Benchmarking Cameras for OpenVSLAM Indoors

Kevin Chappellet, Guillaume Caron, Fumio Kanehiro, Ken Sakurada, Abderrahmane Kheddar

Benchmarking Cameras for OpenVSLAM Indoors

Kevin Chappellet*†§, Guillaume Caron*‡, Fumio Kanehiro*†, Ken Sakurada†
Abderrahmane Kheddar*§,
*CNRS-AIST JRL (Joint Robotics Laboratory), UMI3218/IRL, Japan
†National Institute of Advanced Industrial Science and Technology (AIST), Japan
‡UPJV, MIS Laboratory, Amiens, France
§University of Montpellier, France

***Abstract*—In this paper we benchmark different types of cameras and evaluate their performance in terms of reliable localization reliability and precision in Visual Simultaneous Localization and Mapping (vSLAM). Such benchmarking is merely found for visual odometry, but never for vSLAM. Existing studies usually compare several algorithms for a given camera.**

The evaluation methodology we propose is applied to the recent `OpenVSLAM` framework. The latter is versatile enough to natively deal with perspective, fisheye, 360 cameras in a monocular or stereoscopic setup, an in RGB or RGB-D modalities. Results in various sequences containing light variation and scenery modifications in the scene assess quantitatively the maximum localization rate for 360 vision. In the contrary, RGB-D vision shows the lowest localization rate, but highest precision when localization is possible. Stereo-fisheye trades-off with localization rates and precision between 360 vision and RGB-D vision.

The dataset with ground truth will be made available in open access to allow evaluating other/future vSLAM algorithms with respect to these camera types.

I. Introduction

In the context of a closed-known-environment (i.e. indoor), where an exploration phase for a robot is not necessary, visual Simultaneous Localization and Mapping (vSLAM) algorithms can be used in two phases. The first one is to create a map with a sensor commonly used by the robot. This pre-built map will be used for further localization only. This recently proved to be very efficient in a robotic industrial context, e.g. [1]. Some works show that the field of view, a multiple view points, and the depth have an impact on the localization. In this work, we rather investigate for the first time, how the localization is impacted by the choice of various cameras. We use the unified feature-based vSLAM framework `OpenVSLAM` [2] which allows using various cameras as input. To our best knowledge, the only work evaluating practically the impact of the camera type on vision-based 3D motion estimation concerns visual odometry (hence no relocalization) [3]. The latter focuses on single cameras of conventional and panoramic field-of-views, both indoors and outdoors, and in simulated and actual environments. It is found that a small FoV is preferred for large scale scenario while a large FoV is more suited for small confined environments. Our paper goes beyond the previous study by (i) considering the relocalization problem under varying lighting conditions and scene content, after a first mapping of the scene, and (ii) considering more camera types, including RGB-D, full spherical and stereoscopic types.

Our paper is organized as follows: Section II reviews various vSLAM with their specifications. Section III recalls impacts of camera types on vSLAM. Section IV introduces the evaluation metrics use in this study. Section VI highlights the methodology concerning the data acquisitions, followed by Section VII that analyzes the obtained results. Finally, Sections VIII and IX conclude our study.

II. Related works

Simultaneous Localization And Mapping is a rich research area ranging from filtering algorithms to loop-closure detection, exploiting various sensors as lidars, sonars, inertial measurement units (IMU) and cameras, to cite a few [27]. In this paper, we deal with visual SLAM [28] and vision-based *only* SLAM, i.e. without considering other types of sensors (e.g. IMU). Furthermore, we review the state-of-the-art within the spectrum of camera types including monocular, stereoscopic, multi-camera, either passive RGB (Red Green Blue) or active RGB-D (RGB plus Depth), of conventional, panoramic and full spherical field-of-views. Such line is complementary to existing vSLAM surveys and benchmarks [29]–[31], mostly focusing on visual feature types, filtering or optimization methods and map structures. Table I gathers and classifies a restricted number of key existing vSLAM, highlighting the most versatile ones in terms; i.e. those dealing with various field-of-views and combinations of cameras.

To the best of our knowledge, vSLAM technology started with the first visual SLAM running in real-time based on keypoints detected in images of a conventional camera [4]. Since then, one of the key contributions have been architectural, by running in parallel the features tracking and the mapping stages [5]. They were later renamed as front-end and back-end, respectively. The other type of key contribution is related to the visual feature considered such as very close to the camera measurement, i.e. pixel intensities [7], or much more abstract as Oriented fast and Rotated Brief (ORB) features [6]. All these works are done with the target of robustness and precision while keeping real-time localization and mapping, i.e. at the conventional RGB camera frame rate.

Redundancy brought by stereovision [8], [9] as well as considering the additional Depth modality (RGB-D), either in physics-based [10]–[12] or learning-based [13]–[17] features modeling, also improve precision and robustness of localization and mapping thanks to a better handling of partial

TABLE I: vSLAM related works with camera-type versatility.

Viewpoints			Modality		Field-of-view			References
Single Monocular	Multiple Stereo	Multiple Dual	RGB	Depth	Conventional	Hemispherical	Spherical	
✓			✓		✓			[4], [5], [6], [7]
	✓		✓		✓			[8]
✓	✓		✓		✓	✓		[9], [3]
✓			✓	✓	✓			[10], [11], [12]
✓			✓	✓$^{(a)}$	✓			[13], [14], [15], [16], [17]
✓			✓			✓		[18], [19], [20], [21]
✓	✓		✓	✓	✓			[22], [23], [24]
✓	✓	✓$^{(b)}$	✓		✓	✓		[25]
		✓$^{(c)}$	✓				✓	[26]
✓	✓	✓$^{(c)}$	✓	✓	✓	✓	✓	[2]

$^{(a)}$at least for learning, $^{(b)}$as diverging optical axes, $^{(c)}$as two opposite optical axes.

occlusions, illumination changes and low textured (mainly indoors) environments.

Monocular cameras of very wide field-of-view, i.e. 180 deg panoramic or more with a fisheye or catadioptric optics, have also been considered for vSLAM [18]–[20]. These methods mainly benefit from higher opportunities in catching strong image features than with a conventional camera. And also, from the better conditioning of camera pose estimation that such optics bring. Both characteristics being known to be capable of estimating more reliably trajectories with visual odometry than with a conventional camera [3]. Following a similar way, recently, vSLAM has been adapted to the combination of conventional cameras [25] as well as the combination of fisheye cameras [2], [26], to reach a field of view of up to 360 deg, i.e. a full spherical field of view.

Among the above related works, some are generic enough in terms of visual feature representation, or in terms of camera projection model to handle natively several types of camera [2], [9], [22]–[25]. However, to our best knowledge, `OpenVSLAM` [2] is the only framework that can natively handle RGB and RGB-D modalities, monocular or multi-camera setups of conventional or up to 360 deg field-of-view stereoscopic. That is why `OpenVSLAM` is considered in this paper to evaluate the impact of the camera type on the vSLAM results in term of localization, with the same visual feature type (ORB), estimation method (pose-graph optimization) or implementation.

III. Properties of Camera types and Algorithms

As `OpenVSLAM` is an indirect vSLAM, input frame(s) are pre-processed to extract features. These features are then processed to get 3D points used in the translation and rotation estimation of the current camera pose with respect to the map made of keypoints, whether during the mapping or the localization-only process (when the map is already available). However, depending on the camera type, frames are of different nature and the process to obtain keypoints is different as well. Thus, we briefly recall the different properties and algorithms by camera type as well as their consequences on estimations of both maps and camera poses.

A. Stereo

A stereo configuration links rigidly two monocular cameras. So, features are extracted from the stereo-frame, *ie.* two input frames, left and right. For each feature of the left frame, its corresponding feature is matched on the right frame assuming stereo rectified frames and using epipolar lines. A keypoint is composed of the left feature coordinate and the horizontal right match [22]. Knowing the baseline and the focal length, these keypoints are triangulated to estimate scale. Thus, maps and camera poses are estimated at scale from one stereo-frame.

B. RGB-D

Features are extracted from the RGB frame. For each of these features, the equivalent horizontal right match is obtained by using the corresponding measured depth value [22]. Then, as in the Stereo case, maps and camera poses are estimated at scale, but from measured depths, not estimated ones.

C. Monocular

Features are extracted from the RGB frame. Depth information is not observable from a single frame in case of a monocular configuration. It requires an initialization using structure from motion methods [22], *ie.* the triangulation of keypoints is done from several views of which poses are estimated up to scale. As scale is not constrainted by the camera itself, contrary to Stereo or RGB-D cases, the scale can drift over time, only corrected in case of loop-closure [6]. Hence, maps and camera poses are estimated up to scale.

IV. Evaluation metrics

The two main outputs of a vSLAM system are the estimated camera trajectory along with a map building of the environment. Evaluating the quality of the outcome map is a challenging work that involves 3D model of the existing environment, blueprints [32], or topological data in order to build an accurate groundtruth. Therefore, we will use the estimated camera trajectory that the vSLAM system outputs to evaluate its accuracy during both mapping and localization processes.

For the evaluation, we consider two sequences of poses: estimated trajectory $\mathbf{P}_0, \ldots, \mathbf{P}_n \in SE(3)$ and groundtruth trajectory $\mathbf{Q}_0, \ldots, \mathbf{Q}_n \in SE(3)$. As they may come from different sources such as camera or motion capture system, these sequences may have different length, sampling rates and possibly missing data that require to perform a data association as additional step. This is taken care automatically by `evo` framework [33] based on the common timestamps between the different sources. To simplify the following notations, we assume that sequences are time-synchronized, equally sampled and have the same length. A sequence is a succession of homogeneous transformation matrices of a frame from a reference frame. The considered frame of a monocular camera is often the optical frame. For a stereo rig, the center between optical frames is the first frame of the map used during the acquisition. In case of the motion capture system, the considered frame is arbitrarily deduced from the tracked markers and the reference frame that we chose to be associated to the ground during the calibration process.

In the following part we recall two common evaluation metrics for vSLAM [34], [35]. These metrics must be as low as possible to assess a good result.

A. Relative Pose Error (RPE)

Given a fixed increment $\Delta \in \mathbb{N}^*$ of frames, the relative pose error evaluates the local accuracy of the trajectory over Δ frames. A common and valid choice is to use $\Delta = 1$; which means it evaluates the drift per frame. Another choice is to use $\Delta =$ Hz where Hz is the acquisition rate of the camera, therefore it is estimating the drift over one second. To be fair, we use $\Delta = 1$ for Theta S and $\Delta = 2$ for Azure and T265 cameras thus considering them at 15 Hz to compare them on the same traveled distance.

The relative pose error for a given time step *i* is defined as:

$$\mathbf{E}_i = ({\mathbf{Q}_i}^{-1}\mathbf{Q}_{i+\Delta})^{-1}({\mathbf{P}_i}^{-1}\mathbf{P}_{i+\Delta}). \quad (1)$$

In order to have a single value representing the local accuracy of the trajectory, we compute the root mean squared error (RMSE) of the chosen operator for a sequence of length n (where $m = n - \Delta$) as:

$$\text{RMSE}_{\text{operator}}(\mathbf{E}_{0:n}, \Delta) = \left(\frac{1}{m} \sum_{i=0}^{m} \text{operator}(\mathbf{E}_i)^2 \right)^{1/2}, \quad (2)$$

with operator($\mathbf{E}_i$) defined as follow: 'trans' is the translation part such as operator($\mathbf{E}_i$) = ||trans($\mathbf{E}_i$)||. operator($\mathbf{E}_i$) could also be 'rot', 'full' or 'angle'[cite here]. We choose 'trans' as single operator as it is already impacted by the rotation error.

B. Absolute Pose Error

The absolute pose error (APE) evaluates the global consistency of the estimated trajectory by comparing the absolute error between the estimated and groundtruth poses over a trajectory. As previously mentioned, the reference frame for different trajectories may be different. We must align them before comparison. This alignment is done by least-squares estimation of the rigid-body transformation T between $\mathbf{P_{0:n}}$ and $\mathbf{Q_{0:n}}$ [36]. For a monocular setup, the scale cannot be recovered with vSLAM. An additional step of rescaling the estimated trajectory is necessary. This method requires as input both trajectories and n the number of frames to use in order to compute T. The estimated T slightly changes depending of n. For this study, after proceeding to the data association step, we choose to use the length of the trajectory for n. Once this transformation matrix is estimated, the absolute pose error is defined as:

$$\mathbf{E}'_i = {\mathbf{Q}_i}^{-1} T \mathbf{P}_i. \quad (3)$$

Similarly to the Relative Pose Error (RPE, Sec. IV-A) we compute the root mean squared error of translation, rotation, full and angle components of $\mathbf{E}'_i$ with (2), substituting $\mathbf{E}'_i$ (3) to $\mathbf{E}_i$ in (2).

Fig. 1: 3D printed support holding cameras and markers used by the motion capture system

V. Experimental Setup

A. Data acquisition

All the cameras' data are simultaneously recorded in one-go for each acquisition with a laptop on Ubuntu 18.04 and ROS melodic. The cameras were connected to the laptop while the motion capture system, from MotionAnalysis, was running on another Windows 7 desktop computer. The Cortex software provided by MotionAnalysis allows to publish the markers data on a socket. Both computers were on the same local network linked by an Ethernet cable. It allows us to record all the data using ROS capabilities and to have a synchronized dataset between groundtruth and the various cameras we consider in the benchmark. It simplified the later association process to compare the different data within `evo` framework.

B. Cameras

Table II gathers the specifications of the cameras we used to create this dataset. Figure 1 shows the support made in order to attach rigidly together every camera and markers. Then, we could record every sequence in one-go, bounding experimental biases: every camera follow the same trajectory, up to a constant rigid transformation, at the same pace and sees the same environment with the same light variations. Considered cameras with properties shown in Section III are:

TABLE II: Camera devices

Camera	T265	D435i	Theta S	Azure
Model	Fisheye	Perspective	Equirectangular	Perspective/Stereo
Setup	Stereo	RGB-D	Monocular	RGB-D
Resolution	848×800	640×480	1920×1080	1280×720
Frame Rate	30	30	15	30
FOV	163±5°	69.4°× 42.5°× 77°(± 3°)	360°*	90°× 59°

*Theoretically the FOV is 360°×360° but due to the mask, shown in Fig 2, the FOV is estimated to be 360°×[165;315]°.

1) Intel RealSense T265 stereo fisheye camera. It has a very wide field-of-view. It can be used as a fisheye only or as a stereo-fisheye.
2) Microsoft Azure RGB-D camera. It has the particularity to have a fisheye-depth sensor. The output RGB and Depth images are configurable. We chose to record the wide field-of-view (WFOV) Depth image, 2×2 binned aligned on RGB image.
3) Ricoh Theta S, as an equirectangular camera. It has the widest field-of-view. It is considered as a monocular camera.
4) Intel RealSense D435i RGB-D camera. The depth is computed from stereovision.

Despite the rules described later in Section VI-A about the acquisition methodology, D435i RGB-D camera did not allow having maps for any of the runs. Its field-of-view is the narrowest among the ones considered in this benchmark (Fig. 6). Such characteristics lowers the opportunities to sense features in low textured areas of any environment, particularly when the environment is narrow like ours. This is the reason why `OpenVSLAM` did not produce any result with the D435i. Thus, no mapping nor localization results with it could be shown and exploited to conduct this study.

C. Motion Capture System

For each trajectory we use the MotionAnalysis motion capture system for groundtruth recording. The setup is composed of two different types of Infra-Red (IR) cameras for a total of thirteen IR cameras. There are two Kestrel 2200 and eleven Kestrel Digital IR cameras sharing the same specifications such as a 2048×1088 resolution and 332 FPS. The limited number of available cameras and their coverage volume constrain the size of the described environment in V-E. The motion capture system tracks the 3D position of each marker. To meet tracking reliability, a marker is considered detected only if it is seen by at least three cameras. To get the orientation, we placed several markers in an L-shape configuration at the top of the cameras-support. The rotation is computed with respect to the position of left-marker (green marker in Fig 1). A black tape was put on different reflective parts of cameras and support to avoid possible outliers with the motion capture system markers during acquisitions.

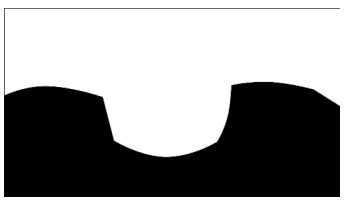


Fig. 2: Required mask for Ricoh Theta S camera to remove undesired part for mapping and localization. It occludes 46.2% of the image.

D. OpenVSLAM

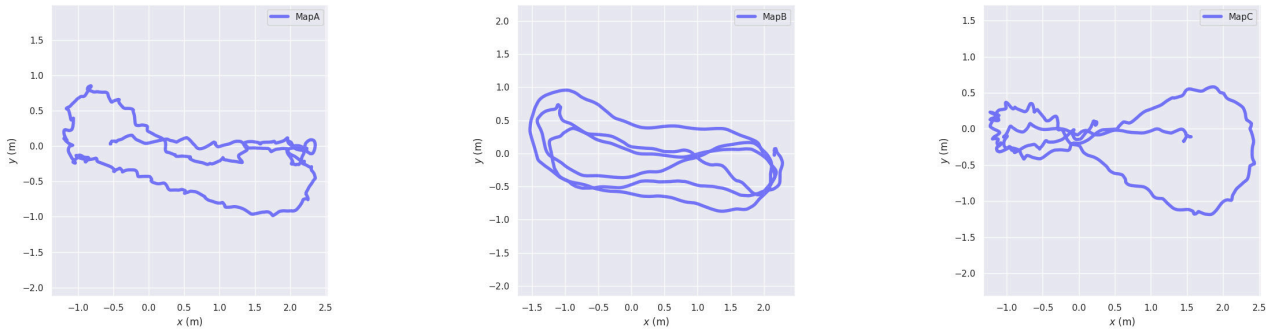
The vSLAM framework `OpenVSLAM` is publicly available under 2-clause BSD license. It is implemented in `C++` and based on well-known libraries such as `Eigen` for matrix computation, `OpenCV` for images manipulation and `g2o` for map optimization. As previously mentioned (Sec. II), `OpenVSLAM` allows using various camera models such as perspective, fisheye and equirectangular. RGB-D, perspective and fisheye camera models can be used in a monocular and stereo setups. However stereo-fisheye is still officially experimental. In addition, we defined a mask to remove undesired part of an image before features detection. The reason is that such a mask, see Fig. 2, prevents features detection on cameras support for Theta S camera.

E. Environment

Figure 4 shows the environment created in a specially equipped room of 8 meters by 5 meters with the motion capture system. It emulates a simple corridor along A-B path of approximately two meters of width and five meters of length with two separated areas. Both of them include two separate parts with different feature densities. For example, behind pose C in Fig 4 there is mainly windows and gray planes which may produce few features compared to the zone in front of pose D with several objects. While moving from A to B, the camera motions are almost limited to a straight line in order to not get too close to a wall on each side. This constrained environment implies to have rotation near pose C and B or near pose D and A.

VI. Data acquisition methodology

In this section we introduce the data recorded during various acquisitions used to produce an estimated trajectory with vSLAM. Records are clustered in two groups, one per purpose: mapping or localization. Indeed, we emphasize that some data are dedicated to obtain a map of the environment prior to perform a localization process with the remaining others. Recall that the main idea behind this decoupled process is to



(a) MapA (b) MapB (c) MapC

Fig. 3: Top-down views of different done trajectories during acquisitions used to build a map.

be able to navigate later on in the entire known environment to perform tasks (see a practical example in [1]).

Whereas dense vSLAM algorithms produce maps which are often visually checked by looking at ghost walls or duplicates of the same actual environment, sparse maps of featured-based vSLAM, as `OpenVSLAM`, makes difficult a similar analysis. Hence, to quantify its quality we use the $\text{RMSE}_{\text{trans}}$ of APE between the groundtruth and the estimated trajectory at the localization only stage.

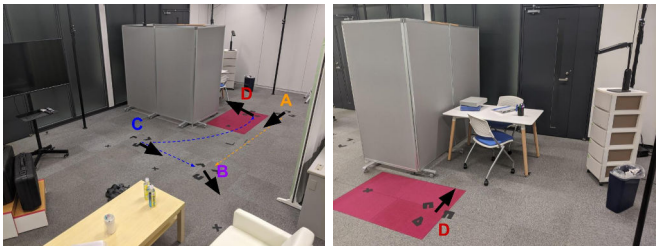


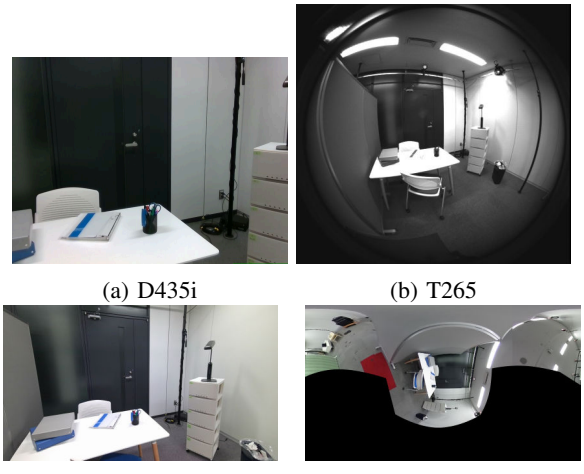
Fig. 4: Environment within the motion capture system with poses A, B, C, D and paths illustrated. Each black arrow is the looking direction of the camera for each pose.

A. *Mapping*

For a given environment, there is an infinite number of possible paths along which one can acquire images to create a map with vSLAM. The path has usually a significant impact on estimations. For example, relatively light variations in altitude along a given path would impact the map [3].

In this study, the cameras are always about 1.2 meters height. The almost constant height is a reasonable experimental characteristics when targeting mobile robots applications. However, to limit the experimental biases, we imposed ourselves few rules during acquisitions: slow motion, avoid rotation without translation and provoke loop closure by looking at previous recorded places. With this set of constraints in mind we recorded several maps in the environment of Fig. 4, a subset of which is illustrated in Figure 3 as MapA, MapB and MapC. A map is described by a couple $(length(m), duration(s))$ where $length$ corresponds to the length of the trajectory that the camera did to record the sequences; we have MapA (18.59, 162), MapB (26.94, 93) and MapC (17.52, 120).

Fig. 5: Two areas prior and after scenery modifications (chair, bucket, tables, tables objects, slippers, sofa).



(a) D435i (b) T265

(c) Azure (d) Theta S with mask

Fig. 6: Cameras field of view at pose D

B. *Localization*

For localization, we considered two cases shown in Fig. 4:

1) *Stationary pose*: four poses named A, B, C and D, are described by a triplet $(x(m), y(m), \theta(deg))$. From A to D, we have (1.96, -0.11, 168), (-1.11, -0.11, 158), (-0.99, 1.24, 121) and (1.68, 0.37, -28) respectively.
2) *Path*: a motion between two stationary poses described by a pair $(length(m), \Delta\theta(deg))$. We consider three different paths: A to B (3.48, -7), C to B (1.66, 39) and C to D (4.58, -151).

Poses A, B, C and D were defined within the environment in front of several areas of various content. For each case, we record three different sequences in order to evaluate statistically the estimated localization within a pre-built map. We repeat this procedure under three different conditions:

- *Nominal*: default environment in which there is no variation between mapping sequences and localization sequences;
- *Lighting*: prior to record the images sequences we modify the lighting conditions of the environment by switching off a subset of the available ceiling lights.
- *Scenery*: prior to record image sequences, we moved several objects of various size (Fig. 5).

Note that `OpenVSLAM` is not designed to support these perturbations. However these conditions reflects the possible changes that may happen in any environment. Therefore, they are used in this study to quantify how considered cameras are sensitive to these perturbations.

VII. Results

First, we compare three mapping runs to select the best reference for the purpose of a fair comparison for further localization-only runs.

A. Mapping

Table III shows indirectly the mapping error thanks to the APE ($RMSE_{trans}$) computed for the three MapA, MapB, MapC runs (Fig. 3). Before computing errors, estimated and measured camera trajectories are aligned to transform the former in the groundtruth reference frame. Only Theta S estimations are scaled. Maps are not modified to be used as they were estimated during the mapping phase for later localization only. To sum up, the Azure camera has the lowest $RMSE_{trans}$ of APE, followed by the T265 and lastly Theta S among the three runs, each having a path turning at both extremities around A and B. Among the three runs, T265 camera has the smallest deviation whereas both Azure and Thetas S cameras share their lowest error within MapA. Then, MapA is selected to serve as reference for a fair comparison between the various cameras.

B. Localization

Using MapA as input for localization, we obtain estimated poses and trajectories for the localization of each stationary and moving cases described in Section VI-B. Estimations are transformed in the groundtruth frame, for further evaluation, using transformations kept from the map alignments (Sec. VII-A). Table IV and table V gathers estimation errors shown as the mean of $RMSE_{trans}$ of APE and RPE, respectively written APE_{trans} and RPE_{trans} hereafter, for shortness. APE_{trans} column evaluates the absolute position within the environment. RPE_{trans} column evaluates the position drift over the trajectory. The percentage column beside APE and RPE columns quantifies the percentage of the trajectory estimated by `OpenVSLAM` with respect to the trajectory recorded in the groundtruth. Table VI gathers percentage changes with respect to Nominal case computed as: $100 * (v_{condition} - v_{Nominal})/v_{condition}$ where $condition \in \{Lighting, Scenery\}$ and $v \in \{APE_{trans}, RPE_{trans}, \%\}$, the closer to 0 it is the better it is.

TABLE III: $RMSE_{trans}$ of APE in meter for MapA, MapB and MapC

	MapA	MapB	MapC
Azure	**0.0977**	0.4710	0.1209
T265	0.1640	0.1694	**0.1634**
Theta S	**0.1907**	0.2401	0.2162

bold highlights the best result for each camera.

VIII. Discussion

Focusing, first, on vSLAM localization rates in narrow environment under nominal conditions, our study confirms the recommendation of using a wide FoV camera for visual odometry in confined environment [3]. In our study, Theta S equirectangular camera is the only monocular one but it leads to better localization rates under nominal conditions than both Azure RGB-D and T265 stereo-fisheye cameras (Tab. V). Clearly, the widest FoV allows matching numerous keypoints in the pre-built map, in accordance with [3]. Furthermore, only Theta S could localize itself at every stationary pose in nominal conditions (Tab. IV).

However, still in the nominal case, Theta S is the best, only regarding localization rates. Indeed, T265 has the smallest RPE_{trans} (Tab. V), showing that a wide FoV alone, even the widest, is not sufficient to get the best local accuracy. The combination of depth estimation (Sec. III) with the wide FoV permits T265 to reach this result. Azure has a depth sensor, but it does not compensate its narrower FoV.

Despite the FoV importance for local accuracy, Azure has the best global accuracy, *ie.* the smallest APE_{trans} (Tab. IV and V), thanks to its depth sensor. Indeed, T265's depth is estimated (Sec. III), thus more sensitive to calibration and matching errors than Azure. Theta S, after scaling its estimations, has 57.4% worse global accuracy than Azure and 19.6% worse than T265, due to scale drift.

These results and analyses must be put into perspective of the map quality. The APE_{trans} from MapA (Tab. III) is a quantitative indicator of the "quality" of the pre-built map. It is similar to APE_{trans} values in Tab. V thus assessing the direct impact of the mapping on the global accuracy of later localization processes.

Beyond conclusions of [3], as in nominal case, lighting and scenery conditions (Tab. V) lead Azure to the lowest

TABLE IV: Mean of $RMSE_{trans}$ of APE and RPE in meter and localization rate w.r.t. ground truth within MapA for stationary poses

Element	Camera	Nominal			Lighting			Scenery		
		APE_{trans}	RPE_{trans}	%	APE_{trans}	RPE_{trans}	%	APE_{trans}	RPE_{trans}	
A	Azure	–	–	–	–	–	–	–	–	–
	T265	**0.1961**	0.0097*	88.1868	–	–	–	0.1971	0.0117*	43.9128
	Theta S	0.3314	**0.0073**	**98.1345**	–	–	–	0.3278	**0.0088**	**95.4245**
B	Azure	**0.0651**	**0.0042**	93.8273*	**0.0590**	0.0049	93.9026	0.1198	0.0125*	92.2759*
	T265	0.1676	0.0045*	**97.8788**	0.1585	**0.0044**	**97.8540**	0.1621*	**0.0049**	**98.7390**
	Theta S	0.1083	0.0045*	96.5242	0.1020	0.0054*	91.9277*	**0.1025**	0.0053	94.5744
C	Azure	–	–	–	–	–	–	–	–	–
	T265	–	–	–	–	–	–	–	–	–
	Theta S	**0.2978**	**0.0158**	**95.0270**	–	–	–	**0.2700**	**0.0229**	**67.6553**
D	Azure	–	–	–	–	–	–	–	–	–
	T265	–	–	–	–	–	–	–	–	–
	Theta S	**0.1601**	**0.0086**	**87.1713**	–	–	–	–	–	–

bold highlights the best result, * marks the worst result.

TABLE V: Mean of $RMSE_{trans}$ of APE and RPE in meter and localization rate w.r.t. ground truth within MapA for paths

Element	Camera	Nominal			Lighting			Scenery		
		APE_{trans}	RPE_{trans}	%	APE_{trans}	RPE_{trans}	%	APE_{trans}	RPE_{trans}	%
A to B	Azure	**0.0974**	0.0110*	81.9243*	**0.0825**	0.0098*	63.0214*	**0.0930**	0.0152*	70.6892*
	T265	0.1592	**0.0061**	99.1146	0.1504	**0.0060**	**72.2806**	0.1601	**0.0067**	91.8179
	Theta S	0.2149*	0.0077	**99.1612**	0.1599*	0.0077	71.6138	0.2250*	0.0082	**98.6805**
C to B	Azure	**0.0725**	0.0083	69.5875*	**0.0786**	0.0106*	58.7756*	**0.0786**	0.0098	63.2116
	T265	0.1681	**0.0049**	70.0007	0.1765*	**0.0064**	60.5449	0.1738	**0.0062**	61.8020*
	Theta S	0.1690*	0.0162*	**90.0909**	0.1238	0.0101	**61.6005**	0.1783*	0.0171*	**83.0214**
C to D	Azure	**0.0992**	0.0157*	67.6783*	**0.0849**	0.0154*	70.5691*	**0.1035**	0.0259*	69.1554*
	T265	0.1688	**0.0085**	69.8565	0.1467	**0.0089**	70.6285	0.1714	0.0141	72.1434
	Theta S	0.2498*	0.0110	**92.0478**	0.2003*	0.0100	**71.0494**	0.2440*	**0.0135**	**91.4822**

bold highlights the best, * marks the worst.

APE_{trans}, T265 to the lowest RPE_{trans} and Theta S to the highest localization percentage (tied by T265 is 1 case over 9). Overall, Azure is the most sensitive to variations in the scenery whereas Theta S is the most sensitive to lighting variations (Tab. VI). Complementing that, Azure is the less sensitive to lighting condition whereas Theta S is the less sensitive to variations in the scenery. The Time-Of-Flight sensor used to acquire the depth of Azure camera is by nature tolerant to lighting conditions thus making the Azure less sensitive to light variations. However ORB features [37] are known to be resilient to lighting variations but non-invariant. Therefore T265 and Theta S cameras, by only relying on features to estimate poses, are more impacted by high lighting variations than Azure. The decreasing trend of APE_{trans} (Tab. VI) is correlated to the localization percentage decrease since only localized frames are, obviously, considered in APE and RPE. Theta S is the less impacted by Scenery changes thanks to its wide FoV, allowing to extract more non-altered features to estimate poses than T265 and Azure. Then, the narrowest the FoV, the highest the impact of scenery modifications on RPE_{trans} and percentage localization.

Finally, T265's global accuracy is the least impacted by lighting and scene changes and is 2^{nd} for the other metrics.

IX. Conclusion

In this paper we evaluated the performance of Microsoft Azure (wide FoV and RGB-D), T265 (stereo fisheye), Theta S (dual-fisheye) and D435i (RGB-D) using `OpenVSLAM`. The environment was chosen to be representative of common robotic applications: an indoor setting where the environment is susceptible to have light variations or changes in the scene after building maps (e.g. indoor production industrial site, homes, offices, hospitals). We quantitatively evaluated localization within pre-built maps using each of these cameras. Results show that a wide field of view improves significantly the localization rate whereas depth information enhances the global consistency. Lighting variations and changes in the scene have less impact on stereo fisheye camera.

For a camera choice, when it comes to vSLAM, stereo fisheye is the best compromise thanks to its wide field of view and its depth estimation.

As future work, this benchmark will be extended to real indoor industrial and hospital setups that are confined environments, considering locomotion while manipulation by a humanoid robot [38]. Thus, our methodology will be applied in these environments to assess the results while the camera is embedded on a real robot.

TABLE VI: Mean of percentage change w.r.t Nominal case within MapA for paths

Camera	APE_{trans}	RPE_{trans}	%
Lighting			
Azure	-7.10	**4.96**	**-11.45**
T265	**-4.54**	11.23	-13.16
Theta S	-24.05*	-15.58*	-27.41*
Scenery			
Azure	2.74*	40.41*	-6.90*
T265	**1.83**	34.08	-5.27
Theta S	2.63	**11.59**	**-2.98**

bold highlights the min, * highlights the max.

REFERENCES

[1] A. Kheddar, S. Caron, P. Gergondet, A. Comport, A. Tanguy, C. Ott, B. Henze, G. Mesesan, J. Englsberger, M. A. Roa, P. Wieber, F. Chaumette, F. Spindler, G. Oriolo, L. Lanari, A. Escande, K. Chappellet, F. Kanehiro, and P. Rabat, "Humanoid robots in aircraft manufacturing: The airbus use cases," *IEEE Robotics & Automation Magazine*, vol. 26, no. 4, pp. 30–45, Dec 2019.

[2] S. Sumikura, M. Shibuya, and K. Sakurada, "OpenVSLAM: A versatile visual SLAM framework," in *ACM International Conference on Multimedia*, 2019, pp. 2292–2295.

[3] Zichao Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," in *IEEE International Conference on Robotics and Automation*, May 2016, pp. 801–808.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1052–1067, 2007.

[5] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE/ACM International Symposium on Mixed and Augmented Reality*, Nov 2007, pp. 225–234.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[7] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry." in *IEEE International Conference on Robotics and Automation*, 2014, pp. 15–22.

[8] J. Engel, J. Stckler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2015, pp. 1935–1942.

[9] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. on Robotics*, vol. 33, no. 2, pp. 249–265, 2017.

[10] S. Bu, Y. Zhao, G. Wan, K. Li, G. Cheng, and Z. Liu, "Semi-direct tracking and mapping with RGB-D camera for MAV," *Multimedia Tools and Applications*, vol. 76, p. 44454469, 2017.

[11] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras." in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2100–2106.

[12] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "Elasticfusion: Real-time dense slam and light source estimation," *International Journal of Robotics Research*, vol. 35, pp. 1697–1716, 2016.

[13] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-aware learning of maps for camera localization," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[14] G. L. Oliveira, N. Radwan, W. Burgard, and T. Brox, "Topometric localization with deep learning," *Robotics Research*, pp. 505–520, 2020.

[15] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks," in *IEEE International Conference on Robotics and Automation*, May 2017, pp. 2043–2050.

[16] Z. Yin and J. Shi, "GeoNet: Unsupervised learning of dense depth, optical flow and camera pose," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[17] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.

[18] H. Matsuki, L. von Stumberg, V. Usenko, J. Stckler, and D. Cremers, "Omnidirectional DSO: Direct sparse odometry with fisheye cameras," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3693–3700, Oct 2018.

[19] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2015, pp. 141–148.

[20] D. Gutierrez, A. Rituerto, J. Montiel, and J. Guerrero, "Adapting a real-time monocular SLAM from conventional to omnidirectional cameras," in *OMNIVIS workshop, IEEE International Conference on Computer Vision*, 2011, pp. 343–350.

[21] D. Scaramuzza and R. Siegwart, "Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1015–1026, Oct 2008.

[22] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[23] M. Labb and F. Michaud, "RTAB-Map as an open-source lidar and visual SLAM library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[24] Y. Zhao, S. Xu, S. Bu, H. Jiang, and P. Han, "GSLAM: A general SLAM framework and benchmark," in *IEEE International Conference on Computer Vision*, October 2019.

[25] J. Kuo, M. Muglikar, Z. Zhang, and D. Scaramuzza, "Redesigning SLAM for arbitrary multi-camera systems." in *IEEE International Conference on Robotics and Automation*, 2020.

[26] S. Im, H. Ha, F. Rameau, H.-G. Jeon, G. Choe, and I. S. Kweon, "All-around depth from small motion with a spherical panoramic camera," in *European Conference on Computer Vision*, 2016, pp. 156–172.

[27] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization And Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Transactions on Intelligent Vehicles*, 2017.

[28] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: A survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, 2017.

[29] L. Nardi, B. Bodin, M. Z. Zia, J. Mawer, A. Nisbet, P. H. J. Kelly, A. J. Davison, M. Lujn, M. F. P. O'Boyle, G. Riley, N. Topham, and S. Furber, "Introducing SLAMBench, a performance and accuracy benchmarking methodology for SLAM," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5783–5790.

[30] S. Saeedi, E. D. C. Carvalho, W. Li, D. Tzoumanikas, S. Leutenegger, P. H. J. Kelly, and A. J. Davison, "Characterizing visual localization and mapping datasets," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 6699–6705.

[31] L. Liu, Y. Wang, L. Zhao, and S. Huang, "Evaluation of different SLAM algorithms using google tangle data," in *IEEE Conference on Industrial Electronics and Applications*, 2017, pp. 1954–1959.

[32] Y. Watanabe, K. Ramrez-Amaro, B. Ilhan, T. Kinoshita, T. Bock, and G. Cheng, "Robust localization with architectural floor plans and depth camera," in *IEEE/SICE Int. Symposium on System Integration*, 2020.

[33] M. Grupp, "evo: Python package for the evaluation of odometry and SLAM," https://github.com/MichaelGrupp/evo, 2017.

[34] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 7244–7251.

[35] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[36] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns." *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[37] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *International Conference on Computer Vision*, 2011, pp. 2564–2571.

[38] A. Tanguy, D. De Simone, A. I. Comport, G. Oriolo, and A. Kheddar, "Closed-loop MPC with dense visual SLAM– stability through reactive stepping," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 1397–1403.