2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

VINSEval: Evaluation Framework for Unified Testing of Consistency and Robustness of Visual-Inertial Navigation System Algorithms

Alessandro Fornasier[1], Martin Scheiber[1], Alexander Hardt-Stremayr[1], Roland Jung[1] and Stephan Weiss[1]

***Abstract*— The research community presented significant advances in many different Visual-Inertial Navigation System (VINS) algorithms to localize mobile robots or hand-held devices in a 3D environment. While authors of the algorithms often do compare to, at that time, existing competing approaches, their comparison methods, rigor, depth, and repeatability at later points in time have a large spread. Further, with existing simulators and photo-realistic frameworks, the user is not able to easily test the sensitivity of the algorithm under examination with respect to specific environmental conditions and sensor specifications. Rather, tests often include unwillingly many polluting effects falsifying the analysis and interpretations. In addition, edge cases and corresponding failure modes often remain undiscovered due to the limited breadth of the test sequences. Our unified evaluation framework allows, in a fully automated fashion, a reproducible analysis of different VINS methods with respect to specific environmental and sensor parameters. The analyses per parameter are done over a multitude of test sets to obtain both statistically valid results and an average over other, potentially polluting effects with respect to the one parameter under test to mitigate biased interpretations. The automated performance results per method over all tested parameters are then summarized in unified *radar charts* for a fair comparison across authors and institutions.**

SOFTWARE & VIDEO

The open-sourced VINSEval framework is made available via https://github.com/aau-cns/vins_eval. A demonstration video of VINSEval is made available on https://youtu.be/KuA3nibxWok.

I. INTRODUCTION

Data-driven algorithms for autonomous robotics gained significant attention over the last years, enabling a paradigm shift in state estimation for mobile robotic applications. This trend allowed the robotic research community to design and develop *Visual-Inertial SLAM (VI-SLAM)*, *Visual-Inertial Odometry (VIO)* algorithms, or, in general, *Visual-Inertial Navigation System (VINS)* algorithms able to reach high performance in terms of accuracy and efficiency. To do so, simulation and synthetic data have been one of the fundamental tools for engineers and researchers during the design and development of such algorithms. They allow fast prototyping, safe, and inexpensive testing without dealing with real-world experiments and hardware issues in the early development stages. Further, simulations provide high repeatability of data and the precise control of various parameters. Despite the progress made to let state-of-the-art estimation algorithms reach high performance in terms of accuracy with respect to commonly used error metrics (i.e., Absolute Trajectory Error (ATE), and Relative Trajectory Error (RTE)), current VINS solutions still lack in robustness and consistency. In addition, as more and more such methods are presented by the community, the lack of a unified comparison and benchmarking tool starts to become an important issue.

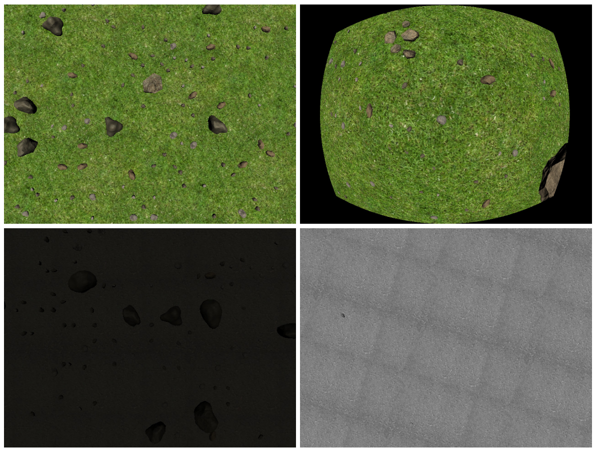

Fig. 1. Views of different rendered scenes. Top row, left to right: a scene with a grass ground texture and stones providing informative visual features, with no camera distortion and with fisheye lens distortion. Bottom row, left to right: a scene with a cement self-similar ground texture and no informative visual features, with low illumination, and with optimal illumination.

Motivated by the shortcomings in robustness and consistency in VINS methods, and in particular the upcoming era of research dubbed *robust-perception-age* [1], in this paper, we present *VINSEval*: a fully automated photo-realistic visual and inertial data generation, simulation, and estimator evaluation framework for fast VINS development, improvement, and unified comparison. VINSEval has two core capabilities: (i) For researchers to speed up the prototyping and development of consistent and robust VINS algorithms through the capability of generating setups and data with very specific parameters and parameter changes, and (ii) for both researchers and end-user engineers to evaluate and compare the performance of VINS algorithms in terms of consistency and robustness in a unified, fully automated fashion over a large set of parameter sweeps. VINSEval is not only general in that sense, but it also allows the generation of very specific edge-case scenarios where VINS can be tested in. Each scenario has unique visual characteristics and requirements, and we believe providing a framework for understanding the constraints therein is critical when assessing performance and guiding subsequent development. State-of-the-art VINS benchmarking approaches tend to ignore differences between individual scenarios leading to solutions that only partially address end-user needs. They rather focus on system-level

[1]Control of Networked Systems, University of Klagenfurt, Austria {firstname.lastname}@ieee.org

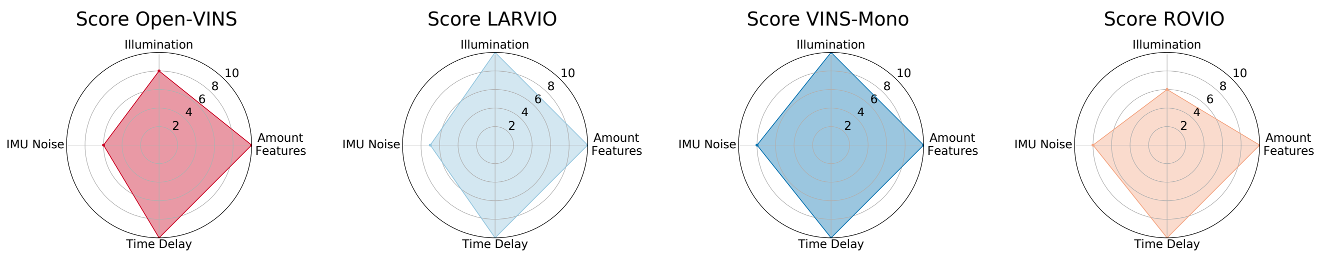

Fig. 2. Robustness overall score and Breaking Point (BP) of the VINS algorithms under examination with increasing difficulty levels for each of the considered environmental and/or sensor parameters. The BP per parameter is visually defined as the level next to the the corner of the polygon.

performance and accuracy but overlook that none of these error metrics, when uncontextualized, indicate how well a VINS algorithm could perform on a given specific scenario.

The multiple key contributions of the presented work are: ***Unified statistical evaluation framework***: To the best of our knowledge, VINSEval is the first work that provides a framework to evaluate with statistical relevance the consistency and robustness of VINS algorithms in a fully automated fashion over a multitude of parameters and parameter ranges. ***Sample evaluation***: We demonstrate how VINSEval can statistically compare the consistency and robustness of four state-of-the-art algorithms when applying parameter sweeps over (i) amount of features seen, (ii) illumination conditions, (iii) IMU noise values, and (iv) sensor time delay. The results are automatically summarized in radar-charts in Fig. 2 for quick information access with minimal user effort.
Extensibility: VINSEval is an easily extendable framework. This is true for the photo-realistic scene, different parameter sweeps, and different evaluation parameters that can further be included. All such extensions are directly included in the fully automated evaluation process enabling VINSEval as a useful tool for VINS evaluation in various different scenarios.
Flexibility and modularity: All the modules of VINSEval are modular and flexible. Indeed the data provided as input to the core of VINSEval can be either synthetically generated or recorded from a real platform. The rendering module then allows automated changes of the rendered scene and flexibility to manipulate rendering parameters, UAV parameters, and sensors noise specifications online.

II. RELATED WORK

With particular regard to UAVs, Hector Quadrotor [2] and RotorS [3] are Gazebo simulators that allow the user to simulate different types of multi-rotor UAVs with specific sensors such as IMU, LIDAR and camera. These environments do not provide photo-realistically rendered camera images – an issue addressed by AirSim [4]. This work proposes a software-in-the-loop simulation with popular flight controllers such as PX4 and ArduPilot and hardware-in-loop for physically and visually realistic simulations. Recently [5] and [6] published their work FlightGoggles and Flightmare, respectively, which are both ROS-based open-source photo-realistic simulation framework for MAVs. They mainly differ from AirSim by having fewer rigid structures and an integrated physics engine for dynamics simulation. InteriorNet [7] proposes an end-to-end pipeline for an RGB-D-inertial benchmark in large-scale interior scene understanding and mapping. The trajectories, the scenes, and rendering parameters have a high level of customizability. However, the simulator lacks flexibility as it is limited to a fixed set of indoor scenes and CAD models of indoor furniture. The authors in [8], [9], [10] presented SlamBench (currently at version 3) which is a dataset- and sensor-agnostic framework for qualitative, quantitative, and easily reproducible evaluation for accuracy and computation time of SLAM systems with plug-and-play algorithm support. SlamBench incorporates a wide range of error metrics, datasets, and evaluation tools for different SLAM algorithms. However, its flexibility is limited since it does not provide a way to generate individual data for a specific scenario. Its focus is on the evaluation of computational complexity and estimation accuracy, not on robustness and consistency. Regarding robustness, the authors in [11] proposed a characterization of state-of-the-art SLAM benchmarks and methods by comparisons of the performance of different SLAM algorithms. They use publicly available datasets, at both real-time speed and slo-mo playback, clustering the results into four classes denoted *fail*, *low*, *medium*, and *high*. Furthermore, the authors in [12] proposed firstly new datasets for wheeled robots, including different locations, day-night shifts, moving objects, and poor illumination, and second a new metric for robustness evaluation based on a judgment of "correctness" through an empirically chosen threshold on the ATE. Like the previously cited SlamBench, the last works' main weakness is the limited flexibility, controllability, and scalability of the data without automated procedures, limiting the possible usage for statistically relevant large scale tests on robustness and consistency.

III. FRAMEWORK ARCHITECTURE

The core of the VINSEval framework architecture, shown in Fig. 3, is organized as a *Robot Operating System (ROS)* package and is composed by two fundamental software modules: the *Data Generation Module* (cyan block) and the *Estimators Evaluation Module* (red block). The former takes as an input a given generated trajectory file containing timestamped ground-truth poses, velocity, and acceleration measurements that could be either noisy (e.g., recorded from a real platform equipped with an IMU and a motion capture system) or noise-free (e.g., synthetically generated). The input data is then processed and used to produce ROS bagfiles of sensor data for the other module, whose primary

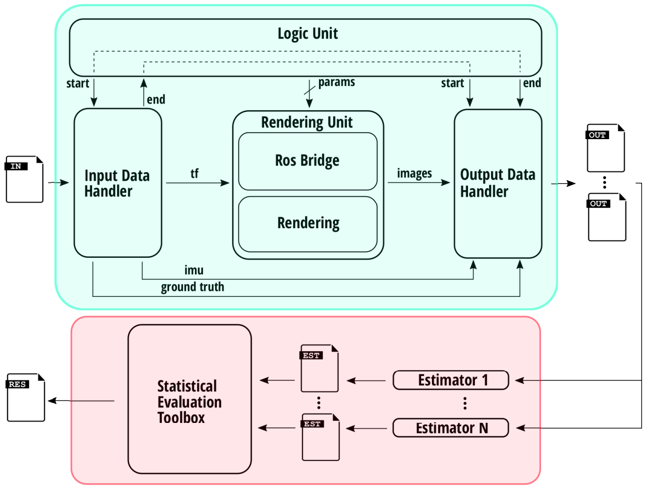

Fig. 3. Framework architecture overview: the full pipeline is composed by two main blocks, in cyan is the data generation block that allows to generate unlimited amount of data containing ground-truth, noisy IMU, and photo-realistic images given a set of trajectories while in red is the estimator evaluation block, that allows to perform consistency and robustness evaluation of estimators with the generated data.

objective is to run different VINS algorithms and provide a statistical evaluation of robustness and consistency.

A. Data Generation Module

The data generation module is divided into four submodules. The *logic unit* directs all the other submodules by providing them with control signals and parameters. Upon the logic unit's start signal, the *input data handler* takes the provided trajectory file as input and parses it. If the data contain noise-free IMU measurements, the input data handler adds noise and biases, for which statistics are provided as parameters following the inertial sensor model described in [13]. This data is then published as ROS messages to be used by subsequent submodules. The *rendering unit* is derived from the photo-realistic simulation framework FlightGoggles [5] with our extended capabilities such as the introduction of a fisheye distortion model for the camera lens, a variable time delay on the image header timestamp, and rendering parameters that allow changes of the visual attributes (e.g., illumination, objects distribution, etc…See Sec. V) in the scene. The rendering of photo-realistic camera images can be set to either asynchronous, such that the camera images are rendered in real-time at a frame rate dependent on the machine's performance, or synchronous, such that the camera images are synchronized with the given trajectory poses. Note that this modularization of the input data handler and the rendering unit allows for proprioceptive data from real systems (i.e., robot motion) to be used for a virtual rendering of precisely controlled exteroceptive measurements (i.e., camera image). The last submodule of the data generation module is the *output data handler*, which manages to start and stop recording the data into bagfiles when triggered by control signals from the logic unit. It automatically applies a realignment of the bagfiles substituting the wall time with the header time.

B. Estimators Evaluation Module

The statistical evaluation toolbox module of VINSEval consists of different submodules such as (i) the conversion of estimated trajectories to CSV files, (ii) finding associations between two trajectories based on their timestamps, (iii) spatial alignment tools to align the estimated trajectory with the true trajectory supporting different alignment types as in [14], (iv) absolute trajectory error evaluation based on associated and aligned trajectories, and finally (v) the computation of the Normalized Estimation Error Squared (NEES) and Average NEES (ANEES). On top of these, the estimator evaluation module supports an automated evaluation of different scenarios and multiple experiments and automated report generation.

IV. Error Metrics and Estimators Evaluation

Before we detail our approach on defining the different parameters to test for and their sweeping range, we first define what we understand under consistency, credibility, and robustness and explain the associated error metrics.

A. Consistency and Credibility

Estimators such as least-squares and Kalman filters provide assessments in terms of their error covariance matrix or Mean Squared Error Matrix (MSE) and the estimated state. The estimation error $\epsilon_j = \hat{\mathbf{x}}_j - \mathbf{x}_j \in \mathbb{R}^k$ is the difference between estimated and actual true value. The NEES is a commonly used metric that normalizes the scalar magnitudes of the estimation error ϵ_j based on the error covariance $\mathbf{P}_j$

$$\|\epsilon_j\|^2_{\mathbf{P}_j^{-1}} = \epsilon_j^T \mathbf{P}_j^{-1} \epsilon_j \in [0, \infty]. \tag{1}$$

The NEES is assumed to be χ^2 distributed with k degrees of freedom and a mean of k. Therefore, a chi-square significance test can be performed to judge if an estimator violates a certain credibility threshold [15]. A too low or too high NEES, depending on k, indicates under- and overconfidence, respectively. As the ground-truth is needed, the NEES is typically computed offline using M Monte Carlo simulations and then averaged over the M runs and normalized with respect to the state dimension k resulting in the ANEES:

$$\text{ANEES} = \frac{1}{kM} \sum_{m=1}^{M} \|\epsilon_{j_m}\|^2_{\mathbf{P}_{j_m}^{-1}}. \tag{2}$$

For our evaluation, we propose to compute the $\overline{\text{NEES}}$, the mean of the NEES, over the time span of each trajectory with D time steps, and then the $\overline{\text{ANEES}}$ as follows:

$$\overline{\text{ANEES}} = \frac{1}{kM} \sum_{m=1}^{M} \frac{1}{D} \sum_{j=0}^{D} \|\epsilon_{j_m}\|^2_{\mathbf{P}_{j_m}^{-1}}. \tag{3}$$

Computing the $\overline{\text{NEES}}$ reduces the significance of sporadic spikes that occur typically at the initialization phase until the filter starts to converge. Based on the $\overline{\text{ANEES}}$ and a credibility threshold, e.g., a probability interval of 99 %, we classify estimators to be credible or not. If the credibility threshold is reasonably high and violated, we assume the filter to be inconsistent.

B. Robustness

"*Robustness is the ability to withstand or overcome adverse conditions*." – [from online dictionary]. In the context of VI-SLAM and VIO, we can say that a robust estimator is resistant to deviations from the assumptions of optimal conditions. Hence if the assumptions are only approximately met, the estimator still has a reasonable performance. Contrary to estimator credibility, finding a metric to judge the robustness of a given estimator is particularly difficult. Here, we adopt a simple metric based on the Root Mean Square Error (RMSE) to define the so-called *Breaking Point (BP)*. Consider a given visual attribute (e.g., the illumination in the scene) that is changed L times from the optimal condition with an increasing amount of changes towards a bad condition. The BP is the point along the scale of change at which a given estimator breaks. Thus, for each attribute value change, the average RMSE of the ATE is computed. The RMSE is then compared to an empirical threshold to distinguish whether the estimator has broken or not.

V. Environment and parameter setup

As mentioned in Sec. III the rendering module inherits all the capabilities of FlightGoggles [5] and thus the various other types of exteroceptive sensors (other than camera and IMU) such as RGB-D cameras, IR beacon sensors and time-of-flight range sensors for which intrinsic, extrinsic parameters and noise specification can be easily adjusted. Moreover we added options for variable sensor time delays and online parameter adaptations in the scene and the system. We also extended the default pinhole camera model with a realistic fisheye lens based on the *atan model* [16]. To improve runtime efficiency the undistortion of each output pixel is calculated at startup and saved in a lookup table, given the diagonal distortion parameter s as described in [16]. However, the undistorted pixel values are most likely non integer values. Therefore the average color value is calculated at runtime with Eq. (4). $C\big(\mathbf{p}_d(j,i)\big)$ is the color value of the distorted integer pixel $\mathbf{p}_d(j,i)$, $\mathbf{p}_u(j,i)$ its corresponding undistorted non-integer pixel value. $C\big(\underline{\mathbf{p}}_u\big)$ is the lower-left, $C\big(\overline{\mathbf{p}}_u\big)$ the upper-left, $C\big(\overline{\mathbf{p}}_{u+1}\big)$ the upper-right, and $C\big(\underline{\mathbf{p}}_{u+1}\big)$ the lower-right surrounding undistorted pixel color values. δx and δy are the differences between the lower-left (integer) undistorted pixel $\underline{\mathbf{p}_u}$ and the calculated undistorted pixel $\mathbf{p}_u(j,i)$, in x- and y-axis respectively.

$$\begin{aligned} C(\mathbf{p}_d(j,i)) = \delta x \cdot \left(\delta y \cdot C\big(\underline{\mathbf{p}}_u\big) + \big(1-\delta y\big) \cdot C\big(\overline{\mathbf{p}}_u\big)\right) \\ + \big(1-\delta x\big) \cdot \left(\delta y \cdot C\big(\underline{\mathbf{p}}_{u+1}\big) + \big(1-\delta y\big) \cdot C\big(\overline{\mathbf{p}}_{u+1}\big)\right) \end{aligned} \tag{4}$$

Further, this framework provides an RGBA color to grayscale conversion based on the methods described in [17]. This work showed that the method used to convert colored images can greatly impact the result. Although all methods are implemented in VINSEval, we opted to use the *Luminance* method in the presented sample evaluation, as it maps the human eye brightness perception most closely [18].

A. Estimator Parameter Setup

Although highly customizable, we suggest here a specific set of parameters and environment settings to use in the proposed VINSEval framework to generate data and evaluate different open source state-of-the-art VINS algorithms. We generate UAVs feasible trajectories and noise-free IMU measurements at 200Hz. Trajectories are generated with a minimum snap trajectory generation approach, as described by [19]. The considered VINS algorithms are: **LARVIO** [20] and **OpenVins** [21] which are both filter-based VIO algorithms leveraging the Multi-State Constraint Kalman Filter (MSCKF) sliding window formulation. Both filters allow online camera-imu calibration, zero velocity update, different landmark parametrizations and first estimate jacobian formulation aiming to improve the filter consistency. **ROVIO** [22], [23], a fully robocentric and direct filter based VIO algorithm which makes use of the pixel intensity errors of image patches, aiming to achieve high level of robustness. **Vins-Mono** [24], an optimization-based sliding window formulation VIO algorithm aiming to provide high accuracy. All the algorithms used in our experiments have been tuned to get the best results in a randomly selected subset of the whole data used. The extrinsic and intrinsic parameters of the camera, as well as the distortion coefficient, have been set to the correct value provided by the rendering unit of VINSEval, and the online calibration of such parameters was turned off. When there is a time delay between the camera and the IMU we turn on the online estimation of such time delay providing the correct value as an initial guess, on the estimators that allow that. Moreover, we provide all estimators with the correct IMU noise statistics. Regarding the feature tracker, we similarly tuned every feature-based algorithm to achieve best results for all involved algorithms.

B. Experiments Setup

In our experiments we have considered mainly four attributes which are particularly relevant in real-world situations: (i) Changes in amount of informative visual features (ii) Changes in illumination (iii) Changes in time delay between the camera and the IMU (iv) Changes of the IMU noise and noise statistics. For each of the considered attributes a, we have defined $L = 10$ different "difficulty" levels for which increasing levels produce a more complicated scenario for an estimator. For every single level $l \in [1, L]$ we run $M = 20$ different simulations where we dynamically change all the other environmental conditions and sensor specifications, including the attributes that are not evaluated and other parameters such as object placement distribution or UAV trajectory while keeping all of them in the range of what we defined to be "optimal". These parameter swaps provide randomness to the evaluation and average over-polluting effects, leaving only the change effects in the single attribute under consideration. Thus, sweeping over one attribute generates 200 test runs per VINS algorithm.

For a given attribute a, the following subsections describe how the level l has been mapped to a change of the considered attribute, and how "optimal" values are defined.

C. Changes in amount of informative visual features

For this attribute, we evaluate the former cited algorithms' performance when the amount of informative features changes. We introduce an *information-density* parameter D, determining the overall amount and placement of recognizable features within the scene compared to either self-similar or featureless ground. A value of 1 corresponds to the approximation of the ideal, informative-rich scene, while 0 will not place any objects. Values in between will decrease the placement probability of objects linearly, with a multiplier based on position-dependent Perlin noise. A value of 0.5 would place half as many objects compared to the ideal scene, with higher object densities around Perlin-based clusters. The attribute level l influences the generated scene twofold: a linear multiplier of the object placement density between the maximum at the easiest and $5\,\%$ at the most challenging level as well as decreasing clustering with growing difficulty.

D. Changes in illumination

For this attribute, the illumination of the virtual scene changes over the different values of the level l, reducing the illumination intensity I for a fixed window of time during the UAV trajectory, from its optimal value to lower values as the level l increases. The effect of decreasing the illumination in a real-world scenario using a camera set with auto-exposure triggers a chain reaction, which increases the camera's exposure time with the consequence of increasing the amount of motion blur that the images will have. However, in these experiments, we are simulating a camera with fixed exposure time and without any simulated motion blur being applied. Thus we aim to evaluate the estimators' performance against abrupt changes of the illumination intensity on the scene. We consider the optimal value to be $I = 1$, corresponding to the attribute level $l = 1$, which emulates a sunlight condition on a clear day. The mapping between the attribute level l and the illumination intensity I has been defined through a second-order function, as follows:

$$I = \alpha\,(l-1)^2 + \beta\,(l-1) + 1 \tag{5}$$

With empirical values $\alpha = 0.0137$ and $\beta = -0.23$ to achieve a fairly dark environment at the most challenging level.

E. Changes in time delay between the camera and the IMU

Let us consider the scenario for which camera images are captured synchronously with the IMU measurements. For time delay between the camera and IMU, we consider the delayed image's timestamp when the image is available to the estimator (e.g., USB delay). Thus, in this scenario, we aim to evaluate how estimators manage such a delay. For a given attribute level l, the images header timestamp is defined:

$$t_{cam} = t_{imu} + k\big(l\,(l-1)\big) \tag{6}$$

With $k = \frac{5}{3000}$ heuristically chosen, leading to a maximum time delay of $150\,\mathrm{ms}$, for $l = 10$ and to no delay for $l = 1$.

F. Changes of the IMU noise and noise statistics

The last attributes we considered within this evaluation are the accelerometer and gyroscope noise densities and random walk. The value changes range from a simulated high grade IMU down to a very low-performing MEMS IMU. For a given attribute level l the IMU noise statistics are thus changed according to:

$$\sigma_* = \varphi\left(10^{\psi(l-1)}\right) l\sigma_{*_{opt}} \tag{7}$$

Where σ_* indicates the continuous-time IMU noise densities and random walks scaled concerning the optimum value $\sigma_{*_{opt}}$ by a scale factor. The value $\varphi = 2$ and $\psi = \frac{2}{9}$ has been chosen empirically leading to the min. and max. values reported in Tab. I. In this case, the optimal (or better, realistic) value has been chosen to correspond to an attribute level $l = 5$, in order to have IMU noise statistics in the same order of magnitude of the majority of the MEMS IMU used nowadays on UAVs, and to avoid cases of having an accurate estimation even if all tracked features are lost.

TABLE I

MINIMUM AND MAXIMUM VALUES OF THE ACCELEROMETER AND GYROSCOPE NOISE DENSITIES AND RANDOM WALKS CONSIDERED

	$\sigma_{*_{min}}$	$\sigma_{*_{max}}$	
σ_a	$1.0e^{-4}$	$2.0e^{-1}$	$m/s^2\sqrt{Hz}$
σ_g	$1.0e^{-5}$	$2.0e^{-2}$	$rad/s\sqrt{Hz}$
σ_{ba}	$1.0e^{-5}$	$2.0e^{-2}$	$m/s^3\sqrt{Hz}$
σ_{bg}	$1.0e^{-6}$	$2.0e^{-3}$	$rad/s^2\sqrt{Hz}$

VI. SAMPLE EVALUATION PROCESS

For each of the previously described attributes we have generated $L \times M = 200$ different bagfiles of raw data, each one containing $40\,\mathrm{Hz}$ rendered VGA camera images and $200\,\mathrm{Hz}$ IMU data for a UAV trajectory of $100\,\mathrm{s}$, resulting in a total of 800 different bagfiles and about $1\,\mathrm{TB}$ of data. With the synchronous rendering option, the data can be generated faster than real-time, meaning more than $22\,\mathrm{h}$ of data has been generated in $8\,\mathrm{h}$[1]. The generated raw data is then used in the estimator evaluation module (cf. Sec. III-B), which feeds the data to the mentioned VINS estimators and starts a, in our case 3-day long, batch run. Each estimated trajectory corresponding to a specific attribute a, level l, run m, and estimator e is first aligned with the corresponding ground-truth trajectory along the unobservable dimensions (i.e. position and yaw). Then the ARMSE and the ANEES over the time span of the trajectory are computed for position and orientation. This automatically generates a report at very high detail level. As last step of the estimator evaluation, the $10\,\%$ of the data, corresponding to the worst runs in terms of normalized sum of the single error metrics, accounting for processor-load hick-ups in the OS, have been removed. Then a final summary report, shown in Fig. 2, 4, 5, is produced by computing, for each attribute a and level l, the ARMSE and ANEES over the M runs for each estimator.

[1]We run VINSEval on a high-performance simulation PC, however, data generation in synchronous mode results faster than real-time even in a mid-performance laptop.

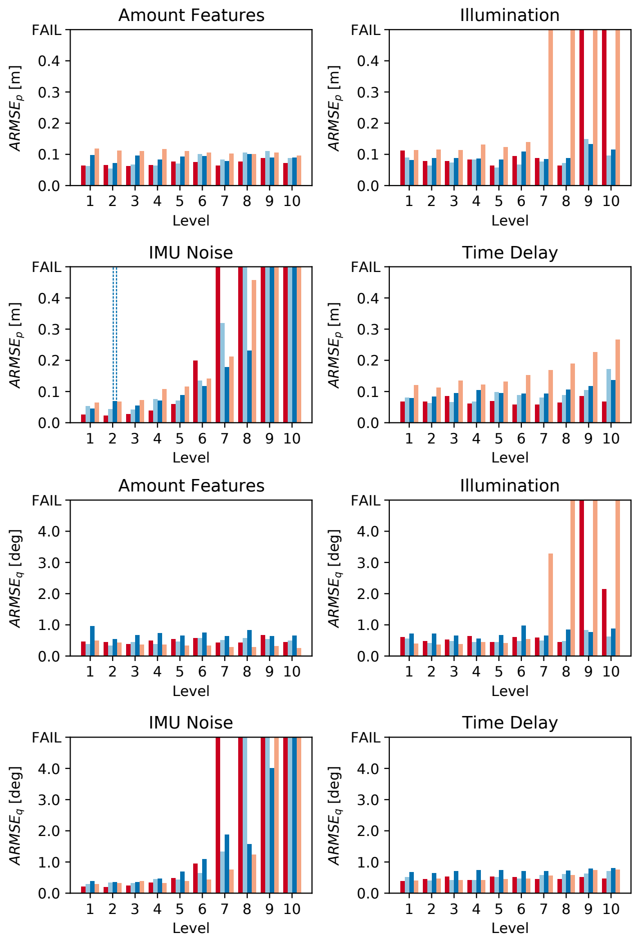

Fig. 4. Performance Comparison: Position and orientation ARMSE. Open-Vins [21] in red, LARVIO [20] in cyan, Vins-Mono [24] in blue and ROVIO [22], [23] in light red. We can notice in dashed blue that Vins-Mono, when tested with increasing IMU noise, is failing at attribute level $l = 1$ but not on further levels. Our investigation lead to the conclusion that very low IMU noise values cause numerical issue and then lead to a failure. We tackled the problem by tuning Vins-Mono with a falsely increased IMU noise. A strength of our framework is precisely to reveal such edge cases.

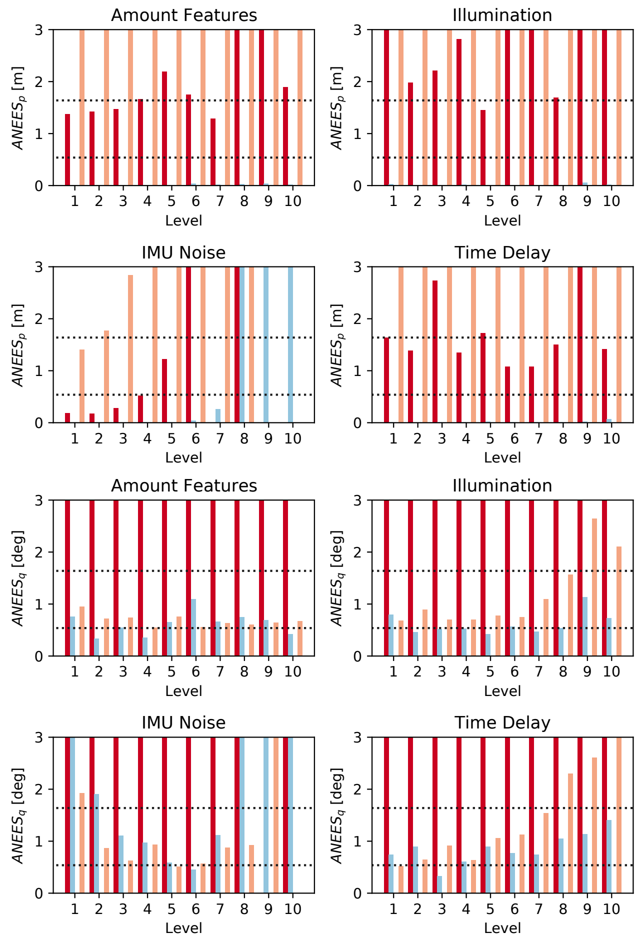

Fig. 5. Performance Comparison: Position and orientation $\overline{\text{ANEES}}$. Open-Vins [21] in red, LARVIO [20] in cyan and ROVIO [22], [23] in light red. Dashed are the confidence bounds. Despite our best tuning efforts, we were not able to reproduce the $\overline{\text{ANEES}}$ for Open-Vins reported by the authors; too little details on their method is given in [21]

As described in Sec. IV-B, we made use of the position and orientation ARMSE for which we defined a threshold to judge the failure of a an estimator. In particular, for a given attribute a, a level l and an estimator e we define the following binary score: ${}^{a}_{l}\mathcal{F}_e = True\ if\ ARMSE_* > T_{H_*};\ \ False\ if\ ARMSE_* < T_{H_*}$. Where the symbols $*$ stand either for position or orientation and T_{H_*} is the chosen threshold. With about $70\,\mathrm{m}$ trajectories, T_{H_*} are heuristically set to be $0.5\,\mathrm{m}$ for position and 5° for orientation. The first occurrence of ${}^{a}_{l}\mathcal{F}_e = True$ for increasing values of l, will define the BP per attribute a, per estimator e.

VII. Conclusion

In this paper we presented VINSEval, a unified framework for statistical relevant evaluation of consistency and robustness of VINS algorithms with fully automated scoreboard generation over a set of selectable attributes. We showed the ability of effective evaluation given by the flexibility on parameter selection, the mitigation of polluting effects through multiple runs with randomization in dimensions not under test, and the inherent detection of edge-cases through the wide test span in an automated fashion. We will open-source VINSEval making it a usable and extendable tool for the community towards unified estimator evaluation.

As a sample VINSEval demonstration, we let a Breaking Point score, in Fig. 2, to be generated to show how robust and consistent current state-of-the-art algortihms are. All tested algorithms generally exhibit low ARMSE when challenged with increased imu-camera time delay, decreasing illumination and amount of informative features showing the ability to compensate for dark scenes and correctly detect and track self-similar features on the background. However, all the algorithms show high sensitivity to IMU noise statistics, with a tendency to fail with a low-preforming MEMS IMU. Particularly interesting, Fig. 4 shows the edge-case of numerical errors encountered in Vins-Mono [24] when having very low IMU noise values. Regarding credibility/consistency results, Fig. 5, show that none of the considered algorithms can be labeled as credible due to its under- or overconfidence and that still much research is required towards this direction.

Acknowledgement

This work is supported by the EU-H2020 project BUGWRIGHT2 (GA 871260); ARL under grant agreement W911NF-16-2-0112; University of Klagenfurt (KPK-NAV).

REFERENCES

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 12 2016.

[2] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor UAVs using ROS and Gazebo," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7628 LNAI, 2012, pp. 400–411.

[3] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "RotorS—A modular gazebo MAV simulator framework," in *Studies in Computational Intelligence*, A. Koubaa, Ed. Cham: Springer International Publishing, 2016, vol. 625, ch. RotorS—A, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23

[4] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," in *Field and Service Robotics*, 2018, pp. 621–635. [Online]. Available: https://arxiv.org/abs/1705.05065

[5] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "FlightGoggles: Photorealistic Sensor Simulation for Perception-driven Robotics using Photogrammetry and Virtual Reality," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, 11 2019, pp. 6941–6948. [Online]. Available: https://doi.org/10.1109/iros40897.2019.8968116

[6] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A Flexible Quadrotor Simulator," *arXiv preprint arXiv:2009.00563*, 2020. [Online]. Available: http://arxiv.org/abs/2009.00563

[7] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger, "Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset," in *British Machine Vision Conference 2018, BMVC 2018*, 2019.

[8] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. Kelly, A. J. Davison, M. Luján, M. F. O'Boyle, G. Riley, N. Topham, and S. Furber, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June, 5 2015, pp. 5783–5790.

[9] B. Bodin, H. Wagstaff, S. Saecdi, L. Nardi, E. Vespa, J. Mawer, A. Nisbet, M. Lujan, S. Furber, A. J. Davison, P. H. Kelly, and M. F. O'Boyle, "SLAMBench2: Multi-Objective Head-to-Head Benchmarking for Visual SLAM," in *Proceedings - IEEE International Conference on Robotics and Automation*, 5 2018, pp. 3637–3644.

[10] M. Bujanca, P. Gafton, S. Saeedi, A. Nisbet, B. Bodin, M. F. Oaboyle, A. J. Davison, P. H. Kelly, G. Riley, B. Lennox, M. Lujan, and S. Furber, "SLAMBench 3.0: Systematic automated reproducible evaluation of slam systems for robot vision challenges and scene understanding," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May. IEEE, 2019, pp. 6351–6358.

[11] W. Ye, Y. Zhao, and P. A. Vela, "Characterizing SLAM Benchmarks and Methods for the Robust Perception Age," 5 2019. [Online]. Available: http://arxiv.org/abs/1905.07808

[12] X. Shi, D. Li, P. Zhao, Q. Tian, Y. Tian, Q. Long, C. Zhu, J. Song, F. Qiao, L. Song, Y. Guo, Z. Wang, Y. Zhang, B. Qin, W. Yang, F. Wang, R. H. M. Chan, and Q. She, "Are We Ready for Service Robots? The OpenLORIS-Scene Datasets for Lifelong SLAM," pp. 3139–3145, 11 2019. [Online]. Available: http://arxiv.org/abs/1911.05603

[13] S. Weiss and R. Siegwart, "Real-time metric state estimation for modular vision-inertial systems," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 231855, pp. 4531–4537, 2011.

[14] Z. Zhang and D. Scaramuzza, "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry," in *IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2018, pp. 7244–7251.

[15] X. R. Li, Z. Zhao, and X. B. Li, "Evaluation of Estimation Algorithms: Credibility Tests," *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 42, no. 1, pp. 147–163, 2012.

[16] F. Devernay and O. Faugeras, "Straight lines have to be straight," *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, 2001. [Online]. Available: http://link.springer.com/10.1007/PL00013269

[17] C. Kanan and G. W. Cottrell, "Color-to-Grayscale : Does the Method Matter in Image Recognition ?" *PLoS ONE*, vol. 7, no. 1, 2012.

[18] W. K. Pratt, "Digital Image Processing, 4th Edition," *Journal of Electronic Imaging*, 2007.

[19] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.

[20] X. QIU, H. ZHANG, and W. FU, "Lightweight hybrid visual-inertial odometry with closed-form zero velocity update," *Chinese Journal of Aeronautics*, 2020.

[21] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A Research Platform for Visual-Inertial Estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4666–4672.

[22] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem. IEEE, 2015, pp. 298–304.

[23] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[24] 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.