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: tracking contact features using visual-inertial object-level SLAM for locomotion

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

***Abstract*— A legged robot is equipped with several sensors observing different classes of information, in order to provide various estimates on its states and its environment. While state estimation and mapping in this domain have traditionally been investigated through multiple local filters, recent progresses have been made toward tightly-coupled estimation. Multiple observations are then merged into an a-posteriori maximum estimating several quantities that otherwise were separately estimated. With this paper, our goal is to move one step further, by leveraging on object-based simultaneous localization and mapping. We use an object pose estimator to localize the relative placement of the robot with respect to large elements of the environments, e.g. stair steps. These measurements are merged with other typical observations of legged robots, e.g. inertial measurements, to provide an estimation of the robot state (position, orientation and velocity of the basis) along with an accurate estimation of the environment pieces. It then provides a consistent estimation of these two quantities, which is an important property as both would be needed to control the robot locomotion. We provide a complete implementation of this idea with the object tracker CosyPose, which we trained on our environment and for which we provide a covariance model, and with the SLAM engine Wolf used as a visual-inertial estimator on the quadruped robot Solo.**

I. Introduction

Navigation of legged robots using on board exteroceptive sensors has gained a lot of traction in recent years due to their progressive deployment for industrial applications [1]. For repeated travels, the map-less teach and repeat methods [2], [3] avoid the need of a metric and globally coherent localization by benefiting from the knowledge of a human operator. This works very well for applications in which such a supervision is available, and a global map is not. However, most of the time, in industrial environments, a map is obtainable as a CAD model or 3D scans, in which important assets can be labelled.

Many representations of the environment are possible depending on the needs of the system. In [4] a prior map defined as a LIDAR pointcloud is used in a bayesian filter to localize a humanoid robot and perform online foot planning.

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

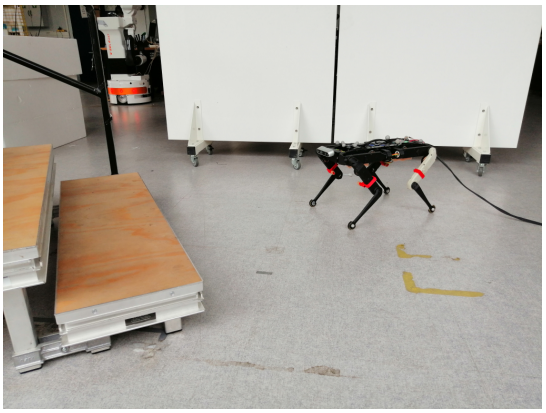

Fig. 1: Experimental setup: a Realsense D435i is mounted on the Solo robot which localizes itself with respect to stairs. A motion capture system provides ground truth of the robot pose.

Ref. [5] takes as an input an external odometry source to produce efficient robot-centric elevation maps while [6] builds a global height map to navigate through cluttered environments. Other approaches [7] rely on a tight fusion between proprioceptive and exteroceptive sensors in order to make the odometry more robust.

These approaches build metric maps that do not usually leverage the presence of known assets in the scene. In [8], the authors develop one of the first object-level SLAM algorithm from a depth sensor. Using a voting process based on point cloud descriptors, a simultaneous recognition and pose estimation of known objects was performed and included as factors in a graph optimization estimator. The method benefited from an active search of the objects in the scene, the detection being done in the SLAM loop. Aside from the robot trajectory and poses of objects, [9] also proposes to optimize the object shapes using a differentiable rendering engine. In such approaches, objects need to be detected, classified, and their relative pose with respect to the camera has to be integrated into the estimator. On the other end, a work like [10] uses a semi-dense mono camera SLAM algorithm to produce a scale-ambiguous feature map. Then, a descriptor-based multi-view object proposal is performed as a post processing step.

Huge progress to the field of object detection was brought about by the advent of CNN frameworks such as [11]. As a result, it seems that the domain of object scene reconstruction is now dominated by methods relying only on RGB cameras, for both object detection and pose retrieval. A major example

is the framework CosyPose [12]. This approach mixes a new state-of-the-art single-view pose estimation algorithm with a multi-view algorithm using RANSAC and Bundle Adjustment. It achieved first in most of the 2020 BOP challenge categories [13]. This system obtains precision in the order of centimeters on real objects whose 3D model is known. Its performances make it a good candidate as a direct 6D pose sensor to perform multi-sensor fusion. In the context of legged robots, this is very useful to localize the robot with respect to objects it needs to interact with, such as objects to manipulate or stairs to climb. While these models are not yet able to be generalized to classes of objects, rapid progress are expected in this direction. Their performances are already very interesting to help with the locomotion in known scenes.

When considering merging a tracker such as CosyPose with other sensor modalities, an important aspect is to predict covariance representing the level of confidence in the tracker estimate. Such data uncertainty awareness has been shown to be crucial to the robustness of SLAM systems involving neural networks subsystems such as [14], while [8] claimed to compute a covariance matrix approximated as the inverse of the Iterative Closest Point output. The methods targeted for deep learning applications are harder to implement, especially if the goal is to use an off-the-shelf pose estimation neural network, as it is the case for this paper. For instance, Bayesian Neural Networks [15] need to be trained explicitly for uncertainty prediction while Monte Carlo (MC) Dropout [16] requires multiple forward passes at run-time.

In this paper we present a practical implementation of a SLAM system based on the design of an off-the-shelf deep learning object pose estimation algorithm [12]. In order to be able to integrate these measurements with other sensors, we propose a noise model based on empirical data. We also detail practical tricks to circumvent outliers in the network output. Experimental validations were conducted with a visual-inertial system, both handheld and mounted on a quadruped robot. Finally we fine tuned the pre-trained models to perform stairs localization.

II. COSY SLAM

The front end of the SLAM system designed for this project includes object detection and object pose estimation. This function is provided by CosyPose [12], a deep learning based 6D tracker that reaches state-of-the-art performances for 6D object pose estimation. In the original paper, a single-view pose estimator and a multi-view algorithm were introduced. In our context, only the single-view module is used, object tracking being handled by the SLAM framework.

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 [11]. 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.

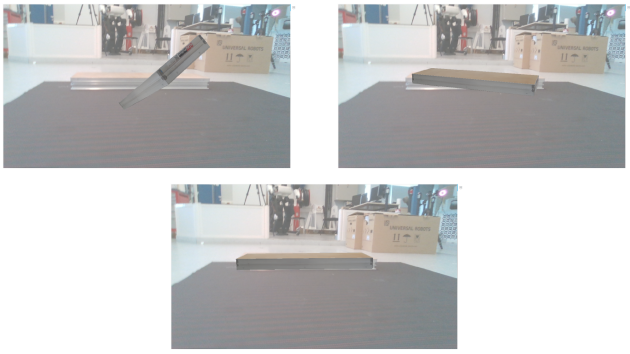

Fig. 2: Progressive convergence of a stair step estimated pose over successive iterations of CosyPose

The single-view pose estimation procedure of CosyPose is an improvement over the one proposed in DeepIm [17]. The general idea is to iteratively use the same neural network to converge to the most precise object pose (Fig 2). It takes as input the cropped image of the bounding box of the detected object and a rendered image based on the current object pose solution ${}^{C}\mathbf{T}_{O_\alpha,k-1}$ at iteration $k-1$. It returns a transformation update ${}^{O_\alpha,k-1}\mathbf{T}_{O_\alpha,k}$ that brings the rendered image closer to the cropped image. 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 *i.e.* the last iterations. 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: Efficient-Net [18] with a spatial average pooling layer added after it. Then, it disables the optical flow sub network during the training. A new rotation parametrization is used for the loss function which was introduced in [19] and which has been shown to bring more stability during training. Then, the focal length of the cropped images is recomputed during training to fit the virtual camera of these images. Finally, the object symmetries 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 $S(l)$, we define the symmetric distance D_l which measures the distance between two 6D poses represented by transformation $\mathbf{T}_1$ and $\mathbf{T}_2$. Given an object l associated to a set $\mathcal{X}_l$ of 3D points $\mathbf{x} \in \mathcal{X}_l$,

we have:

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

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. In practice, for continuous symmetries that are rotations around an axis (like for instance for a textureless cylinder), $S(l)$ is discretized using 64 angles.

B. Covariance model

CosyPose does not provide an evaluation of its uncertainty. The two main families of solutions available to estimate uncertainties of neural network predictions consist in MC dropout [16] or Bayesian Neural Network (BNN) [15]. Using BNNs would require to change the architecture of CosyPose and to retrain it. MC dropout would require several forward passes through the network for each iterations, which would be computationally expensive.

We need to compute the 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:

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

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 [20]. 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).

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. 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:

$$c = \frac{n_f}{\Delta t} \quad (5)$$

Δt is the time since the landmark initialisation 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 [21]. The render and compare loop uses PyBullet [22].

We generate 10.000 synthetic images that are labeled with the pose ground truth by design of the scene in Blender. We

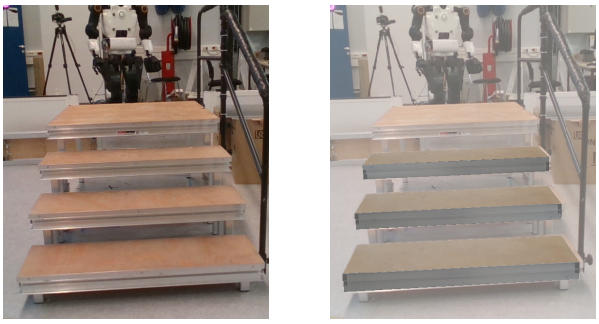

Fig. 3: On the left, a picture of our climbing module at LAAS and on the right the 3D model of the stair projected according to CosyPose measurements on the same picture

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 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:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_i ||\mathbf{r}_i^I(\mathbf{x})||^2_{\Sigma_i^I} + \sum_j ||\mathbf{r}^V(\mathbf{x})||^2_{\Sigma_j^V} \quad (6)$$

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.

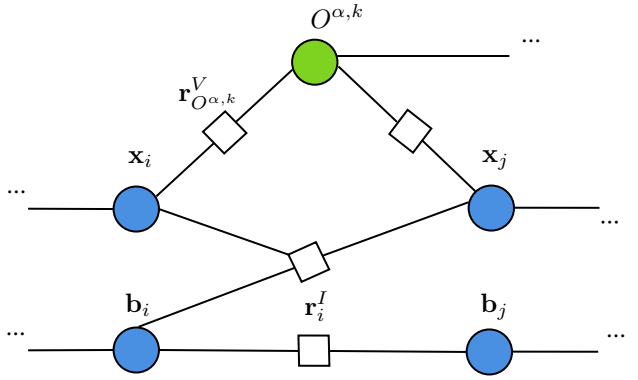

Fig. 4: This graph involves state blocks corresponding to keyframes $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{v}_i, \mathbf{R}_i)$, biases $\mathbf{b}_i$ and objects $O^{\alpha,k}$. IMU factors relate consecutive keyframes and the IMU bias whose drift is represented on the lower branch. Visual factors relate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

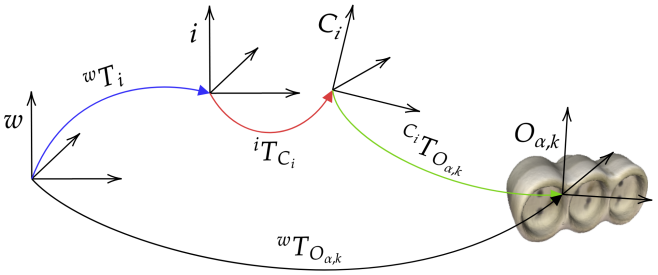

Fig. 5: Kinematic chain of the visual observation of a landmark

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 ${}^{C_i}\widetilde{\mathbf{T}}_{O^{\alpha,k}} = {}^{B}\widetilde{\mathbf{T}}_{C_i}^{-1}\,{}^{W}\widetilde{\mathbf{T}}_{B}^{-1}\,{}^{W}\widetilde{\mathbf{T}}_{O^{\alpha,k}}$ and CosyPose measurement ${}^{C_i}\widetilde{\mathbf{T}}_{O^\alpha}$:

$$\mathbf{r}^V(\mathbf{x}_i, O^{\alpha,k}) = \mathrm{Log}\left({}^{C_i}\widetilde{\mathbf{T}}_{O^{\alpha,k}}^{-1}\,{}^{B}\widetilde{\mathbf{T}}_{C_i}^{-1}\,{}^{W}\widetilde{\mathbf{T}}_{B}^{-1}\,{}^{W}\widetilde{\mathbf{T}}_{O^{\alpha,k}}\right) \quad (7)$$

This residual is weighted in (6) by its covariance $\Sigma^V_{\alpha,k}$ that is computed with the empirical error model detailed in the previous section.

B. IMU pre-integration factor

As it is shown in [23], [24], 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 [25].

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 $J^{\Delta_{i,j}}_{\mathbf{b}}$ and the

pre-integrated delta $\widetilde{\Delta}_{i,j}$. A corrected delta $\Delta_{i,j}$ is computed with the bias and its Jacobian with a linearized update:

$$\Delta_{i,j} = \widetilde{\Delta}_{i,j} \oplus J_{\mathbf{b}}^{\Delta_{i,j}}(\mathbf{b}_i - \tilde{\mathbf{b}}_i) \tag{8}$$

We compute the predicted delta $\widehat{\Delta}_{i,j}$ from the state estimates using the increments introduced in [24]. Finally, the residuals are given by:

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

IV. Experimental validation

We have produced datasets in the robotic experimental arena at LAAS-CNRS in Toulouse. This is a 3D environment about $10m \times 5m$ made of flat floors, stairs and beams. The robot environment was augmented with objects of the datasets that were used to train CosyPose. Each dataset 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 [26] 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. Our datasets are publicly available at https://homepages.laas.fr/mfourmy/icra22_cosyslam.

A. Empirical covariance

As explained in Sec. II, 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 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

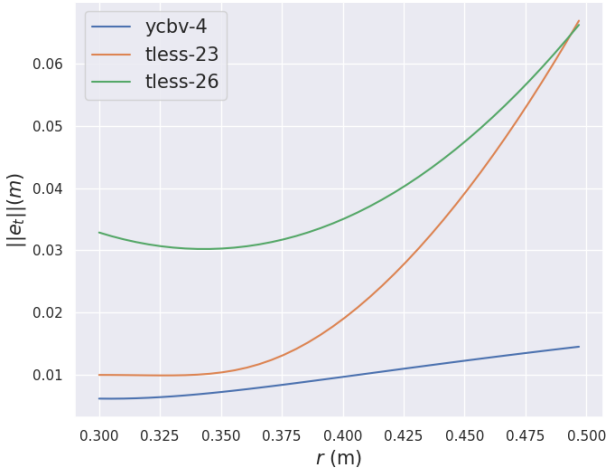

Fig. 6: The norm of the translation error returned by the models of three different objects with respect to r, the distance between the camera and the object. The other parameters θ, ϕ and s are fixed to the average values of our training data.

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

	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

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.

B. Object level VI-SLAM

In order to validate the performances of the fusion of CosyPose estimates and inertial measurements, we evaluated three scenarios with the camera held by hand and T-LESS objects[1] in the scene. The first one is a short and slow trajectory, i.e. an ideal scenario. The second one is a slow but long trajectory, to validate the consistency of our system over time. The last one is a highly dynamic scenario with a lot of motion that can blur some frames and lose 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

[1]T-LESS is one of the dataset for which CosyPose is trained by default and whose object can be bought in Czech Republic [27]. It features several small electric devices, whose symmetry at lack of texture make them an interesting benchmark for realistic scenarios.

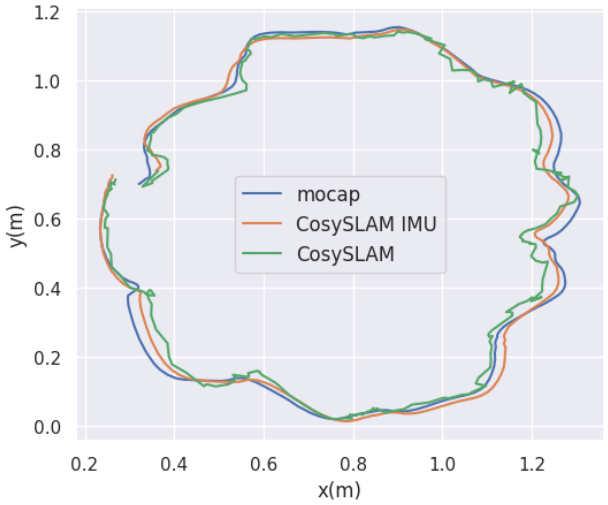

Fig. 7: Comparison between the MoCap, the output of CosySLAM with visual factors only and the output of CosySLAM with IMU fusion on the circular trajectory.

TABLE II: Datasets description and results of the hand held videos

Scenario	Length(m)	Duration(s)	MTE[1](cm)	STE[2](cm)
V-only - Circular	3.7	23.7	3.8	1.6
V-only - Short	2.5	12	3.8	2.4
V-only - Dynamic	3.5	17.8	7.8	4.0
V-IMU - Circular	3.7	23.7	1.9	0.7
V-IMU - Short	2.5	12	1.9	0.5
V-IMU - Dynamic	3.5	17.8	1.7	1.2

[1] Mean translation error
[2] Standard deviation of translation error

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 MTE is also reduced (Table II). Indeed, the motion model is more precise thanks to IMU data. This makes the outlier rejection more efficient than the visual only CosySLAM which makes a zero velocity assumption between keyframes.

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 III).

V. Conclusion and Discussion

We have proposed using a visual-inertial object-based SLAM to simultaneously estimate the robot state and the location of large objects in the environment, such as stair steps. The main idea was to demonstrate that a technology efficient for object manipulation and scene reconstruction could be applied to another field of robotics. The object pose estimator provides an informative yet noisy estimate of the location of objects in the scene, as well as an interesting alternative to dense mapping systems. The inertial measurements accurately smooth out the SLAM estimates and increase the bandwidth of the perception system. The a-posteriori maximization then leverages the best of the two sensor modalities to provide fast and accurate estimates of both the robot state and the environment mapping. To this end, we have proposed an original model to estimate the uncertainty of the object pose estimator. We also reported the first re-training of the CNN of the CosyPose object pose estimator, which the reader would hopefully find useful if also aiming at using it. The empirical study of CosyPose gave us a better understanding of its behavior and strengthened our idea that it was suitable for SLAM application. We have shown that our SLAM system performs well enough on two different object scales and platforms. By mapping and tracking the contact surface on the stair scenarios, our work can contribute to help Solo climbing a set of stairs.

TABLE III: Datasets description and results of the videos taken on Solo

Scenario	Length(m)	Duration(s)	MTE(cm)	STE(cm)
V-IMU - Approach	1.3	18.7	2.0	0.9
V-IMU - Module	1.3	15.5	2.4	1.5

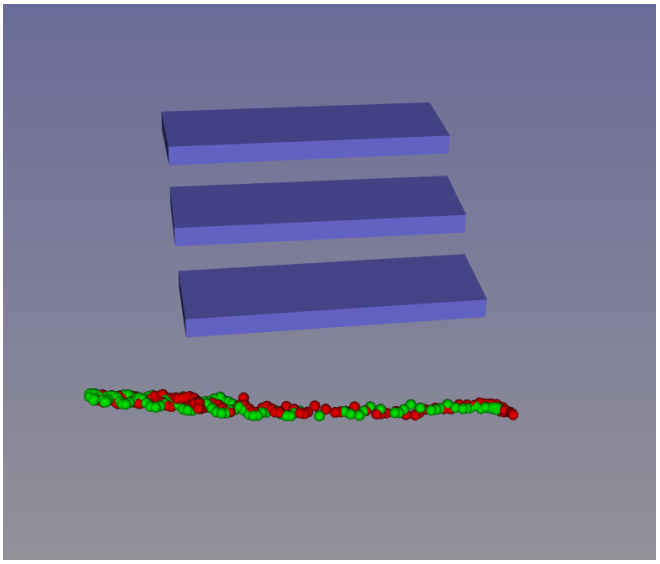

Fig. 8: This trajectory was recorded on Solo walking along a climbing module made of three stairs using [28]'s controller. The green dots represent the trajectory of Solo provided by the MoCap, and the red dots the one produced by our visual-inertial SLAM. The blue rectangles represent the map of the SLAM made of stairs.

References

[1] C. D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Günther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world

applications for legged robots," *Journal of Field Robotics*, vol. 35, no. 8, pp. 1311–1326, 2018.

[2] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *Journal of Field Robotics*, vol. 27, no. 5, pp. 534–560, 2010.

[3] M. Mattamala, M. Ramezani, M. Camurri, and M. Fallon, "Learning camera performance models for active multi-camera visual teach and repeat," *arXiv preprint arXiv:2103.14070*, 2021.

[4] M. F. Fallon, M. Antone, N. Roy, and S. Teller, "Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 112–119.

[5] 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.

[6] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2464–2470.

[7] D. Wisth, M. Camurri, and M. Fallon, "Vilens: Visual, inertial, lidar, and leg odometry for all-terrain legged robots," *arXiv preprint arXiv:2107.07243*, 2021.

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

[9] E. Sucar, K. Wada, and A. Davison, "Nodeslam: Neural object descriptors for multi-view shape reconstruction," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 949–958.

[10] S. Pillai and J. Leonard, "Monocular slam supported object recognition," *arXiv preprint arXiv:1506.01732*, 2015.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[12] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," 2020.

[13] T. Hodaň, M. Sundermeyer, B. Drost, Y. Labbé, E. Brachmann, F. Michel, C. Rother, and J. Matas, "Bop challenge 2020 on 6d object localization," in *European Conference on Computer Vision*. Springer, 2020, pp. 577–594.

[14] N. Yang, L. v. Stumberg, R. Wang, and D. Cremers, "D3vo: Deep depth, deep pose and deep uncertainty for monocular visual odometry," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1281–1292.

[15] 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.

[16] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[17] 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, p. 657–678, Nov 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01250-9

[18] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[19] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," 2020.

[20] 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.

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

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

[23] 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.

[24] 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*, 01 2015.

[25] D. Atchuthan, A. Santamaria, N. Mansard, O. Stasse, and J. Solà, "Odometry based on auto-calibrating inertial measurement unit attached to the feet," 06 2018, pp. 3031–3037.

[26] 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, pp. 3650–3657, 2020.

[27] 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 *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017, pp. 880–888.

[28] P.-A. 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 *2021 IEEE International Conference on Robotics and Automation - ICRA*, Xi'an, China, May 2021. [Online]. Available: https://hal.laas.fr/hal-03052451