information about the contact, or a feature based visual front end). We demonstrate the interest of object-based SLAM on several locomotion sequences, by some absolute metrics and in comparison with other mono SLAM.**

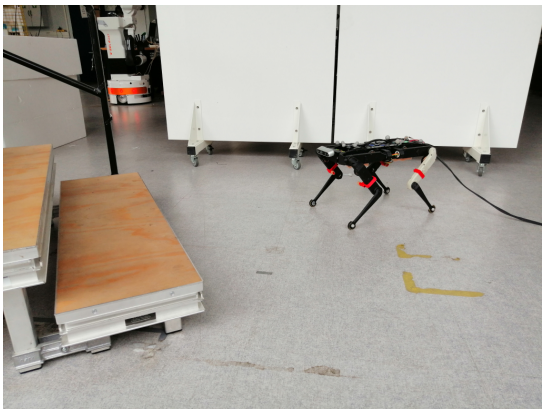

Fig. 1: Experimental setup: a RealSense D435i is mounted on the Solo robot which localizes itself with respect to stairs. A motion capture system provides ground truth of the robot pose.

I. Introduction

State estimation is a central aspect of the design of legged robots systems. Using estimates of the base velocity and gravity direction, modern locomotion controllers [1], [2] can achieve remarkable robustness without precise knowledge of their environment. In these cases, the state estimation relies simply on inertial, kinematic, and contact measurements to achieve balance [3]–[5]. However, to accomplish reactive contact planning over uneven surfaces [6], we need to go beyond a blindfolded system by using exteroceptive sensors.

Most of the current approaches exploit depth sensors which can be used to build local elevation maps of the robot surroundings [7] later processed to extract the contact surfaces needed by locomotion planners [8]. Current methods loosely couple the elevation mapping with the estimation of the robot state. Other approaches strive to extend reconstruction capabilities by using 3D voxels [9], dense surfaces [10], or meshes representations of the environment [11].

[‡] ISAE-Supaero, Toulouse
[†] LAAS-CNRS, Université de Toulouse, France
[△] Inria, École normale supérieure, CNRS, PSL Research University, Paris, France
[*] Institut de Robòtica i Informàtica Industrial, Barcelona
[+] Artificial and Natural Intelligence Toulouse Institute, Toulouse, France

This work has been supported by the MEMMO European Union project within the H2020 Program under Grant Agreement No. 780684, by the HPC resources from GENCI-IDRIS (Grant AD011011181R1), the Spanish Ministry of Science, Innovation, and Universities project EB-SLAM (DPI2017-89564-P), by the EU H2020 project GAUSS (DPI2017-89564-P) and by the Spanish State Research Agency through the Maria de Maeztu Seal of Excellence to IRI MDM-2016-0656. The experiments have been performed on the cluster DeepRAP, graciously provided by Patrick Danès (LAAS-CNRS), who we thank respectfully.

On the opposite, recent legged estimators, based on the tight coupling of exteroceptive inertial SLAM system with leg kinematics, have been shown to provide accurate and robust source odometry [12], [13]. The use of sparse features in these approaches is however not intended for contact surfaces extractions. Yet they tend to show the importance of tightly coupling all locomotion estimators, to which we also intend to contribute.

In indoor environments, a localization system may benefit from the presence of known objects. SLAM++ [14] showed that object-level semantic SLAM could be achieved in office environments using depth sensors and scanned models of objects such as chairs and tables. More recent works on the matter rely mainly on deep-learning-based object segmentation to detect objects in RGB images [15]. This information is used along with depth measurements to perform semantic SLAM with static [16] or moving [17]–[19] objects. Using solely monocular cameras, object SLAM can also be achieved without object shape priors by directly integrating segmentation bounding box in classical feature-based SLAM frameworks [20], [21]. Aside from the camera trajectory and object poses, object shapes may also be optimized using a differentiable rendering engine [22].

Object-level SLAM has not been explored in the context of legged robot locomotion yet. In this paper, we propose to build a visual-inertial object-level SLAM by merging an object pose estimator based on shape priors with pre-integrated inertial measurements. We rely on the open-source pose estimator CosyPose [23], for which we first propose

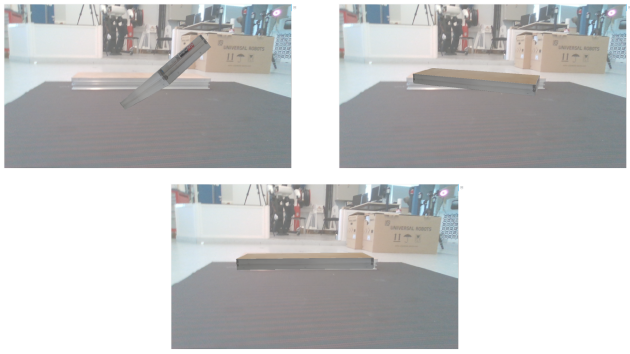

Fig. 2: Progressive convergence of a stair step estimated pose over successive iterations of CosyPose

in Section II-A a uncertainty model and an outlier rejection procedure. We then set up a factor graph encompassing CosyPose information and IMU measurements in Section III. A large part of the paper is devoted to the experimental analysis of the performance in the locomotion context, on quadruped scenarios where the system can accurately estimate the robot state and the location of some stair steps either aligned or randomly spaced.

II. COSY SLAM

The visual front end used for this project includes image object detection and object pose estimation. These functionalities are provided by CosyPose [23], a framework that reaches state-of-the-art performances for 6D object pose estimation [24]. In the original paper, a single-view pose estimator and a multi-view algorithm based on object level Bundle Adjustment were introduced. In our context, only the single-view module is used, object tracking and multi-view reconstruction being handled by our factor graph estimation framework WOLF [25].

A. CosyPose

CosyPose takes as input a single image I and a set of 3D models, each associated with an object label l. The camera C is assumed to be calibrated. A set of object detections is performed using the object detector Mask-RCNN [15]. Then, each 2D candidate detection in view I is identified by an index α and associated with an object candidate O_α and its 6D pose ${}^{C}\mathbf{T}_{O_\alpha} \in SE(3)$ with respect to the camera C.

The single-view pose estimation procedure of CosyPose is an improvement over the one proposed in DeepIm [26]. The general idea is to iteratively refine an object pose estimation using a render-and-compare approach (Fig. 2). The network takes as input a cropped image obtained from the detection bounding box and a rendered image based on the current object pose solution ${}^{C}\mathbf{T}_{O_\alpha,k-1}$ at iteration $k-1$. It outputs a transformation update ${}^{O_\alpha,k-1}\mathbf{T}_{O_\alpha,k}$ that refines the pose estimation. In practice, two neural networks with the same structure are trained independently: one for a coarse pose estimation, *i.e.*, the first iteration of the iterative process, and one for the refinement of the pose. The coarse network gives the first transformation ${}^{C}\mathbf{T}_{O_\alpha,0}$, and the prediction of the pose of the object is obtained by composing the N successive transformations:

$$ {}^{C}\mathbf{T}_{O_\alpha} = {}^{C}\mathbf{T}_{O_\alpha,0} \prod_{k=1}^{N} {}^{O_\alpha,k-1}\mathbf{T}_{O_\alpha,k} \tag{1} $$

CosyPose reuses the neural network architecture of DeepIm with a new backbone for feature extraction: EfficientNet [27] with a spatial average pooling layer added after it. Among other technical improvements in the architecture and training, a special care is given to object symmetries that are taken into account during training thanks to the *symmetric distance*. Each 3D model l is associated with a set of symmetries $S(l)$, that is the set of transformations that leave the aspect of the object unchanged:

$$ S(l) = \{\mathbf{S} \in SE(3) | \forall \mathbf{T} \in SE(3), \mathcal{R}(l,\mathbf{T}) = \mathcal{R}(l,\mathbf{TS})\} \tag{2} $$

where $\mathcal{R}(l,\mathbf{T})$ is the rendered image of the object l captured in pose M. Given a set of symmetries $S(l)$, the symmetric distance D_l which measures the distance between two 6D poses represented by transformation $\mathbf{T}_1$ and $\mathbf{T}_2$ is defined. Given an object l associated to a set $\mathcal{X}_l$ of 3D points $\mathbf{x} \in \mathcal{X}_l$, we have:

$$ D_l(\mathbf{T}_1,\mathbf{T}_2) = \min_{\mathbf{S}\in S(l)} \frac{1}{|\mathcal{X}_l|} \sum_{\mathbf{x}\in\mathcal{X}_l} ||\mathbf{T}_1\mathbf{S}\mathbf{x} - \mathbf{T}_2\mathbf{x}||^2 \tag{3} $$

Equation (3) measures the average distance between the points of the object model transformed by $\mathbf{T}_1$ and $\mathbf{T}_2$ according to the symmetry that best aligns the transformed points.

B. Covariance model

Probabilistic fusion with other sensors requires a metric covariance of the output of CosyPose, which is not available out of the box. Two main methods for neural network predictions uncertainty quantification include Monte Carlo (MC) dropout [28] or Bayesian Neural Network (BNN) [29]. Using BNNs would require to change the architecture of CosyPose and to retrain it. MC dropout would require several forward passes through the network at evaluation time, which would be computationally expensive for a SLAM system.

In this work, we strive to compute this covariance without changing the architecture of the network and at an affordable cost. We propose to make an empirical error model based on polynomial regression in order to compute the covariance matrix. The idea is to parametrize the average error on each $\mathfrak{se}(3)$ component returned by CosyPose. We conduct an empirical study on several video sequences that explore the variations of the parameter set for several object types. The error is computed by comparing the $SE(3)$ transformations between the camera C and an object O returned by CosyPose ${}^{C}\mathbf{T}_{O}$ with the same transformation given by a motion capture system. We then use the error predictions of our parametric model as a proxy to the true 6 covariances during model fitting. A different model is fit for each object type due to their diversity of shapes, sizes and textures.

The parameters to compute the error need to represent as much as possible the error sources of CosyPose. To cover the error due to the configuration of the object in space, we need to include the 3D coordinates of the camera in the object frame. We also want to take into account some invariance that can occur by rotating around the object if it is textureless for example. For this reason, we choose spherical coordinates of the camera origin with respect to the object frame to parametrize the model. Another source of error can be the occlusion of the object in the scene, as well as the motion blur, or any inherent noise in the image. This can affect the quality of the detection and of the pose estimation. Having an idea of the quality of the object detection informs us on the quality of the object representation that may be occluded or blurry. The Mask-RCNN object detector returns a confidence score s for each detection that we will include in our model. This score is the output of the final softmax layer of the detector, which is designed to represent a probability distribution.

To sum up, our model is parametrized by four values: r - distance, φ - azimuth, θ - elevation, s - Mask-RCNN softmax output. We can then compute the error of CosyPose with respect to the motion capture data:

$$\mathbf{e} = [\mathbf{e}_t, \mathbf{e}_\theta] \triangleq \left[{}^{O}\widehat{\mathbf{p}}_C - {}^{O}\widetilde{\mathbf{p}}_C,\ \mathrm{Log}\left({}^{O}\widehat{\mathbf{R}}_C^{-1}\, {}^{O}\widetilde{\mathbf{R}}_C \right) \right] \quad (4)$$

where $\widehat{\cdot}$ and $\widetilde{\cdot}$ denote quantities obtained from the motion capture system and CosyPose respectively. Log denotes the logarithm application mapping elements of SO(3) to the $\mathbb{R}^3$ representation of its Lie algebra $\mathfrak{so}(3)$.

We want to find a polynomial function $f(r, \varphi, \theta, s) \in \mathbb{R}^6$ that returns the error given the set of training data $\{X, E\}$. For each object, we capture a set of video sequences and we compute the error with the motion capture data for each measurement. We then perform polynomial regression with a pipeline in Scikit Learn [30]. A simple linear regression leads to a high root-mean-square error (RMSE) on a test dataset. Over degree 3, the model overfits and the high curvature of the polynomial returned high error values outside of the training data range. Thus, a degree 2 polynomial regression seemed to offer a good compromise. Quantitative results are given in the experimental validation section (see Fig. 6 for a few examples of fitted polynomials).

In the SLAM system, we use this error model to compute the covariance matrix. To do so, we make the assumption that the cross-correlations between the 6 DOF are negligible, building a diagonal covariance matrix:

$$\Sigma^V = diag(\mathbf{e}^T\mathbf{e}) \in \mathbb{R}^{6\times 6} \quad (5)$$

C. Data association and Outlier rejection

A key part of our SLAM system is the association of landmarks with the rejection of erroneous pose estimates. First of all, each object is associated with a label α so that a detection can only match a landmark with the same label. Then, the position of the robot is propagated by integrating the IMU measurements with the current estimated biases.

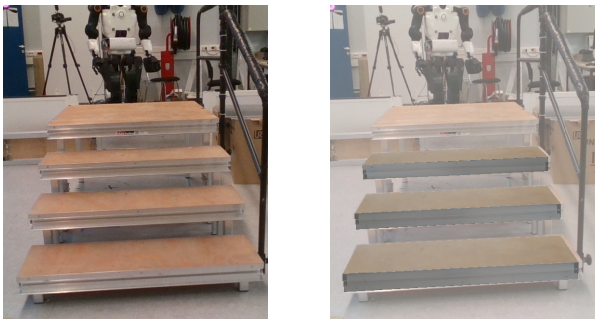

Fig. 3: On the left, a picture of our climbing module at LAAS and on the right the 3D model of the stair projected according to CosyPose measurements on the same picture

Thus we can then have the pose of a detection in the world frame at each keyframe. We check if this pose is similar to the one of a landmark with the same label with a threshold on the distance between the pose in $SE(3)$. If a detection does not match any landmark then a new one is created.

CosyPose can return poses of objects that are not included in the scene because of false detections, of Mask-RCNN, or wrong pose estimations. To handle these outlier detections, each landmark is associated with a score c that corresponds to its repeatability over time:

$$c = \frac{n_f}{\Delta t} \quad (6)$$

Δt is the time since the landmark initialization and n_f is the number of factors associated to it. The lowest scores are filtered with a threshold determined empirically and the associated landmarks are removed from the map.

D. Retraining with stairs

In order to produce a realistic SLAM scenario in the context of walking legged robots, we retrained CosyPose with staircases present in our lab. We made a textured mesh of a stair step used in our experimental platform. This textured mesh was used both for training and using the trained model. The generation of photorealistic synthetic data was handled by BlenderProc [31]. The render and compare loop uses PyBullet [32].

We generate 10,000 synthetic images that are labeled with the pose ground truth by design of the scene in Blender. We retrain the three modalities of CosyPose: the Mask-RCNN detector, the coarse pose estimator and the pose refiner. We slightly tune the training parameters used for the object from the BOP challenge as the scale is different: the stair is $1m \times 0.3m \times 0.07m$ whereas a T-LESS object is never wider than $10cm$. Thus, we generate training data on $10m \times 10m$ scenes, and we increase the noise used to train the refiner.

An illustration of the performances of CosyPose on a set of 3 stair steps is given on Fig. 3: the pose of each step is here independently estimated which leads to local accuracy but global inconsistency.

III. Factor graph formulation

In our factor graph SLAM, the problem is represented as a bipartite graph where nodes represent either variables

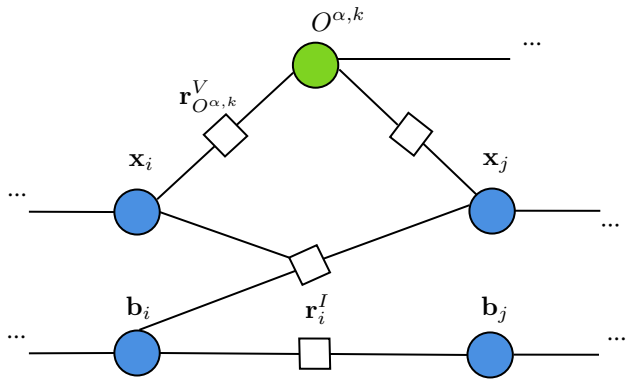

Fig. 4: The problem factor graph involves state blocks corresponding to keyframes $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{v}_i, \mathbf{R}_i)$, biases $\mathbf{b}_i$ and objects $O^{\alpha,k}$. IMU factors relate consecutive keyframes and the IMU bias whose drift is represented on the lower branch. Visual factors relate landmarks poses and keyframes from which the landmark has been observed.

of the problem or geometrical constraints between sets of variables: the *factors*. The state $\mathbf{x}$ is modeled as a multivariate Gaussian distribution that includes robot poses and velocities at a given keyframe i, $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{v}_i, \mathbf{R}_i)$, the sensor bias $\mathbf{b}_i$ and the object poses ${}^W\mathbf{T}_{O^{\alpha,k}} \in SE(3)$ in the world frame W. Thus, we can formulate finding the Maximum A Posteriori as the following non-linear-least-squares problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_i ||\mathbf{r}^I_i(\mathbf{x})||^2_{\Sigma^I_i} + \sum_j ||\mathbf{r}^V(\mathbf{x})||^2_{\Sigma^V_j} \quad (7)$$

with $\{\mathbf{r}^I, \Sigma^I\}$ and $\{\mathbf{r}^V, \Sigma^V\}$ being the residuals and covariances of the inertial and visual factors respectively. A typical fraction of our factor graph is represented on Fig. 4.

A. Visual factor

As seen previously, CosyPose returns the pose of an object labeled α expressed in the camera frame. At a keyframe time i, it is noted ${}^{C_i}\widetilde{\mathbf{T}}_{O^\alpha} \in SE(3)$. In our SLAM framework, an object can be considered as a landmark whose pose in the world frame can be simply obtained by applying the composition chain ${}^W\widetilde{\mathbf{T}}_{O^{\alpha,k}} = {}^W\widetilde{\mathbf{T}}_B\, {}^B\widetilde{\mathbf{T}}_{C_i}\, {}^{C_i}\widetilde{\mathbf{T}}_{O^{\alpha,k}}$, as seen in Fig. 5. We indexed the object with k as several objects of the same label α can be present in the scene. The frame B is the IMU frame that we consider being the robot base frame of our problem. The transformation between the robot frame and the camera frame ${}^B\widetilde{\mathbf{T}}_{C_i}$ is assumed to be known.

The factor's residual $\mathbf{r}^V(\mathbf{x}_i, O^{\alpha,k}) \in \mathbb{R}^6$ is defined as the logarithmic difference between the expected pose of the object ${}^{C_i}\widetilde{\mathbf{T}}_{O^{\alpha,k}} = {}^B\widetilde{\mathbf{T}}^{-1}_{C_i}\, {}^W\widetilde{\mathbf{T}}^{-1}_B\, {}^W\widetilde{\mathbf{T}}_{O^{\alpha,k}}$ and CosyPose measurement ${}^{C_i}\widetilde{\mathbf{T}}_{O^\alpha}$:

$$\mathbf{r}^V(\mathbf{x}_i, O^{\alpha,k}) = \mathrm{Log}\left({}^{C_i}\widetilde{\mathbf{T}}^{-1}_{O^{\alpha,k}}\, {}^B\widetilde{\mathbf{T}}^{-1}_{C_i}\, {}^W\widetilde{\mathbf{T}}^{-1}_B\, {}^W\widetilde{\mathbf{T}}_{O^{\alpha,k}}\right) \quad (8)$$

This residual is weighted in (7) by its covariance $\Sigma^V_{\alpha,k}$ that is computed with the empirical error model detailed in the previous section.

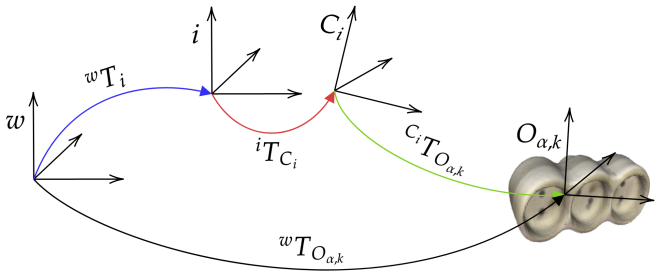

Fig. 5: Kinematic chain of the visual observation of a landmark.

CosyPose measurements may be erroneous if the single view iterative pose estimation converges to a local minimum, which often happen for objects presenting strong symmetries. Ideally, we would like to take this phenomenon into account in the estimator, as is done during the training of CosyPose. However, this would require to model a multi-modal behaviour which is not handled by MAP based estimator. Instead, we remove outlier factors and landmarks by the procedure described in Section II-C. This method might also be complemented by the use of RANSAC [33] or M-estimators [34].

B. IMU pre-integration factor

As it is shown in [35], [36], IMU measurements can be pre-integrated between two keyframes to avoid re-integration at each iteration of the solver. We obtain *delta quantities* $\Delta = [\Delta\mathbf{p}, \Delta\mathbf{v}, \Delta\mathbf{R}]$ that are independent of the initial conditions for position, orientation and velocity and depend only on IMU data and bias. The effect of changes in the bias estimates is linearized so that the deltas can be corrected using precomputed Jacobians. This delta pre-integration theory is implemented in WOLF, our factor graph framework, and every detail about Jacobians and deltas computation is given in [37].

We can then exhibit the residuals for the IMU delta factors between keyframes i and j. It requires the states estimates $\mathbf{x}_i$ and $\mathbf{x}_j$, the current bias estimates $\mathbf{b}_i$, the bias used during pre-integration $\tilde{\mathbf{b}}_i$, the pre-integrated Jacobian $J^{\Delta_{ij}}_{\mathbf{b}}$ and the pre-integrated delta $\widetilde{\Delta}_{ij}$. A corrected delta Δ_{ij} is computed with the bias and its Jacobian with a linearized update:

$$\Delta_{ij} = \widetilde{\Delta}_{ij} \oplus J^{\Delta_{ij}}_{\mathbf{b}}(\mathbf{b}_i - \tilde{\mathbf{b}}_i) \quad (9)$$

We compute the predicted delta $\widehat{\Delta}_{ij}$ from the state estimates using the increments introduced in [36]. Finally, the residuals are given by:

$$\mathbf{r}^I(\mathbf{x}_i, \mathbf{x}_j, \mathbf{b}_i) = \begin{bmatrix} \Delta\mathbf{p}_{ij} - \widehat{\Delta}\mathbf{p}_{ij} \\ \Delta\mathbf{v}_{ij} - \widehat{\Delta}\mathbf{v}_{ij} \\ \mathrm{Log}(\widehat{\Delta}\,\mathbf{R}^{-1}_{ij}\Delta\mathbf{R}_{ij}) \end{bmatrix} \in \mathbb{R}^9 \quad (10)$$

IV. Experimental validation

We have produced a dataset of trajectories in the robotic experimental arena at LAAS-CNRS in Toulouse. The robot environment was augmented with objects of the datasets (YCBV [38] and T-LESS [39]) that were used to train

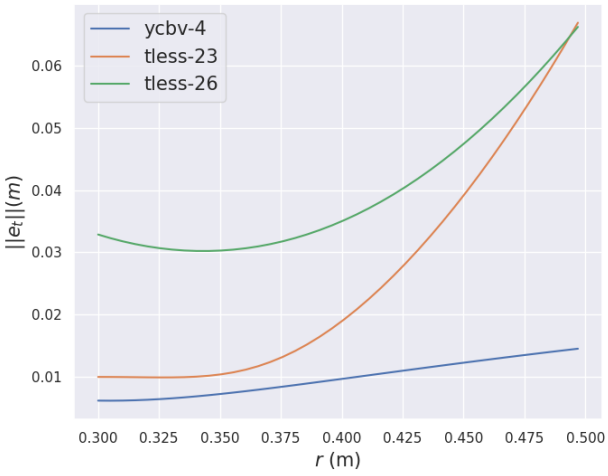

Fig. 6: The norm of the translation error returned by the models of three different objects with respect to r, the distance between the camera and the object. The other parameters θ, ϕ and s are fixed to the average values of our training data.

CosyPose[1]. Each of our trajectories is composed of three sequences:

- A sequence of RGB images captured at 30 Hz
- A sequence of IMU measurements captured at 200 Hz
- A sequence of motion capture (MoCap) measurements at 200 Hz used as ground truth

We recorded two types of datasets: one for the uncertainty models and one for the SLAM. For the uncertainty models, reflective MoCap markers were attached to the object to obtain ground truth of their pose. For the SLAM, only the camera was tracked. We used the monocular RGB camera and the Bosh BMI085 IMU of an Intel RealSense L515 Camera for handheld trajectories. The Intel RealSense d435i was used with the same modalities for the experiments on the quadruped robot Solo [40] as shown in Fig. 1. The extrinsic calibration between the IMU and the camera was provided by Intel and the delays observed between IMU and Camera measurements were negligible. Links to our code and datasets are publicly available at https://gepettoweb.laas.fr/articles/cosyslam.html.

A. Empirical covariance

As explained in Section II-B, we have trained empirical models to evaluate the covariance of the estimation of CosyPose. To validate these models, we propose to exhibit a few intuitive observations and a quantitative statistical analysis. One of the parameter involved in our model is the absolute distance between the camera and the object, noted r. Our trained models show an expected behavior regarding this parameter: the global error increases when the camera moves away from the object. Fig. 6 sheds light on these

[1]YCBV is composed of daily life objects, many of them presenting rich textures such as inscriptions. T-LESS features small Czech electric devices, whose symmetries and lack of texture makes them a challenging benchmark.

TABLE I: A quantitative evaluation of our models, these values are computed on test samples that were not used for training.

Object	R^2	RMSE ang. err. (°)	RMSE trans. err. (cm)
YCBV-4	0.55	5,1	0.6
T-LESS-23	0.5	11.7	1.5
T-LESS-26	0.68	22	0.6

phenomena and gives an explicit comparison between the models of different objects. The error of the object from the YCBV dataset seems more stable and smaller than the one of the objects from the T-LESS dataset. This can be explained by the texture of the object and the absence of symmetries: T-LESS objects are known to be more challenging for pose estimation and this is confirmed by our model.

A more quantitative evaluation can be deduced from Table I. The translation error seems to be captured pretty well, as the RMSE is around the centimeter. However, the angular error seems a little less predictable, especially for T-LESS objects which orientation estimation can suffer from an important measurement noise due to the lack of textures. The $R^2 \in [0, 1]$ score is the coefficient of determination and is often used to evaluate statistical models. Its interpretation is subject to debate and cannot conclude to a "good" or a "bad" model. However, a score higher than 0 demonstrates that our model is more accurate than a simple average model.

Like our pose model, our covariance model is driven by data. This leads to practical limitations to this method. This empirical model is adapted to our experimental setup, but can't be generalized easily without fitting the model again. However we tried to minimize the feature number of the model as well as its complexity to alleviate this phenomenon. Moreover, one of our future research directions is to investigate more general models that could be trained in simulation, taking advantage of domain randomization.

B. Object level VI-SLAM

In order to validate the performances of the fusion of CosyPose estimates and inertial measurements, we evaluated two scenarios with the camera held by hand and T-LESS objects in the scene. The first one is a slow circular trajectory, in which the system is in a comfort zone. The second one is a highly dynamic scenario with a lot of motion that can blur some frames and where the camera loses sight of objects for more extend period of times. Moreover, T-LESS objects being the most difficult objects for pose estimation with CosyPose, they may return many outliers and noisy measurements. This is therefore a challenging dataset to test the robustness of our algorithm. Keyframes are selected at 10 Hz, only if objects are detected in the images.

It is interesting to analyse the gains brought by the IMU fusion. The most evident observation is that the output trajectory is smoother, which gives more consistency to the result (Fig. 7). But we can notice that the Mean Translation Error (MTE) is also reduced (Table II). Indeed, the motion model is more precise thanks to IMU data. This makes the outlier rejection more efficient in the VI CosySLAM: the vision only CosySLAM relies on a zero velocity assumption.

TABLE II: Comparison of CosySLAM with ORBSLAM3 (as a purely visual SLAM system) and VINS-Mono based on the Mean Translation Error metric in cm. For visual inertial system, it is computed by aligning the estimation with motion capture data using a rigid transformation. For purely visual system, a similarity transformation is used to obtain the scale factor.

			Visual only		Visual Inertial	
Scenario	Distance (m)	Duration (s)	CosySLAM	ORBSLAM3	CosySLAM	VINS-Mono
Hand circular	3.7	23.7	3.8	3.3	3.1	3.7
Hand dynamic	3.5	17.8	7.8	6.0	1.5	-
Solo approach	1.3	18.7	2.6	2.3	2.0	3.1
Solo modules	1.3	15.5	1.9	1.7	1.4	2.6
Stairs on floor	10.5	43.5	13.7	15.8	6.5	13.0

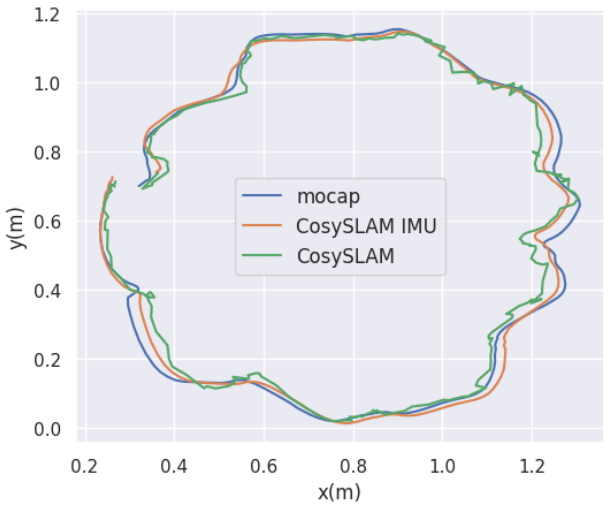

Fig. 7: Comparison between the MoCap, the output of CosySLAM with visual factors only and the output of CosySLAM with IMU fusion on the circular trajectory.

We also ran two open source mono SLAM system, ORB-SLAM3 [41] and VINS-MONO [42], on these sequences to compare our performances. On these two scenarios, our visual-inertial SLAM is the more accurate according to the MTE. VINS-MONO was not able to initialize in the dynamic scenario in which IMU fusion offers significant improvement.

C. Localization and Mapping of stairs by Solo

With our retrained model we were able to perform SLAM in our experimental area, without augmenting it with fake objects. We recorded video sequences including stairs with a camera fixed on a Solo robot (Fig. 8). A stair has three discrete symmetries that are hard to handle for an object pose estimator and the images provided by Solo were noisy because of the walk. These scenarios are challenging for our SLAM system, but it maps successfully the stairs and the error on the position of the base of Solo remains reasonable (Table II).

Stairs objects allowed us to make a longer scenario as they are larger than T-LESS object. We recorded a longer trajectory with a hand held camera with stair cases on the floor. It was a particularly challenging scenarios as the images contained not many features apart from the stairs. VINS-MONO and ORBSLAM3 performed poorly on this sequence whereas our system was able to localize itself precisely using high level features.

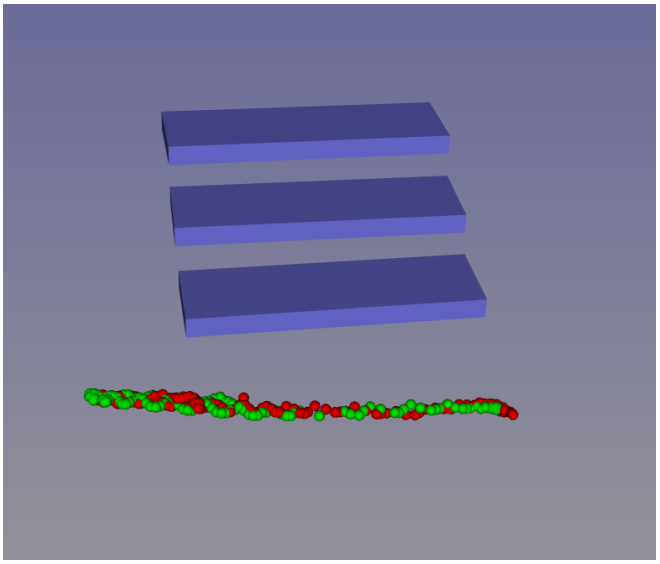

Fig. 8: This trajectory was recorded on Solo walking along a climbing module made of three stairs using Leziart's [2] controller. The green dots represent the trajectory of Solo provided by the MoCap, and the red dots the one produced by our visual-inertial SLAM. The blue rectangles represent the map of the SLAM made of stairs.

D. Implementation details and performance

We used a CPU node (Core i5 2.3GHz) for optimizing the factor graph under Wolf, and a GPU node (NVIDIA Quadro RTX 6000) to run CosyPose. Our implementation is a python-based prototype, yet the performance of each solver provides a proper scaling of the SLAM efficiency for real-time purpose. Timings of CosyPose are reported on Table III. CosyPose is used both for detecting new objects and for tracking known landmarks. Following classical parallel tracking and mapping separation, detection of new landmarks (that are not yet in the map) can be done asynchronously and requires about 0.2 sec per detected object. As explained in Sec. II.A, we can discard the detection stage of CosyPose and reduce the number of iterations by warm-starting it with the prediction from the SLAM for tracking landmarks that are already in the map. In that case, each iteration of CosyPose requires 0.06 sec and must be run synchronously, enabling it to produce visual factor at 10Hz (we used a maximum

TABLE III: Mean timings (in seconds) of the different elements of the CosyPose framework on one frame. Total timing include the timing of Mask-RCNN detection, 2 iterations of the pose refinement as used in the experiments (which includes rendering and an Efficientnet forward pass) as well as surrounding data manipulation.

	Detections	Rendering	Neural network	Total
Hand circular	0.038	0.041	0.021	0.208
Hand dynamic	0.038	0.041	022	0.211
Solo approach	0.032	0.023	0.017	0.138
Solo modules	0.035	0.028	0.020	0.167
Stairs on floor	0.033	0.027	0.02	0.165

TABLE IV: MTE in cm when a feature is discarded with a probability p, on the last column the solver was not able to converge

p	0.0	0.3	0.5	0.6	0.65	0.7
Hand circular	3.1	3.2	3.2	3.2	3.2	-

of 2 iterations). CosyPose can process several landmarks in parallel although we have not fully investigated the resulting performances.

E. Robustness

We push our method to its limit on the hand-held circular trajectory, by artificially preventing visual factors to be added in the graph, in order to evaluate the robustness to potential errors of the pose estimator. Firstly, we randomly discarded CosyPose detection with a probability p to check the robustness to a lack of repetition in feature detection. On Table IV, we reported that the performances are not affected by a loss of feature until a feature is rejected with a probability of 0.7. This empirically validates the ability to deal with punctual loss of information. Then we simulated blind navigation by cutting the image flow during $\Delta_t = 5$ s in the middle of the sequence. During the black-out, localization only relies on inertial measurements. While open-loop integration may diverge quickly, the proper estimation of the IMU intrinsic parameter by the visual SLAM provides a good estimates in the meantime. When the image flow is starting again, the system is able to relocalize itself properly in this case, as shown in Fig. 9.

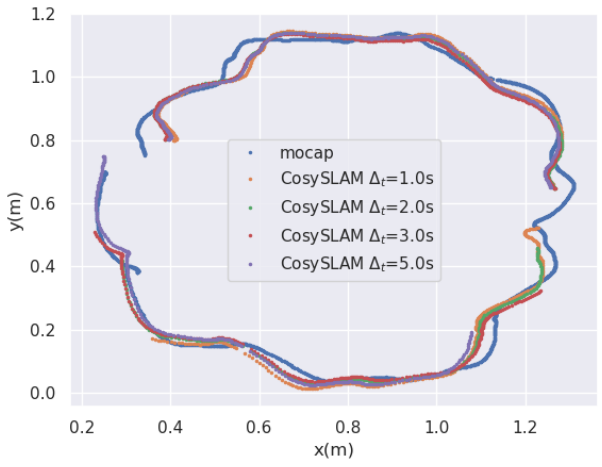

Fig. 9: Comparison between the MoCap and CosySLAM visual inertial on the hand-held circular scenario with blind navigation during various Δ_t, starting at 10s (around 1.2 m on the x axis). We purposely hide the state estimate to highlight the parts where no CosyPose measurements are integrated.

V. Conclusion and Discussion

In this paper, we have proposed a complete experimental setup to evaluate the interest of object-level SLAM to tightly-coupled legged robot state estimation, navigation-level localization and localization of surrounding candidate contact surfaces. To that end, we have proposed a visual-inertial SLAM implementation. The visual front-end relies on a state-of-the-art pose estimator, CosyPose, which processes separated frames to evaluate the object-camera relative transformation. As reported in the experiments, the estimates are accurate yet noisy, and may be locally inconsistent due to object symmetries. We have proposed a confidence model, which is needed to insert the measurement in a factor-graph estimator. The inertial measurements are handled through pre-integrated factors, enabling the observation of the robot velocity and orientation in the gravity field, along with IMU biases in real-time. We have then reported a complete experimental evaluation of the proposed SLAM system, on new visual-inertial datasets tailored to investigate locomotion scenarios. We have reported the validity and interest of the approach, which provides accurate and robust estimates of both the robot state and the location of candidate contact surfaces in real-time.

The main limitation of the proposed method is due to the need to off-line train the pose estimator on known objects, which, for the time being, limits its application to laboratory scenarios. Yet several recent works [43], [44] show the possibility to extend such a pose estimator to classes of objects (*i.e.* estimating any kind of stairs, or tables, or stair steps, without needing to fine-tune the training for each new instance of the class), which would open the possibility to extend our system to classical indoor environments. We have limited our study to only two types of factors, yet it immediately extends to other factors, for example merging a classical visual front end based on 2D landmarks [41], or to kinematic information coming from the robot legs [45]. We have reported a prototype implementation that cannot yet run in real-time onboard our quadruped robot, but we will now work on a complete C++ implementation that we intend to use for closing the loop and controlling the robot locomotion.

References

[1] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[2] P. Léziart, T. Flayols, F. Grimminger, N. Mansard, and P. Souères, "Implementation of a reactive walking controller for the new open-hardware quadruped solo-12," in *IEEE International Conference on Robotics and Automation*, 2021.

[3] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots-consistent fusion of leg kinematics and imu," *Robotics*, vol. 17, 2013.

[4] N. Rotella, M. Bloesch, L. Righetti, and S. Schaal, "State estimation for a humanoid robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[5] T. Flayols, A. Del Prete, P. Wensing, A. Mifsud, M. Benallegue, and O. Stasse, "Experimental evaluation of simple estimators for humanoid robots," in *IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017.

[6] J. Carpentier, A. Del Prete, S. Tonneau, T. Flayols, F. Forget, A. Mifsud, K. Giraud, D. Atchuthan, P. Fernbach, R. Budhiraja *et al.*, "Multi-contact locomotion of legged robots in complex environments–the loco3d project," in *RSS Workshop on Challenges in Dynamic Legged Locomotion*, 2017.

[7] P. Fankhauser, M. Bloesch, and M. Hutter, "Probabilistic terrain mapping for mobile robots with uncertain localization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

[8] D. Kim, D. Carballo, J. Di Carlo, B. Katz, G. Bledt, B. Lim, and S. Kim, "Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[9] A. W. Winkler, C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[10] M. F. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," in *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015.

[11] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control," *IEEE Transactions on Robotics*, vol. 36, no. 6, pp. 1635–1648, 2020.

[12] D. Wisth, M. Camurri, and M. Fallon, "Robust legged robot state estimation using factor graph optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4507–4514, 2019.

[13] ——, "Vilens: Visual, inertial, lidar, and leg odometry for all-terrain legged robots," *arXiv preprint arXiv:2107.07243*, 2021.

[14] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. Kelly, and A. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1352–1359, 2013.

[15] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017.

[16] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger, "Fusion++: Volumetric object-level slam," in *international conference on 3D vision (3DV)*, 2018.

[17] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2018.

[18] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "Mid-fusion: Octree-based object-level multi-instance dynamic slam," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[19] M. Strecke and J. Stuckler, "Em-fusion: Dynamic object-level slam with probabilistic data association," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[20] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 925–938, 2019.

[21] Y. Wu, Y. Zhang, D. Zhu, Y. Feng, S. Coleman, and D. Kerr, "Eao-slam: Monocular semi-dense object slam based on ensemble data association," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[22] E. Sucar, K. Wada, and A. Davison, "Nodeslam: Neural object descriptors for multi-view shape reconstruction," in *International Conference on 3D Vision (3DV)*, 2020.

[23] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *European Conference on Computer Vision*, 2020.

[24] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, "BOP challenge 2020 on 6D object localization," *European Conference on Computer Vision Workshops (ECCVW)*, 2020.

[25] J. Sola, J. Vallve, J. Casals, J. Deray, M. Fourmy, D. Atchuthan, A. Corominas-Murtra, and J. Andrade-Cetto, "Wolf: A modular estimation framework for robotics based on factor graphs," *IEEE Robotics and Automation Letters*, 2022.

[26] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "Deepim: Deep iterative matching for 6d pose estimation," *International Journal of Computer Vision*, vol. 128, no. 3, 2019.

[27] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019.

[28] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *International conference on machine learning*, 2016.

[29] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on bayesian neural networks–a tutorial for deep learning users," *arXiv preprint arXiv:2007.06823*, 2020.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi, and H. Katam, "Blenderproc," *arXiv preprint arXiv:1911.01911*, 2019.

[32] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[33] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[34] K. MacTavish and T. D. Barfoot, "At all costs: A comparison of robust cost functions for camera correspondence outliers," *12th Conference on Computer and Robot Vision*, pp. 62–69, 2015.

[35] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, 2011.

[36] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," *Robotics: Science and Systems (RSS) conference*, 2015.

[37] D. Atchuthan, A. Santamaria-Navarro, N. Mansard, O. Stasse, and J. Solà, "Odometry based on auto-calibrating inertial measurement unit attached to the feet," in *European Control Conference (ECC)*, 2018.

[38] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[39] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

[40] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.

[41] C. Campos, R. Elvira, J. J. G. Rodriguez, J. M. Montiel, and J. D. Tardos, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, 2021.

[42] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[43] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation," in *Conference on robot learning (CoRL)*, 2020.

[44] M. Durner, W. Boerdijk, M. Sundermeyer, W. Friedl, Z.-C. Marton, and R. Triebel, "Unknown object segmentation from stereo images," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[45] M. Fourmy, T. Flayols, N. Mansard, and J. Solà, "Contact forces pre-integration for the whole body estimation of legged robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.