A survey on real-time 3D scene reconstruction with SLAM methods in embedded systems

Quentin Picard, Stephane Chevobbe, Mehdi Darouich, Jean-Yves Didier

A survey on real-time 3D scene reconstruction with SLAM methods in embedded systems

Quentin Picard, Stephane Chevobbe, Mehdi Darouich, and Jean-Yves Didier

***Abstract*—The 3D reconstruction of simultaneous localization and mapping (SLAM) is an important topic in the field for transport systems such as drones, service robots and mobile AR/VR devices. Compared to a point cloud representation, the 3D reconstruction based on meshes and voxels is particularly useful for high-level functions, like obstacle avoidance or interaction with the physical environment. This article reviews the implementation of a visual-based 3D scene reconstruction pipeline on resource-constrained hardware platforms. Real-time performances, memory management and low power consumption are critical for embedded systems. A conventional SLAM pipeline from sensors to 3D reconstruction is described, including the potential use of deep learning. The implementation of advanced functions with limited resources is detailed. Recent systems propose the embedded implementation of 3D reconstruction methods with different granularities. The trade-off between required accuracy and resource consumption for real-time localization and reconstruction is one of the open research questions identified and discussed in this paper.**

***Index Terms*—SLAM, real-time systems, robot sensing systems, embedded systems, survey.**

I. Introduction

Autonomous transport systems, such as cars, service robots, UAVs/MAVs (unmanned/micro air vehicle) and mobile AR/VR (augmented reality/virtual reality) devices, require accurate and robust perception for high-level functions based on obstacle avoidance or interaction with its physical environment. Each transport system has several constraints with a different level of criticality for real-time processing. Autonomous cars offer much more space to include powerful and expensive computing hardware [1] than drones or AR/VR devices where the power consumption budget [2] and the robust localization [3] are critical.

Simultaneous Localization And Mapping (SLAM) is an active area of research and is widely used by the community to provide accurate and robust real-time localization and reconstruction of the surrounding environment without prior knowledge. It is reflected in three main questions [4]: localization (where am I?), reconstruction (how is my environment?) and image segmentation (what are the objects around me?). The main challenge of SLAM is the global consistency. As it is mostly based on relative measurements from visual and inertial sensors, uncertainty accumulates gradually and the effect of drift begins to be noticeable over time. SLAM methods include an optimization module responsible for local and global consistency. The loop closure detection corrects the drift when a reconstructed scene has already been visited. Depending on the used techniques, SLAM offers several types of reconstruction, such as point clouds, surfels (oriented points), meshes (triangle meshes), volumetric (surface meshes based on voxels). This paper defines the term mapping as cartography for point clouds [5], [6] and 3D reconstruction for mesh [7] and volumetric models [8].

While visual(-inertial) SLAM takes advantage of imaging sensors and inertial measurements, several lines of research make use of other sensors. In [9], [10], a multi-sensor system based on LiDAR [11] and a monocular infrared camera [12] is used for real-time localization. In [13], a hybrid state estimation pipeline combines event-based sensors, visible cameras and inertial measurements. Event-based sensors overcome the limitations of visual cameras against rapid movement or changes in lighting. Instead of capturing directly the light intensity, they acquire the change of intensity in the scene [14]. Today, they are mainly used as a complementary sensor to take advantage of the large amount of information provided by visible cameras.

Existing surveys on SLAM have reviewed the fundamental challenges for accurate and robust large-scale applications [15], [16], [17], from early probabilistic approaches and data association [18], [19] to the potential use of deep learning [20]. SLAM components, including sensors to the embedded localization [21] have been intensively studied to provide a robust solution to many applications, like autonomous driving [22], search and rescue tasks, infrastructure inspection and 3D reconstruction in static and dynamic environments [23] with challenging conditions [24]. The robustness of real-time methods under difficult conditions has been reviewed and quantified including low visiblity [24], dynamic movement, illumination changes, changed viewpoints and lifelong scenarios [25]. Experiments [25] highlight that state-of-the-art approaches struggle with these challenging conditions. It also shows that the SLAM method based on feature extraction [5] provide the best trade-off in terms of robustness. However, feature extraction for visual SLAM accentuates the lack of flexibility due to the dependence of a certain type of feature and the difficult localization in the presence of noise [26]. From the real-time localization to global mapping, the SLAM problem has been intensively studied using learning-based approaches [20]. The evolution of deep neural networks (DNNs) and its impact on SLAM [15] opens several directions for lifelong scenarios including the type of model, scalability, and hardware deployment.

This survey provides a broader view of SLAM, from localization to 3D reconstruction in the embedded context, with an in-

Q. Picard, S. Chevobbe and M. Darouich are with CEA, LIST, 91191 Gif-sur-Yvette, France. (e-mail: quentin.picard@cea.fr)

Q. Picard and J-Y. Didier are with IBISC, Univ Evry, Université Paris-Saclay, 91025, Evry, France.

depth analysis of the implementation of advanced functions on resource-constrained hardware platforms. It focuses on methods using lightweight and low power consumption imaging sensors and inertial measurements. This article presents the following main contributions:

- The description of each function for real-time localization and 3D reconstruction based on imaging sensors and inertial measurements in Section II. A discussion about strengths and limitations of existing visual(-inertial) SLAM methods as well as the potential use of deep learning is provided.
- A comprehensive review of the implementation of localization and reconstruction functions in low power consumption embedded systems with limited resources in Section III.

II. SLAM pipeline from sensors to 3D reconstruction

The SLAM community has made remarkable improvement on the accuracy and robustness of large-scale applications in the recent years [16], [17], [15].

Figure 1 illustrates a conventional 3D scene reconstruction pipeline using imaging sensors and inertial measurements as inputs. It is decomposed in two main modules, localization and 3D reconstruction. The first one is based on the front-end (FE) and the back-end (BE). The FE processes images and estimates motions. The BE uses preintegrated inertial measurements [28] and manages the topological consistency through local and/or global optimizations of 3D poses and points [29]. The 3D reconstruction module provides a model of shapes and gives geometric properties. The mesh reconstruction allows a 3D mesh of the environment based on a 2D triangulation from FE and 3D poses provided by BE. This representation is useful for obstacle avoidance functions. The interaction with the physical environment requires a volumetric reconstruction provided by the voxel-based computation and the rendering of surface meshes. It takes as input the depth map from the image depth estimation and 3D poses from the BE. The semantic segmentation gives information about surrounding objets [30], [31], [32]. For instance, in the case of a 2D RGB images, its semantic corresponds to a labelled classification of each pixel, used for volumetric reconstruction, mesh and point cloud cartography.

This section describes each part of the pipeline from real-time localization to 3D scene reconstruction and the potential use of deep learning.

A. Localization module

Real-time 3D reconstruction for mobile robots in an unknown environment requires an accurate localization that visual SLAM [5], [6], [33], [34] and the subpart visual(-inertial) odometry methods (VO/VIO) [7], [35], [36] provide from relative measurements.

1) Front-end: The front-end processes the input image. Indirect and direct techniques are the two main approaches, which are respectively described below.

Indirect methods consist of feature detection, feature matching (or tracking) and motion estimation from observations with geometric verification based on n-point random sample consensus (RANSAC) [37], [38], [39]. Feature detection extracts corners on an n gaussian pyramidal levels generated from the grayscale image. The process of the gaussian pyramid levels is downsizing an image through n levels with the same factor (640×480 to 80×60 in four levels with a factor of two) [40]. In [5], a scale factor of 1.2 accross 8 levels in the scale pyramid has been configured by default. This technique allows the feature extraction to be scale invariant. Several feature detection methods have been developed since Moravec [41] which give a trade-off between robustness and computing complexity [42], [43], [44]. Feature matching involves detecting and describing features to match consecutive images [45], [46], [47] whereas tracking with the KLT tracker estimates the displacement based on the detected features without any feature description [48]. The next step estimates the motion between two images and performs geometric verification with RANSAC for outlier rejection. Depending on the approach, features are specified in two or three dimensions for 2D-2D, 3D-3D and 3D-2D techniques. The first 2D to 2D case estimates the motion from the features extracted in 2D. This method is usually performed during the initialization using epipolar constraints to obtain geometric relations between two consecutive images [37], [39], [49]. In the case of 3D to 3D motion estimation, features are specified in three dimensions based on the triangulation of 3D points by using, for instance, stereo visible cameras. The 3D-3D estimation is based on minimizing the Euclidean distance between corresponding 3D features. Finally, the 3D to 2D motion estimation approach is based on the perspective from n points (PnP) which minimizes the reprojection error between the 2D feature and its 3D counterpart:

$$\mathbf{T}_k = \underset{T}{argmin} \sum_i \|u'_i - \pi(p_i)\|^2 \tag{1}$$

where $\mathbf{T}_k$ is the transformation matrix between the current view point and an arbitrary origin, u'_i the detected features, π the projection function and p_i the corresponding 3D point. As pointed in [50], the 3D to 2D motion estimation is more accurate than 3D-3D estimation because of the uncertainty generated by the triangulated 3D points.

While indirect methods extract and use features for the pose estimation, direct methods use all pixels available in the input image [6], [51], [52], [53]. The pipeline differs as they do not contain the feature detection and matching stages. It estimates an optical flow with the photometric error minimization:

$$\mathbf{T}_{k,k-1} = \underset{T}{argmin} \sum_i \|I_k(u'_i) - I_{k-1}(u_i)\|^2 \tag{2}$$

where $\mathbf{T}_{k,k-1}$ is the transformation matrix between two frames, $I_k(u'_i)$ the intensity I of the pixel u'_i in frame k.

Table I summarizes the benefits and drawbacks of indirect and direct approaches. Indirect methods allow robust and efficient computation due to a sparse detection of features. However, they rely on detected features and require robust estimation to prune outliers that impair the localization process. Direct methods have the benefits to estimate motion

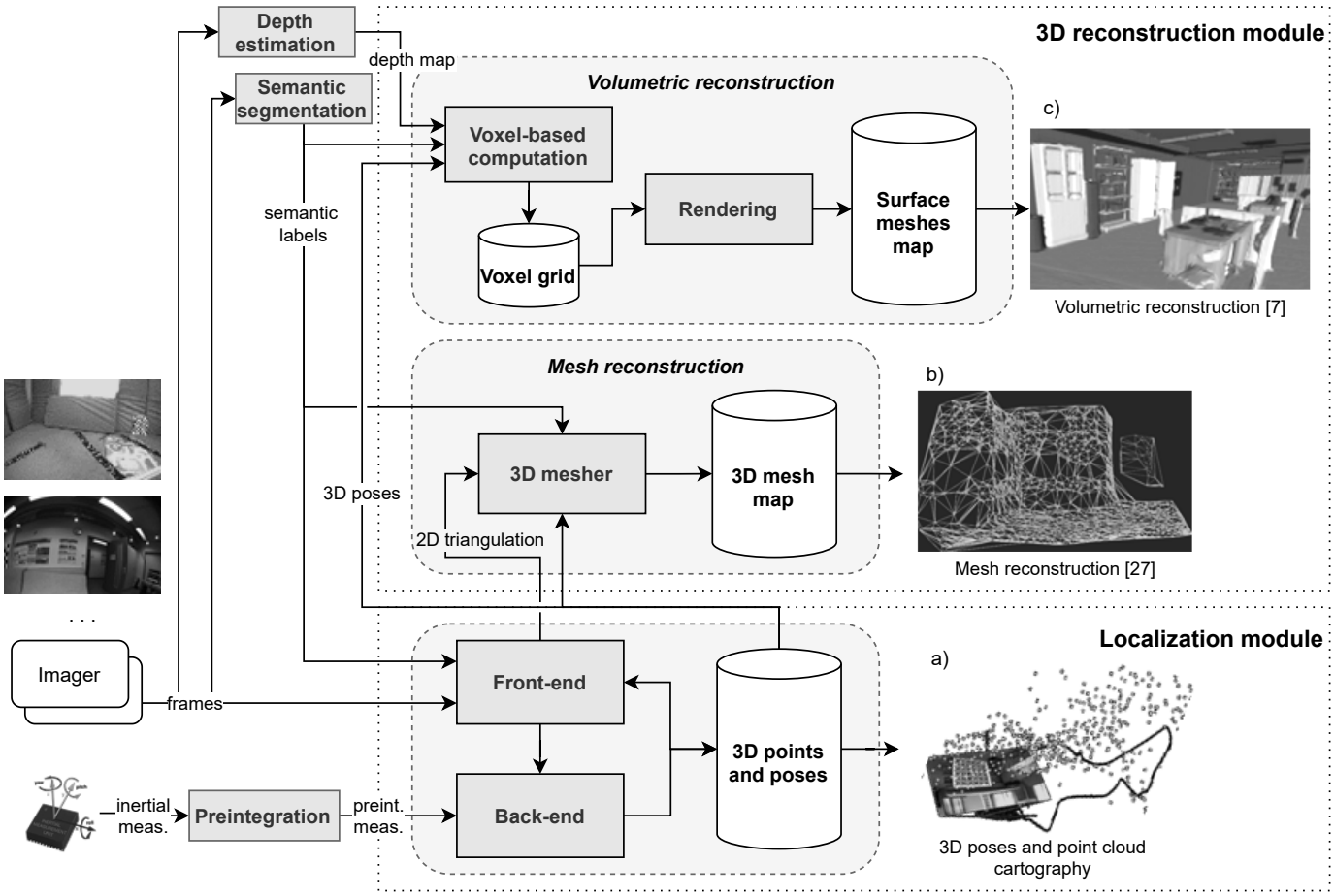

Fig. 1. Conventional real-time 3D scene reconstruction pipeline from imaging sensors and inertial measurements. a) represents an accurate trajectory estimation and a sparse point cloud scene representation. b) corresponds to the visualization of a 3D mesh introduced in [27]. c) is the volumetric voxel-based reconstruction provided by [7].

by minimizing a photometric error. However, they require an accurate photometric calibration. In [52], it takes into account exposure time, a non-linear response function and lens vignetting which allows a comprehensive model of the brightness transformation. In the presence of a large amount of intensity gradients from the image, low computational speed is one of the drawbacks for direct methods. Dense representations with direct image alignement used the commodity of GPU hardware to achieve real-time performances [51], [54]. In order to reduce the computation time, a semi-dense depth filtering [55] has been developed on CPUs [6], [56]. In [52], a direct and sparse model uses only a selection of well-distributed points in the image with a high image gradient magnitude.

In order to combine the strengths of direct and indirect methods, semi-direct approaches exploit pixels with strong gradients while relying on sparse features [57], [58].

Real-time processing for these approaches is achieved through the use of keyframe (KF)-based techniques to provide a balance between accuracy and efficiency [59]. The front-end module estimates if an input frame is considered as a keyframe for back-end optimization. Poses and 3D points are produced over time by these keyframes to obtain a local and/or global consistency. Keyframes are generated at a lower framerate (8-16 FPS) than input frames (30-60 FPS).

2) Back-end: In early work, SLAM methods were composed of, an extended Kalman filter (EKF) [60], [61], [62] and with a separate back-end for bundle adjustment [59] and in the form of a node graph [63], [64], [65]. The optimization back-end function is a crucial step to obtain an accurate pose estimation related to constraints of new measurements. It consists of optimizing the map composed of 3D poses and points in a global or local manner. This subsection addresses the challenge of optimization as follows: First, the input of inertial measurements in the BE. Second, the approaches used to optimize states and 3D points of the map. Then, the functions to provide global map consistency, which is identified as a fundamental key topic in SLAM [15], [66].

In a visual-inertial system, inertial measurements are preintegrated to improve localization by solving the scale factor problem through linear acceleration and angular velocity between frames. IMUs provide data at high rate (100 Hz to 1 kHz) compared to the framerate (20-60 FPS). To cope with the acquisition difference, a preintegration method is included in a real-time localization pipeline that analyses and consolidates all inertial data between two keyframes into a single measurement [28]. This technique uses a structureless approach which avoids optimizing over the 3D points to

TABLE I
DESCRIPTION, BENEFITS AND DRAWBACKS OF BOTH INDIRECT AND DIRECT APPROACHES USED IN STATE OF THE ART METHODS.

	Indirect approach	**Direct approach**
Description	- Feature extraction - Feature matching or tracking - Reprojection error minimization	- Uses all pixels - Photometric error minimization
Benefits	- Robust and fast due to feature extraction - Allows sparse reconstruction with efficient computation	- More robust and accurate than indirect - Allows dense scene representation
Drawbacks	- Dependant to features for matching - Requires robust estimation to overcome outliers	- Longer computation time due to optical flow estimation - Requires a good initialization - Requires an accurate photometric calibration

improve the computation time during optimization.

In order to fuse visual and inertial data and optimize states generated by the localization module, two main approaches are used, filtering-based or factor graph [67]. In a filtering system, only the latest state is estimated. The EKF fuses data provided by different sensors, predicts the future state with respect to the initial estimation and updates the prediction. The uncertainty is represented using a covariance matrix. The complexity of this approach increases over time as the generated map and estimated 3D points become larger. Therefore, the memory consumption increases significantly in addition to the computational cost. In [34], real-time processing has been reached by taking into account only a small number of features. In [68], direct photometric errors have been used within the EKF update and employed a numerical minimal distance representation of features to address the computational issue. Alternative methods based on the Multi-State Constraint Kalman Filter (MSCKF) framework [36] use a structureless strategy that marginalizes the 3D points [69], [70].

Another optimization technique is the factor graph [71], which represents all states, points, data related to each other in the form of a nonlinear graph [72], [73], [74], [75]. This graph is solved to optimize local (*fixed-lag smoothing*) and/or global (*full smoothing*) states and 3D points while exploiting the sparsity of SLAM algorithms. In a fixed-lag smoother configuration, only the poses within a sliding time window are optimized while a full smoother takes into account the full history of poses. The optimization based on the factor graph is described as follows:

1) Linearization of the factors in the graph (IMU, visual data, etc.) in a linear equation system

$$H\Delta x = \varepsilon \tag{3}$$

where H is the Hessian matrix, ε is a vector that describes how front-end measurements affect the state of each keyframes and Δx, a vector describing the updated states.
2) Use of the Cholesky factorization and back-substitution to solve the linear equation system.
3) Marginalization of states outside the sliding temporal window for local optimization to maintain real-time performances.
4) Solutions of the linear system are used to update the remaining states of keyframes.

In [76], a set of non-linear visual-inertial information about the motion estimation between keyframes are recovered from the first layer odometry and combined using a global bundle adjustment. Based on a nonlinear optimization method [77], VINS-Fusion [35] supports the use of multiple sensors (camera, IMU, GPS, etc.) integrated in a pose graph structure as a Maximum Likelihood Estimation (MLE) problem. In [78], the nonlinear optimization integrates both the reprojection errors and a temporal error term from inertial measurements while old keyframes are marginalized from the optimization window to ensure real-time processing. The odometry module of Kimera-VIO [7] is based on the IMU preintegration approach [28] and provides SLAM capabilities with a pose graph optimization module responsible for loop closures.

Global consistency in SLAM is provided with loop closures to detect that a scene has already been visited and to correct the accumulated drift. Without it, visual SLAM becomes visual odometry for local consistency [15]. Loop closure is performed in three main steps:

1) Detect candidates between newly created features from poses and the current active map
2) Correct the detected poses affected by the loop closing
3) Optimize the map in order to verify if the accumulated drift has been corrected

The mainstream method for loop closure is the use of a bag-of-words (BoW) vector. It implements a recognition database of visual words vocabulary describing image features [79]. Therefore, the recognition database is queried to find loop candidates. In terms of timing, loop closures of the visual-inertial ORB-SLAM3 system [5] takes around 10ms for the detection, 124.77ms for the correction, which includes the loop closure and the correction of the whole map and 1529.69ms for the full map optimization. In order to maintain real-time performances, the latter is only performed if the number of keyframes to be optimized is below a fixed threshold. A common conclusion is that the extraction and comparison of feature descriptors require too much computational resources. Therefore, alternatives methods have been developed, such as the extraction of the shape of each object as binary content [80] and edge-based verification for loop closures [81]. The latter provides an average accuracy of 2.36 cm, which is slightly worse compared to 1.14 cm with ORB-SLAM2 [82] on RGB-D dataset [83].

B. 3D reconstruction module

While the localization module provides a point cloud cartography, the 3D reconstruction module considers two types of map representation, the mesh and the volumetric reconstruction based on voxels to produce surface meshes in real-time.

1) Mesh reconstruction: Mesh representation is widely used to model surfaces, shapes and provides a topology of the scene based on points [7] or surfels [84]. The Delaunay triangulation has been used for many applications, notably in computer vision [85] to provide accurate mesh reconstruction and cover potentially usable planar surfaces [86]. Several algorithms derive the Delaunay triangulation in two [7], [27], [87], [88] and three [89] dimensions. In 2D, this technique follows the empty circle property, which generate a triangle when the circle is the only one to pass through all three vertices. Therefore, there are no vertices in it. The triangulation maximizes the minimum angle of each generated triangles. Thus, the reconstruction ensures the consistency of triangulations and high quality meshes.

In [27], a lightweight Delaunay mesh [90] is generated with a keyframe-less approach and monocular depth estimation for MAVs. The reconstruction provides a per-frame mesh reconstruction without taking into account the previous meshes for fast computation. The approach was able to process each frame onboard an MAV with an Intel Skull Canyon NUC flight computer at over 90 Hz. In [7], the multi-frame mesh gives a single mesh built over time on the basis of fusing per-frame meshes. On an Intel Xeon CPU E3-1505M v6 3 GHz, the multi-frame mesh is generated at around 67 Hz. This configuration shows that a reliable map composed of a large mesh built over time is maintained and updated with each new mesh generated in real-time.

2) Volumetric reconstruction: A volumetric reconstruction is defined as a voxel-based computation in which a surface mesh is rendered from the volume to allow a detailed 3D model [91].

Voxel-based computation: A common method to quickly and accurately represent surfaces is the truncated signed distance field (TSDF) [92]. This technique represents the 3D environment as a grid of voxels. In a TSDF volume, which integrates depth data, the value of each voxel corresponds to the signed distance to the nearest surface. Positive and negative values correspond respectively to voxels outside or inside the volume. Thus, the surface is defined by the isosurface boundary between negative and positive values (zero-crossing). Beyond a certain distance, the information becomes irrelevant. Therefore, the distance is truncated to take advantage of the values close to the surface.

Each voxel stores a truncated signed distance D and weight values W updated for each 3D point p in the volume from frames $1...k$.

$$D(p) = \frac{\sum w_k(p) d_k(p)}{\sum w_k(p)} \tag{4}$$

$$W(p) = \sum w_k(p) \tag{5}$$

where d is the signed distance and w the weight function of the sensor measurements. The cumulative $D_k(p)$ and $W_k(p)$ are expressed as follows [92]:

$$D_{k+1}(p) = \frac{W_k(p) D_k(p) + w_{k+1}(p) d_{k+1}(p)}{W_k(p) + w_{k+1}(p)} \tag{6}$$

$$W_{k+1}(p) = W_k(p) + w_{k+1}(p) \tag{7}$$

The choice of the weighting have a strong impact on the accuracy of the representation as it models the uncertainty of surface measurements [8]. Therefore, the greater the uncertainty, such as noisy data, the higher the weighting. For instance, to allow real-time 3D reconstruction, KinectFusion [93] adopted a common constant weight approach $w_k(p) = 1$ resulting in a simple average.

$$W_{k+1}(p) = min(W_k(p) + w_{k+1}(p), W_{max}) \tag{8}$$

In [8], a comparison of two weighting strategies is proposed. It consists in comparing the 3D reconstruction qualitatively and quantitatively with a constant and a quadratic weight. Equation 9 defines the truncated distance so that $\delta = 4v$ and $\epsilon = v$, where v is the voxel size.

$$w(p) = \begin{cases} \frac{1}{z^2}, & -\epsilon < d \\ \frac{1}{z^2}\frac{1}{\delta - \epsilon}(d + \delta), & -\delta < d < -\epsilon \\ 0, & d < -\delta \end{cases} \tag{9}$$

Qualitatively, the quadratic weighting strategy provides a better representation of the structure with less error than constant weights. Higher level of robustness with less distorsion can be observed with a voxel size v ranging from 0.02m to 0.2m.

Rendering: In order to render the voxel 3D perspective, two main approaches are used for mesh extraction, raycasting and projection mapping [94]. Raycasting casts a ray from sensor to the truncated signed distance behind the 3D point within the voxel and extracts the zero-crossing from the TSDF values to obtain an approximate depth rendering [92], [95]. Projection mapping calculates the distance between the center of the voxel and the depth value in the image. The amount of data processed by the projection mapping technique is bounded by the number of voxels while the raycasting one is potentially unbounded as it includes the number and the length of rays being integrated. Timing results shows that raycasting is faster than projection mapping with a required time of about 62ms and 106ms respectively on a Tango Yellowstone tablet device with quadcore CPU, 4GB RAM and a Tegra K1 GPU [94].

In pioneer work [93], the Kinect sensor has been used to provide RGB frames and depth measurements at a 640×480 resolution. It uses the TSDF integration and 3D rendering with raycasting. The 6DoF pose is computed with the iterative closest point (ICP) approach [96]. The volumetric reconstruction is limited in area of a 512^3 volume. The implementation on NVIDIA GeForce GTX 560 updates TSDF values at 2ms [97] and the main system operates in real-time at 40 FPS with a resolution of 512^3. Fully implemented with CUDA in order to use NVIDIA GPUs, this method is not memory efficient with 512 MB allocated for 32-bit voxels in the restricted

space. A voxel hashing method [98] has been used in [99] to obtain a memory efficient and large scene representation. The algorithmic approach introduced intermediate online visualization to speed up the calculation. While raycasting is performed for each frame in a full configuration, a forward projection parameter uses the most recent raycasting result. After a fixed threshold of n frames a full raycast is performed. Implemented on NVIDIA Tegra K1 and Apple iPad Air 2, the method achieves a computation time per frame of 21.04ms and 48.43ms respectively.

In [8], the same hierarchical data structure with voxel hashing [98] as well as a grouped raycasting approach has been used. The latter merges the cast point with all other 3D points to the same voxel to perform raycasting only once, on the average position. It needs around 55ms and 5ms to process a voxel size of 0.05m and 0.20m respectively compared to 250ms and 100ms with a standard approach.

C. Learning-based modules in a conventional pipeline

Advanced research in deep neural networks (DNN) allows the use of a large set of data while providing accurate and robust results for many applications [100]. This subsection provides an overview of DNN methods for real-time localization, a comprehensive description of intermediate representations and the complexity of deep learning in the context of embedded systems.

1) Deep learning for real-time pose estimation: Several research papers discussed the use of neural networks for real-time pose estimation with an end-to-end or hybrid implementation. The first one only relies on deep learning to estimate the pose from consecutive images. In [101], a convolutional neural network (CNN) [102] was used to extract features from RGB images, and a recurrent neural network (RNN) to model sequential information. Therefore, long short-term memory (LSTM) is used for the sequential representation with a retained memory of previous hidden states. In the case of a visual-inertial configuration, a multi-rate LSTM has been implemented to process inertial measurements [103]. End-to-end methods based on supervised [101] or unsupervised [104] learning are evolving, but the performances of current approaches are limited by training data to provide an accurate and robust estimation in all situations.

A hybrid pipeline uses deep learning for specific back-end (BE) and front-end (FE) functions, including local [105], [106] or global [107], [108] BE optimization and FE feature extraction [109], [110], [111], [112]. In [113], results based on a CNN detector and descriptor demonstrated better distributed features with a lower number of detection compared with ORB [47]. The extraction runs at 40 FPS while the SLAM pipeline based on ORB-SLAM2 [82] runs at 20 FPS on Jetson TX2 with a tiny version of the network. The accuracy of the hybrid method deteriorates for sequences that require fine details, like the *fr1_floor* and *fr1_360* of the TUM RGB-D dataset [83] due to the scale of feature maps.

2) Intermediate representations based on neural networks: Intermediate representations have been proven useful for high-level functions based on pixels-to-action models. In a visual-based urban driving task where an agent has to reach a target location, success accuracy improved by around 15% with the addition of depth estimation and by 20% with semantic segmentation compared to an RGB image only [32].

Semantic segmentation: In order to recognize surrounding objects, the image segmentation, like semantics, allows a set of objects to be labelled at pixel-level [30], [114].

In [115], the concept of semantics for object recognition and semantic segmentation in SLAM is detailed and argues that it provides effective solutions for data association [116] and long-term consistency [117] to obtain a reliable localization. Deep learning techniques enhance the capabilities of SLAM by using semantic segmentation for local loop closure detection and medium-term consistency [116], [118] or for handling dynamic environments. As conventional static SLAM considers dynamic features as outliers, semantics is used to label dynamic and static elements to improve the accuracy and robustness of localization [119], [120]. Semantic extraction has also known active research for the segmentation of the map, which leads to real-time object recognition based on point clouds [121], surfels [122], [123], meshes [7], [124] and voxels [7], [125], [126] representations. Semantic reconstruction is achieved by two main approaches, view-based or map-based labelling [127]. Semantic extraction per view processes raw 2D input image data but performs unnecessary computations from one view to another whereas the segmentation of the whole map involves a 3D CNN. It avoids redundant computation but depends on the quality of the reconstruction. Using the mean intersection over union (IoU) metric, the segmentation achieves an accuracy of 89% and 92% after 1000 frames for view-based and map-based respectively [127].

Several questions remain open, including the accuracy of the segmentation required for a 3D reconstruction pipeline and its impact on the precision of the reconstruction. The complexity of semantic generation in terms of computational cost and use of hardware resources for 2D or 3D segmentation tasks is also part of future open directions in the embedded context.

Depth estimation: Deep neural networks have been used for depth estimation of monocular images [106], [128], [129]. Depth estimation based on a CNN overcomes the limitations of monocular cameras by handling pure rotational motions [128] and resolving the scale factor problem by updating the transformation matrix with the scale parameter α [130]:

$$\mathbf{T}_{k,k-1} = \begin{bmatrix} \mathbf{R}_{k,k-1} & \alpha \mathbf{t}_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{10}$$

where $\mathbf{T}_{k,k-1}$ is the transformation matrix between two frames (rotation $\mathbf{R}_{k,k-1}$, translation $\mathbf{t}_{k,k-1}$) and α the scale parameter defined as a maximum likelihood estimation taking into account the estimated depth value of each pixel.

Although this representation has several benefits in the case of monocular SLAM, its accuracy remains limited for long sequences, compared to traditional stereo methods with a strongly drifting trajectory [131]. In [132], the depth map has been estimated from the fusion of a dense stereo pipeline with a monocular depth estimation based on a self-supervised CNN with data generated from the sensors. Qualitatively, the fused map refines the noisy result of the stereo vision and gives the global composition of the image provided by the

monocular estimation. The median absolute error between depth estimation and the ground truth (GT) depth distance for the stereo and mono is less than 5 m and around 20 m respectively for a GT of 80 m and around 1 m and less than 4 m respectively for a GT of 10 m.

In the embedded context with restrictive constraints, the right trade-off between required depth accuracy for 3D reconstruction and computational cost in terms of memory usage and processing time remains the main open line of research.

3) Complexity of deep learning in the context of embedded systems: The complexity of deep learning models for real-time localization and reconstruction can be characterized by three main points: the type of model, the use of large amounts of data for scalability, the resource consumption and hardware implementation for real-time processing. End-to-end models are currently computationally expensive, as they involve the sequential representation of previous states. Hybrid models have the advantage of combining the strengths of deep learning with the maturity of conventional methods for specific functions to overcome several sensors limitations and to compute intermediate representations for high-level understanding of the surrounding environment.

The accuracy of deep learning methods is highly dependent on the training data. However, in order to obtain a reliable system, the training stage must learn from many situations, especially for real-time localization. Visual odometry methods based on DNNs can fail, resulting in low accuracy if the model is not well-fitted [101]. The majority of the localization methods are trained and evaluated on the KITTI benchmark [133], which provides car driving trajectories without significant rotation changes. This can lead to significant errors in different situations.

In addition to the time and resources required for the training stage of deep neural networks, specific hardware platforms allow real-time processing for DNN inference, especially with the parallelization of computations for better use of GPUs. Hybrid pipelines take advantage of both the CPU and the GPU for geometric constraints and neural network calculations, respectively [106]. For real-world deployment, size, power consumption and resource constraints must be considered. In [134], feature extraction with a CNN has been accelerated on FPGA in fixed-point number [135] and the visual odometry network [136] is run on CPU. This implementation achieves an execution time of 5ms on FPGA and 340ms on CPU with a monocular camera at 20 FPS.

D. Overview of existing methods

This subsection provides a description of the benchmarking tools for qualitative and quantitative evaluation of state-of-the-art methods. It also highlights the techniques used, from real-time localization to DNNs, in existing approaches.

1) Benchmarking tools: Many benchmarking tools present various sequences recorded at different locations for evaluation of visual-based methods under challenging conditions such as illumination changes, long trajectories, different motion speeds and dynamic environments.

Depending on the transport use cases, public datasets provide outdoor environments for autonomous cars with densely populated area [137], [138], diverse urban environments [139] and large scale sequences with different speeds up to 80 km/h [133]. Dynamic and long-term environments with illumination changes [140], [141] are available for service robots, where human activity is also included [142]. Challenging motions indoors and outdoors are provided with MAV's datasets through aggressive trajectories [143], fast motion and motion blur [144] and in urban streets at low altitude (5-15 m above the ground) [145].

Data recorded with a handheld sensor is particularly useful for AR/VR applications in dynamic sequences [146] and global/local illumination changes [147]. The TUM datasets provide several handheld trajectories with many imaging sensors for short and long sequences [148], [149], [150]. Capturing data in real-world scenes has several limitations [151] that can be overcome with synthetic sequences. The accuracy of surface reconstruction with RGB-D information under realistic [152] and challenging conditions [147], [153] is also evaluated on the basis of synthetic datasets.

The evaluation of the estimated trajectory and the accuracy of the scene reconstruction requires ground truth usually provided by an RTK-GPS or motion capture systems.

2) Performance comparison of existing methods: Table II provides an overview of existing visual(-inertial) odometry, SLAM and 3D reconstruction methods. It highlights the strategies employed for the localization front-end (FE) and back-end (BE), the type of generated representation from cartography (point clouds and surfels) to volumetric reconstructions, the potential use of DNNs and the hardware (HW) implementation. The FPS measurements correspond to the performance of the entire pipeline.

The localization strategy shows that most methods employ a tracking front-end that refers to the KLT algorithm [48]. It allows a faster computation for local optimization compared to feature detection and description (Det.+Des). In [7], tracking takes an average of 4.5ms with 300 features per frame while Det.+Des takes around 15ms to extract 1200 ORB features per frame [5]. However, the latter allows loop closures by the description of features.

Most of the methods are based on the graph back-end to exploit the sparsity of SLAM and to provide more accurate estimated trajectories [159] compared to filtering techniques. The hardware implementations highlight the complexity of each approach for embedded platforms. For instance, the MSCKF-based [161] provides a higher performance in FPS than VINS-Fusion [35], which is based on a graph BE on UP Board with 20 FPS and 7 FPS respectively. However, the SVO graph-based method [58] runs at 40 FPS on the same platform. The front-end parameters, multithreading strategies, the sparsity of the map reconstruction used are important factors to take into account for real-time performance. As it can be seen, several 3D reconstruction methods [27], [84], [87], [89], [8], [126] do not provide a localization strategy (referred as: -), which means that they take advantage of 3D poses generated by SLAM methods.

Some approaches are based on deep learning in an end-to-end or hybrid configuration. At the moment, end-to-end pipelines do not achieve the performances of conventional

TABLE II

EXISTING VISUAL(-INERTIAL) ODOMETRY, SLAM AND 3D RECONSTRUCTION METHODS WITH THE POTENTIAL USE OF DEEP NEURAL NETWORK (DNN) AND THE HARDWARE (HW) IMPLEMENTATION.

Methods	Year	Sensors				Localization strategy		Type of representation			Usage of DNN	HW implementation
		Visible cam.		IMU	Others	Front-end	Back-end	Carto.	Mesh	Volum.		
		mono	stereo									
DeepSLAM [104]	2021	✓	✓	-	-	End-to-end DL model		Dense	-	-	Depth (48ms), poses (25ms), loop (120ms)	i7-6820HK 2.7GHz, GeForce GTX 980 M (20 FPS)
ORB-SLAM3 [5]	2020	✓	✓	✓	-	Det.+Des.	Graph	Sparse	-	-	-	[82] on Jetson TX2 (28 FPS) [154] [82] on Pi 3B+ (6 FPS), Jetson Nano (10 FPS) [155]
Basalt [76]	2020	-	✓	✓	-	Tracking	Graph	Sparse	-	-	-	E5-1620 CPU (19 FPS global opti., 128 FPS local opti.)
OpenVINS [70]	2020	✓	✓	✓	-	Tracking	Filtering	Sparse	-	-	Depth estimation [156]	Jetson TX2 (36 FPS), Jetson Nano (28 FPS) [156]
DXSLAM [109]	2020	✓	-	-	-	Det.+Des.	Graph	Sparse	-	-	Feature extraction (46.2ms w/ opti.)	Core i7-10710U (15W)
[9]	2020	✓	-	✓	IR+LiDAR	Det.+Des.	Graph	Sparse/Dense	-	Offline [157]	-	NI IC-3173 (10 FPS visual-SLAM, 20 FPS lidar-SLAM)
SurfelMeshing [84]	2020	-	-	-	RGB-D	-	-	Surfels	✓	-	-	Core i7 6700K, Geforce GTX 1080 (178 FPS for mesh)
Kimera [7]	2020	✓	✓	✓	-	Tracking	Graph	Sparse	✓	✓	Semantic segmentation	Jetson TX2 [153]
DeepFactor [106]	2020	✓	-	-	-	Semi-direct	Graph	-	-	✓	Depth estimation	GTX 1080 GPU (network, camera tracking) CPU (geometric error factors)
[111]	2020	✓	✓	-	-	Det.+Des.	-	-	-	-	Feature extraction (59ms)	Xilinx ZCU102 FPGA
ST-VIO [118]	2020	✓	-	✓	-	Direct	Graph	-	-	-	Semantic segmentation [158]	Jetson TX2
[134]	2020	✓	-	-	-	End-to-end DL model		-	-	-	Feature extraction on FPGA (5ms) VO on ARM CPU (340ms)	Xilinx ZCU102 MPSoC
GCN-SLAM [113]	2019	-	-	-	RGB-D	Det.+Des.	Graph	Sparse	-	-	Feature extraction (25ms)	Jetson TX2 (20 FPS)
VITAMI-E [88]	2019	✓	-	-	-	Tracking	Graph	Dense	✓	✓	-	Core i7-7820 HQ CPU
RESLAM [81]	2019	-	-	-	RGB-D	Edge	Graph	Sparse	-	-	-	i7-4790 desktop computer with 32 GB RAM
[110]	2019	✓	✓	-	-	Det.+Des.	Graph	Sparse	-	-	Feature extraction (around 16ms)	GeForce GTX Titan X
VINS-Fusion [35]	2019	✓	✓	✓	LiDAR	Tracking	Graph	Sparse	-	-	-	[77]: UP Board (7 FPS), ODROID XU4 (7 FPS) [159]
PanopticFusion [126]	2019	-	-	-	RGB-D	-	-	-	-	✓	Semantic segmentation (315ms) [160]	Image segmentation on GPU
MSCKF-based [69]	2018	-	✓	✓	-	Tracking	Filtering	Sparse	-	-	-	[161]: UP Board (20 FPS), ODROID XU4 (20 FPS) [159]
[89]	2018	-	-	-	-	-	-	-	✓	-	-	Core i7-4770S, 3.10 GHz with [162]
DS-SLAM [120]	2018	-	-	-	RGB-D	Det.+Des.	Graph	-	-	✓	Semantic segmentation (37.6ms) [163]	Intel i7 CPU, P4000 GPU
CNN-SLAM [128]	2017	✓	-	-	-	Direct	Graph	Dense	-	-	Semantic segmentation [164] Depth estimation [165]	Intel Xeon CPU, 2.4GHz Quadro K5200 GPU for CNN networks
FLaME [27]	2017	✓	-	-	-	-	-	-	✓	-	-	Intel Skull Canyon NUC (90 FPS) with [166]
DeepVO [101]	2017	✓	-	-	-	End-to-end DL model		-	-	-	Feature extraction LSTM pose estimation	Training on Tesla K40 GPU
VINet [103]	2017	✓	-	✓	-	End-to-end DL model		-	-	-	Features extraction (160ms) IMU LSTM (5ms) and Core LSTM	Training on Tesla K80 GPU
Voxblox [8]	2017	-	-	-	RGB-D	-	-	-	-	✓	Semantic segmentation [125]	i7 2.1 GHz CPU [167]
SVO [58]	2016	✓	✓	✓	-	Semi-direct	Graph	Sparse	-	-	-	UP Board (40 FPS), ODROID XU4 (50 FPS) [159]
DSO [52]	2016	✓		-	-	Direct	Graph	Sparse	-	-	-	i7-4910MQ CPU (7 FPS)
[87]	2016	✓	-	-	-	-	-	-	✓	-	-	i7 4700MQ (around 143 FPS per KF with [78])
BundleFusion [168]	2016	-	-	-	RGB-D	Det.+Des.	GPU-solver	-	-	✓	-	Titan X GPU (around 36 FPS)
Chisel [94]	2015	✓	-	✓	RGB-D	Tracking	Filtering	-	-	✓	-	Tango tablet 4GB RAM, quadcore CPU, Tegra K1 GPU Tango mobile phone, 2GB RAM, quadcore CPU
ElasticFusion [169]	2015	-	-	-	RGB-D	Direct	Graph [170]	Surfels	-	-	Semantic segmentation [122]	Core i7-4930K, GeForce GTX 780 Ti (32 FPS)
InfiniTAM [99]	2015	-	-	✓	RGB-D	Direct	ICP / [171]	-	-	✓	-	Tegra K1 (47 FPS), iPad Air 2 (21 FPS) DE5 PCIe board (44 FPS), DE1 FPGA SoC (2 FPS) [172]
ROVIO [68]	2015	✓	✓	✓	-	Direct	Filtering	Sparse	-	-	-	UP Board, ODROID XU4 (22 FPS) [159]
OKVIS [78]	2015	✓	✓	✓	-	Det.+Des.	Graph	Sparse	-	-	-	UP Board (11 FPS), ODROID XU4 (3 FPS) [159]
LSD-SLAM [6]	2014	✓	-	-	-	Direct	Graph	Dense	-	-	-	FPGA Zynq-7020 SoC (4.55 FPS) [173]
SLAM++ [174]	2013	-	-	-	RGB-D	Direct	Graph	-	-	✓	-	GPGPU implementation
KinectFusion [93]	2011	-	-	-	RGB-D	Direct	ICP	-	-	✓	-	Zynq UltraScale+ MPSoC ZCU102 (27.5 FPS) [175]
PTAM [59]	2007	✓	-	-	-	Detection	LBA / GBA	Sparse	-	-	-	ODROID XU4 (up to 12 FPS) [176]
MonoSLAM [34]	2007	✓	-	-	-	Detection	Filtering	Sparse	-	-	-	Intel Pentium M 1.60 GHz (53 FPS)

TABLE III
PERFORMANCES OF HARDWARE IMPLEMENTATION ON EMBEDDED PLATFORMS. (∗) REPRESENTS THE BACK-END RATE TO UPDATE THE STATES AND THE SPARSE 3D MAP [177], (∗∗) CORRESPONDS TO THE MEASURED TIME BETWEEN THE INPUT IMAGE TO THE UPDATED STATE, (†) INDICATES THAT THE METHOD FAILED ON ONE OR MORE SEQUENCES, WHICH HAVE NOT BEEN INCLUDED IN THE AVERAGE RESULT [159].

Methods	Year	HW implementation	Rate	Power	ATE RMSE (m)	Dataset
CNN-SLAM proc. [178]	2019	ASIC	80 FPS	243.6mW 61.8mW	97.90% in tr.; 99.34% in rot.	KITTI
Navion [177]	2018		71 FPS 19 FPS*	24mW	0.23	EuRoC
VINS-Mono [77]	2018	ODROID	7 FPS**	10W	0.16	EuRoC
MSCKF-based [161]	2017		20 FPS**		0.56	
SVO+MSF [58], [179]	2016		50 FPS**		0.69†	
SVO+GTSAM [58], [73]	2016		66 FPS**		0.11†	
ROVIO [68]	2015		22 FPS**		0.35	
OKVIS [78]	2013		3 FPS**		0.26†	
VINS-Mono [77]	2018	UP Board	7 FPS**	15W	0.15	EuRoC
MSCKF-based [161]	2017		20 FPS**		0.53	
SVO+MSF [58], [179]	2016		40 FPS**		0.69†	
SVO+GTSAM [58], [73]	2016		50 FPS**		0.12†	
ROVIO [68]	2015		-		-	
OKVIS [78]	2013		11 FPS**		0.27†	

methods in terms of accuracy of the estimated trajectory. For instance, the conventional monocular configuration [180] is outperformed by the supervised method [101] but not the stereo configuration, which provide a t_{rel} of 17.48%, 5.96% and 1.89%. In [104], the end-to-end unsupervised approach gives a t_{rel} of 5.58% compared to 3.21% and 1.89% for conventional approaches [162], [180] respectively on testing sequences of the KITTI dataset [133]. However, hybrid pipelines give suitable solutions to include specific DNN-based functions. In [156], the depth estimation from DNN [181] provides a processing time of 17.07ms and 7.09ms on the GPUs of Jetson Nano and TX2 respectively and 66.41ms and 105.07ms on the CPUs respectively with the Apache TVM optimization. On the same platforms, the block matching (BM) of the OpenCV library performs at 19.24ms, 12.38ms on GPUs and 27.76ms, 19.49ms on CPUs respectively. It demonstrates that the network prediction is faster on GPU with the TVM optimization, but the BM provides real-time performances on both implementations with lower processing times on CPUs. The accuracy of the dense conventional depth algorithms has not been quantified. However, the robustness of the deep learning-based estimation is demonstrated by a cleaner depth map than the noisy results of conventional methods [132].

III. LOCALIZATION AND 3D RECONSTRUCTION METHODS ON EMBEDDED SYSTEMS

Localization and 3D reconstruction functions require a lot of hardware resources. The main HW implementations highlighted in table II include high power and flexible CPUs/GPUs (i7 desktop/GTX), embedded CPUs/GPUs (Pi 3B+, ODROID, UP Board/TX2, Nano), and specific HW/SW co-design (FPGAs SoC). Transport systems, including autonomous cars to restricted MAVs, miniaturized robots, AR/VR systems, have a limited form factor and power budget. Those constraints affect the choice of implementation to meet the required accuracy and real-time performances. For instance, the power budget is around 10W-300W for autonomous cars [1], 10W-15W for MAVs [159], [182] and 10mW-10W for miniaturized robots and AR/VR devices [2], [177], [183], [184].

Solutions and trade-offs for computing localization and 3D reconstruction functions in resource-constrained systems motivates the purpose of this section.

A. Embedded platforms for localization and 3D reconstruction

A heterogeneous system is composed of various calculation resources. Components off-the-shelf (COTS), energy-efficient hardware accelerators or compact systems with low power consumption like vision chips are part of the available embedded platforms allowing the partitioning of advanced functions from sensors to 3D reconstruction.

1) Components Off-The-Shelf (COTS): In order to achieve real-time processing with limited resources, a trade-off between accuracy, robustness, execution time, memory management and power consumption must be reached [159]. Several VIO methods [58], [68], [77], [78], [161] have been implemented on two hardware platforms for MAVs. It includes the UP Board with a quad-core Intel Atom x5-Z8350 1.44GHz CPU, 4Go RAM, a power consumption around 12W, and the ODROID XU4 with a hybrid ARM, a quad-core ARM

TABLE IV
NUMBER OF PIXELS PROCESSED PER SECOND FOR FEATURE EXTRACTION FUNCTIONS BASED ON SW AND HW IMPLEMENTATIONS.

HW implementation	Features	Res.	FPS	MP/s
Intel Core2Duo [185]	Harris	WVGA	40	14.44
Intel i7-4790 [77]	Shi-Tomasi	WVGA	66	23.82
ARM Cortex-A15 [177]	KLT Shi-Tomasi	WVGA	19	6.86
Jetson TX2 [153]	KLT Shi-Tomasi	WVGA	100	36.10
FPGA Xilinx Zynq [185]	Harris	WVGA	333	120.20
FPGA Xilinx Zynq [186]	Harris+SURF	VGA	320	98.30
ASIC VIO [177]	KLT Shi-Tomasi	WVGA	71	25.63
ASIC CNN-VO [178]	CNN features	VGA	80	24.58

A7 1.5GHz and an ARM big.LITTLE configuration quad-core A15 at 2.0GHz. ODROID has 2 GB RAM and a power consumption of 10W.

Table III shows the real-time performances (in FPS) for each VIO method on COTS platforms. For instance, the graph-based VINS-Mono [77] provides 7 FPS while the MSCKF-based [161] is at 20 FPS on ODROID compared to 20 FPS and 40 FPS respectively on the reference laptop. Real-time performance is achieved by reducing the number of features per frame, the size of the optimization sliding window and by including advanced single instruction multiple data (SIMD) instructions, like Intel SSE and ARM NEON optimizations for UP Board and ODROID respectively. The accuracy is not impacted by the optimizations performed for VINS-Mono with an average of the absolute translation error (ATE RMSE) of 0.16m, 0.16m, 0.15m for the laptop, UP Board and ODROID respectively. Although, the MSCKF-based is affected with an average ATE RMSE of 0.41m, 0.53m, 0.56m respectively. ROVIO [68] is the only one that does not run on UP Board due to the CPU clock speed, which highlights the complexity of implementing VIO methods on different embedded platforms.

Partitioning a complete localization method with loop closures requires high computing resources. In [155], ORB-SLAM2 [82], which integrates a loop closure module has been optimized using NEON instructions to take advantage of the advanced SIMD used in the ARM processors of the Raspberry Pi 3B+ and Jetson Nano. It achieves an average tracking time of 6.11 FPS on Raspberry Pi 3B+ and 9.64 FPS on Jetson Nano with input images at a 752×480 resolution. This work focused on processing time and not on accuracy for the embedded implementations. The fact that the literature provides few information on the optimizations to be performed when using embedded platforms points to a line of research towards embedded SLAM, which incorporates loop closures.

A dense 3D representation is a challenge task to implement on embedded platforms due to the computation requirements. While the SVO method [58] implemented on ODROID U3 provides 3D poses, a W530 Lenovo laptop is used to compute a dense point cloud representation [54] on an NVIDIA Quadro K2000M GPU [187]. The 3D poses and input images are broadcasted at a frequency of 5Hz on a WiFi communication between the embedded platform and the laptop. Memory and speed efficient data structures tackle the limitations of a fixed-size volume and the large amount of memory required for volumetric reconstructions [93]. A moving TSDF volume [188], an octree-based approach [189], [190] and a hashing scheme [98] enable the reconstruction of large scale environments with compact data structures. The hashing scheme has been used in several researches to save memory and CPU budget [8], [94], [191] as it allows a complexity of $\mathcal{O}(1)$ compared to $\mathcal{O}(\log n)$ for octree structures [189].

In order to provide more available resource in the main embedded system, partitioning the localization pipeline with custom hardware implementations is useful to cope with the high computational complexity of advanced algorithms, like feature extraction.

In [173], specific units of the localization part of a semi-dense SLAM [6] has been accelerated on FPGA with a high-level synthesis (HLS) compiler. HLS compiler is used for performing low-level design optimizations based on conventional algorithms to increase the overall performance of the system. It achieves a framerate of 4.55 FPS compared to 2.27 FPS with a software-only implementation and a total power consumption of about 2.5W. In [185], the visual-inertial (VI) system consists of two imaging sensors with a resolution of 752×480 (Aptina MT9V034) synchronized with an IMU (ADIS16488) through an ARM-FPGA Xilinx Zynq 7020 processing. The latter has been used to speed up the detection of Harris [42] and FAST [44] features in order to allocate more CPU resources for other tasks [185]. Table IV shows the performances of feature extraction methods on SW laptops CPU [77], [185], Jetson TX2 [153] and HW FPGAs [185], [186], ASICs [177], [178]. Software solutions are easier to program with more flexibility, but less efficient than dedicated HW FPGAs and ASICs. The number of megapixels processed per second (MP/s), which is related to the image resolution (Res.) of 752×480 (WVGA) and 640×480 (VGA) and the processing time (FPS), increases significantly with the implementation on FPGAs as the HW design is entirely dedicated to the feature extraction function. The application-specific integrated circuit (ASIC) VIO and ASIC CNN-VO are not only focused on this function. They are designed to integrate a full VO/VIO pipeline, which explains the difference in performance compared to FPGAs.

2) Energy-efficient accelerators: Specialized hardware, such as ASICs, give more freedom to design specific localization and 3D reconstruction functions. They offer high real-time performance and low power consumption. They are also more expensive in terms of development and fabrication [192].

HoloLens2 [193] is one of the most advanced perception system based on the development of advanced functions to perceive an accurate topology of the environment with the generation of a mesh by leveraging multiple sensors with restrictive resources. It integrates a custom ASIC to perform the 6DoF localization and 3D mesh reconstruction functions, called the Holographic Processor Unit (HPU). The HPU processes input sensors data, including an IMU, a time-of-flight (ToF) depth sensor and four grayscale cameras at 30 FPS. Over all 160×480 four channel resolutions, an image of 640×480 is represented. The ASIC consumes less than 10W, can process more than 1 TOPS (Tera Operations Per Second) and contains 125 MB SRAM. It consists of 2 billion transistors in a 79mm^2 die size, 7 SIMD Fixed Point (SFP)

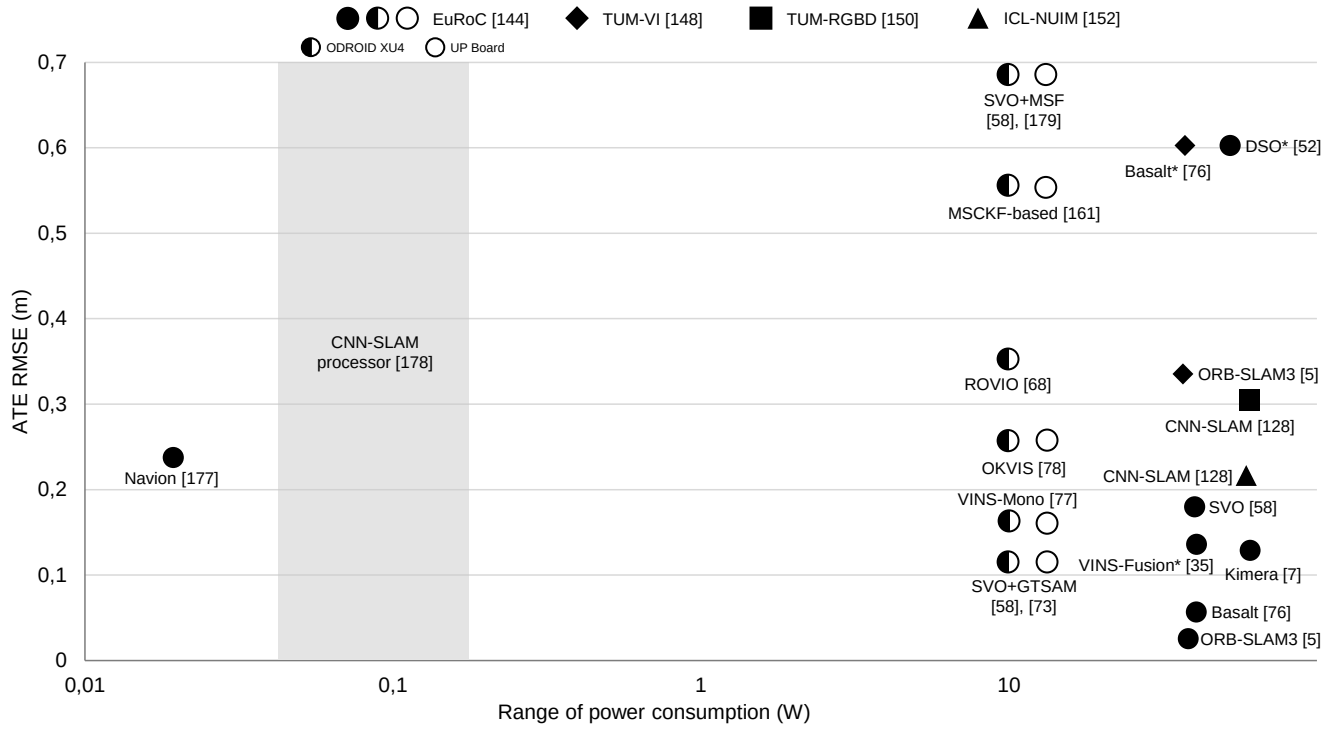

Fig. 2. Comparison of state-of-the-art methods in terms of accuracy and range of power consumption (W) between several hardware platforms, from ASIC to CPU-based. The vertical axis corresponds to the average RMSE in meters in all successful dataset sequences provided by the authors. Methods denoted by (*) means that the error has been obtained from [5].

for 2D processing, 6 Vector Floating Processor (VFP) for 3D processing and one dedicated core to DNNs processing programmable by Microsoft [184].

Navion is an energy-efficient VIO accelerator [177], which consists of three main parts: the Vision Front-End (VFE), the IMU Front-End (IFE) and the Back-End (BE) with local optimizations. Feature tracking (FT) is the only function in VFE to be performed per-frame. The remaining pipeline is based on the processing of keyframes. Based on input images at a 752×480 resolution, the feature detection and tracking achieves an average of 71 FPS, which is comparable to other hardware platforms illustrated in table IV. The performance is similar to software implementation on Intel i7. The ASIC provides efficient optimizations to reduce the memory usage. These include image compression in VFE, memory size reduction in BE and the way tracked features are stored in BE. The architecture optimizations reduce the initial memory size from 3.5MB to 854KB. To the best of our knowledge, the energy-efficient accelerator has not been publicly evaluated on a real MAV device, but on the Euroc dataset [144]. With an average 71 FPS tracking process at framerate, the chip produces 19 FPS at keyframe rate and consumes around 24mW. In terms of energy-efficiency, Navion performs 0.43-2.5 TOPS/W.

Navion is entirely based on geometric methods. In [178], the VO architecture is organized in three main parts. A CNN architecture to extract features from the input VGA image, a Perspective-n-Points (PnP) unit to compute the 2D-3D ($[R|t]$) 6DoF camera motion and a bundle adjustment (BA) unit to optimize on the last 20 keyframes. The CNN-based ASIC achieves an energy-efficiency of 3.6-5.34 TOPS/W, a latency of 12.5ms and consumes 243.6mW at 80 FPS VGA and reduces to 61.8 mW at 30 FPS VGA.

Partitioning the use loop closures for SLAM in a localization pipeline has been addressed by NeuroSLAM that integrates a SLAM architecture with a spiking neural network-based (SNN-based) [194]. While the VIO accelerators from the literature are designed with digital signals, NeuroSLAM additionally provides analogue signals to mimic SNNs. It achieves an energy-efficiency of 7.25-8.79 TOPS/W with a power consumption of 17.27-23.82mW respectively. Motivated by ultra-low power applications, the use of SNNs for SLAM opens new lines of research on this type of network.

Figure 2 illustrates ASIC implementations in the range of power consumption in mW. Several datasets are used to assess the accuracy of these implementations under real conditions. The accuracy of the method differs widely from one dataset to another. For instance, Navion [177] has an average error of 0.23m in the Euroc sequences, while Basalt [76] provides an average error of 0.051m with the same sequences and 0.6m on handheld sequences with the TUM-VI dataset. The figure also exhibits the gap between the number of real-time methods developed with high resources, embedded COTS hardware and specific energy-efficient accelerators.

3) Vision chips: Vision chips offer a specific implementation for integrating complex algorithms for a wide range of applications requiring low latency image processing [195], [196]. In addition to the imaging sensor, processing units are integrated for in-sensor computing, which increases the overall performance of the system. Vision chips have the capacity to compute feature extraction functions at very high

TABLE V
IMPLEMENTATION OF 3D SCENE RECONSTRUCTION METHODS ON EMBEDDED PLATFORMS.

Methods	3D reconstruction	HW implementation	Rate	Use case
FLaME [27]	Mesh	Intel Skull Canyon NUC i7 CPU flight computer	90 FPS	MAV
Voxblox [8]	Volumetric	Asctec Firefly Intel i7 2.1 GHz CPU	>4 FPS	
InfiniTAM [172]	Volumetric	Cyclone V Terasic DE1 FPGA SoC	2 FPS	Synthetic dataset [152]
KinectFusion [175]		SoC FPGA Zynq UltraScale+ MPSoC ZCU102	27.5 FPS	
InfiniTAM [172]		Stratix V Terasic DE5 PCIe board	44 FPS	

speed in a compact and ultra low power system. In [197], the vision system is designed to extract FAST features [198] and describe them with a 44-bit binary-edge descriptor. This system operates at 300 FPS. The remaining part of the VO pipeline runs on an Intel i7-6700HQ CPU with binary edges and corners images that are tolerant of motion blur compared to conventional visible cameras.

B. Algorithmic methods in embedded platforms

This subsection provides an overview of algorithmic approaches used for several embedded systems from COTS platforms to specific HW/SW co-design implementations. The localization part is dissociated from the 3D reconstruction to obtain a broader view of the current implementations.

1) Real-time localization: HoloLens [193] enables spatial mapping with mesh generation, spatial processing for finding planes and spatial understanding with semantic labels. The HPU integrates a localization pipeline [199] to provide an accurate pose estimation. Based on the use of inertial measurements and imaging sensors, a block filter consisting of an EKF and a sensor fusion algorithm is developed. On a closed loop path of 287m, the HoloLens localization system drifts 2.39m from start to finish [200] [201]. The overall quality of the 3D model provided by the device has some holes in the mesh. Its accuracy has been measured from a ground truth point cloud. The reconstruction of several offices provided by the AR/VR device gives an average Euclidean distance of 0.023m with a fixed scale between the 3D reconstruction and the ground truth.

The localization strategy implemented on the HPU gives a large view of the type of algorithms to be implemented that provide real-time pose estimations. The autonomous quadrotor system [187] highlights the localization parameters to limit the resource usage. The SVO visual odometry method [58] implemented in ODROID U3 uses two threads to estimate the camera motion and to insert keyframes into the extended map. The *fast* parameter of the approach has been used, which limits to 120 the number of detected features per frame and keeps in memory a maximum of 10 generated keyframes in the map. In order to obtain a robust system, the IMU data and the poses are merged via the MSF method [179], which uses an EKF. The experiments were carried out on a 20m long, 1.7m high indoor path and on a 100m and 20m outdoor path respectively. The system comprising SVO+MSF achieves a maximum drift of 0.5% of the travelled distance and an average trajectory error (ATE RMSE) of about 0.05m for a closed loop trajectory.

2) Real-time 3D reconstruction: Table V illustrates that only few methods from mesh to voxel-based have been used onboard MAVs or by taking advantage of HW/SW co-design on FPGA/SoC.

The FLaME mesh reconstruction method [27] has been implemented on an Intel Skull Canyon NUC flight computer. The MAV was equipped with a Point Grey Flea 3 camera operating at 60 FPS with an image resolution of 320×256 in addition to an IMU. The experiments were conducted in indoor and outdoor environments with a vehicle speed of $2.5m/s$ and $3.5m/s$ respectively. It reconstructed a very detailed mesh representation at a framerate over 90 FPS. The same use case has been used for the Voxblox volumetric reconstruction [8], [167]. The ROVIO odometry method [68] provides pose estimations to the reconstruction approach. Voxblox has been evaluated on a MAV platform equipped with an Intel i7 2.1 GHz CPU and a stereo camera synchronized to an IMU. With a voxel size representation of 0.2m, the computation time of the complete system is less than 250ms. In order to obtain an accurate scene representation with a lower granularity, the use of HW optimizations is particularly useful for maintaining real-time performance.

Table V shows several voxel-based reconstructions performed on HW FPGA/SoC. In [175], KinectFusion [93] has been optimally implemented on a SoC FPGA Xilinx UltraScale+ MPSoC ZCU102. Raycasting is the only part of the pipeline to be computed on the ARM CPU due to complex memory access. The reconstruction method achieves a performance of 27.5 FPS with 320×240 input images from the ICL-NUIM dataset. The optimizations focused on parameters that improve the execution time, so that the accuracy error of the method increases from around 0.018m to 0.08m from one sequence to the next. In [172], InfiniTAM [99] has been implemented on a low-cost Terasic DE1 FPGA SoC and on a high-cost Terasic DE5 PCIe board. The real-time performance widely differs depending on the available resources, with a performance of 2 FPS and 44 FPS respectively with input depth images at a 320×240 resolution.

Although FPGA SoC allows the acceleration of advanced functions on the hardware with the use of a CPU for other computations. The volumetric method [167] offers a CPU-based implementation, which provides research directions for implementing complex tasks for more accurate scene perception in a heterogeneous system.

IV. Conclusion

In this paper, we have reviewed visual(-inertial) SLAM methods, from real-time localization with scene cartography to volumetric reconstruction in the context of resource-constrained embedded platforms. It highlights the different strategies employed for localization and reconstruction functions, including the potential use of deep neural networks. The latter is particularly useful in a hybrid configuration to combine the strengths of deep learning and the maturity of model-based methods for specific functions, including feature detection, description, matching and intermediate representations. This study also provides an overview of the hardware implementation of localization and reconstruction functions from COTS systems to specific ASIC/SoC integration. It shows the gap between algorithmic methods developed with the high resources available in conventional laptops and those developed for transport systems with limited resources, including MAVs, miniaturized robots and mobile AR/VR devices.

Several odometry methods are developed with limited algorithmic complexity and provide parameters to be configured for real-time processing on restrictive platforms. This survey shows that few SLAM and volumetric methods are developed in this specific context. The implementation of loop closures capability for SLAM remains a challenge to integrate due to the required computational resources. As real-time processing, memory management and low power consumption are essential, several questions remain open to find the best trade-off between accuracy, robustness, scalability and resource consumption. The required precision of intermediate representations, including depth estimation and semantic segmentation for an accurate 3D model, is one of the research areas to be explored, as well as the computational cost and use of hardware resources, especially with deep learning methods. The granularity of the 3D reconstruction also raises several questions. For instance, what is the required granularity or space limitation in the reconstruction? What is the right trade-off for accurate real-time localization and reconstruction with limited available resources?

References

[1] L. Liu, S. Lu, R. Zhong, B. Wu, Y. Yao, Q. Zhang, and W. Shi, "Computing systems for autonomous driving: State of the art and challenges," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6469–6486, 2020.

[2] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.

[3] A. Couturier and M. A. Akhloufi, "A review on absolute visual localization for uav," *Robotics and Autonomous Systems*, vol. 135, p. 103666, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889020305066

[4] A. J. Davison, "Futuremapping: The computational structure of spatial ai systems," *ArXiv*, vol. abs/1803.11288, 2018.

[5] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, pp. 1–17, 2021.

[6] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[7] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1689–1696.

[8] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[9] P. Alliez, F. Bonardi, S. Bouchafa, J.-Y. Didier, H. Hadj-Abdelkader, F. I. I. Muñoz, V. Kachurka, B. Rault, M. Robin, and D. Roussel, "Real-time multi-slam system for agent localization and 3d mapping in dynamic scenarios," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4894–4900, 2020.

[10] P. Alliez, F. Bonardi, S. Bouchafa, J.-Y. Didier, H. Hadj-Abdelkader, F. I. Ireta Muñoz, V. Kachurka, B. Rault, M. Robin, and D. Roussel, "Indoor Localization and Mapping: Towards Tracking Resilience Through a Multi-SLAM Approach," in *MED 2020 - 28th Mediterranean Conference on Control and Automation*, Saint Raphael, France, Sep. 2020, pp. 465–470. [Online]. Available: https://hal.inria.fr/hal-02611679

[11] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems*, 2014.

[12] V. Kachurka, D. Roussel, H. Hadj-Abdelkader, F. Bonardi, J.-Y. Didier, and S. Bouchafa, "SWIR Camera-Based Localization and Mapping in Challenging Environments," in *20th International Conference on IMAGE ANALYSIS AND PROCESSING (ICIAP 2019)*, ser. Lecture Notes in Computer Science, vol. 11752, Trento, Italy, Sep. 2019, pp. 446–456. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02271971

[13] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios," *IEEE Robotics and Automation Letters*, vol. 3, pp. 994–1001, 2018.

[14] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[15] 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, 2016.

[16] D. M. Rosen, K. J. Doherty, A. Terán Espinoza, and J. J. Leonard, "Advances in inference and representation for simultaneous localization and mapping," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 215–242, 2021. [Online]. Available: https://doi.org/10.1146/annurev-control-072720-082553

[17] C. Stachniss, J. J. Leonard, and S. Thrun, *Simultaneous Localization and Mapping*. Cham: Springer International Publishing, 2016, pp. 1153–1176. [Online]. Available: https://doi.org/10.1007/978-3-319-32552-1_46

[18] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[19] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part ii," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[20] C. Chen, B. Wang, C. X. Lu, A. Trigoni, and A. Markham, "A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence," *ArXiv*, vol. abs/2006.12567, 2020.

[21] I. Salhi, M. Poreba, E. Piriou, V. Gouet-Brunet, and M. Ojail, "Chapter 8 - multimodal localization for embedded systems: A survey," in *Multimodal Scene Understanding*, M. Y. Yang, B. Rosenhahn, and V. Murino, Eds. Academic Press, 2019, pp. 199–278. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128173589000147

[22] 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*, vol. 2, pp. 194–220, 2017.

[23] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb, "State of the art on 3d reconstruction with rgb-d cameras," *Computer Graphics Forum*, vol. 37, 2018.

[24] Y. Alkendi, L. Seneviratne, and Y. Zweiri, "State of the art in vision-based localization techniques for autonomous navigation systems," *IEEE Access*, vol. 9, pp. 76 847–76 874, 2021.

[25] M. Bujanca, X. Shi, M. Spear, P. Zhao, B. Lennox, and M. Lujan, "Robust slam systems: Are we there yet?" in *IEEE/RSJ International*

Workshop on Intelligent Robots and Systems (IROS 2021). (Accepted/In press), 2021.

[26] R. Azzam, T. Taha, S. Huang, and Y. Zweiri, "Feature-based visual simultaneous localization and mapping: a survey," *SN Applied Sciences*, vol. 2, 02 2020.

[27] W. N. Greene and N. Roy, "Flame: Fast lightweight mesh estimation using variational smoothing on delaunay graphs," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4696–4704.

[28] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[29] C. Engels, H. Stewénius, and D. Nistér, "Bundle adjustment rules," in *In Photogrammetric Computer Vision*, 2006.

[30] S. Garg, N. Sünderhauf, F. Dayoub, D. Morrison, A. Cosgun, G. Carneiro, Q. Wu, T. J. Chin, I. Reid, S. Gould, P. Corke, and M. Milford, *Semantics for Robotic Mapping, Perception and Interaction: A Survey*. Now Foundations and Trends, 2020.

[31] I. Kostavelis and A. Gasteratos, "Semantic mapping for mobile robotics tasks: A survey," *Robotics and Autonomous Systems*, vol. 66, pp. 86–103, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889014003030

[32] B. Zhou, P. Krähenbühl, and V. Koltun, "Does computer vision matter for action?" *Science Robotics*, vol. 4, no. 30, p. eaaw6661, 2019. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.aaw6661

[33] Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 1403–1410 vol.2.

[34] A. Davison, I. 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.

[35] T. Qin, S. Cao, J. Pan, and S. Shen, "A general optimization-based framework for global pose estimation with multiple sensors," *ArXiv*, vol. abs/1901.03642, 2019.

[36] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 3565–3572.

[37] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[38] B. K. P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Opt. Soc. Am. A*, vol. 4, no. 4, pp. 629–642, Apr 1987. [Online]. Available: http://www.osapublishing.org/josaa/abstract.cfm?URI=josaa-4-4-629

[39] M. C. Laurent Kneip and R. Siegwart, "Robust real-time visual odometry with a single camera and an imu," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 2011, pp. 16.1–16.11, http://dx.doi.org/10.5244/C.25.16.

[40] E. H. Adelson, P. J. Burt, C. H. Anderson, J. M. Ogden, and J. R. Bergen, "Pyramid methods in image processing," 1984.

[41] H. P. Moravec, "Towards automatic visual obstacle avoidance," in *IJCAI*, 1977.

[42] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey Vision Conference*, 1988.

[43] Jianbo Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.

[44] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 430–443.

[45] G. LoweDavid, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[46] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006*, A. Leonardis, H. Bischof, and A. Pinz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 404–417.

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

[48] J. yves Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[49] J. Civera, O. G. Grasa, A. J. Davison, and J. M. M. Montiel, "1-point ransac for ekf-based structure from motion," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 3498–3504.

[50] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1, 2004, pp. I–I.

[51] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*, 2011, pp. 2320–2327.

[52] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[53] D. Cremers, "Direct methods for 3d reconstruction and visual slam," in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, 2017, pp. 34–38.

[54] M. Pizzoli, C. Forster, and D. Scaramuzza, "Remode: Probabilistic, monocular dense reconstruction in real time," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2609–2616.

[55] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1449–1456.

[56] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1935–1942.

[57] X. Dong, L. Cheng, H. Peng, and T. Li, "Fsd-slam: a fast semi-direct slam algorithm," *Complex & Intelligent Systems*, 2021.

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

[59] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[60] J. Leonard, H. Durrant-Whyte, and I. Cox, "Dynamic map building for autonomous mobile robot," in *EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications*, 1990, pp. 89–96 vol.1.

[61] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous robot vehicles*. Springer, 1990, pp. 167–193.

[62] J. Leonard and H. Durrant-Whyte, "Simultaneous map building and localization for an autonomous mobile robot," in *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, 1991, pp. 1442–1447 vol.3.

[63] J. Folkesson and H. Christensen, "Graphical slam - a self-correcting map," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, 2004, pp. 383–390 Vol.1.

[64] U. Frese and L. Schroder, "Closing a million-landmarks loop," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 5032–5039.

[65] F. Dellaert and M. Kaess, "Square root sam: Simultaneous localization and mapping via square root information smoothing," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006. [Online]. Available: https://doi.org/10.1177/0278364906072768

[66] J. Aulinas, Y. Petillot, J. Salvi, and X. Lladó, "The slam problem: A survey," in *Proceedings of the 2008 Conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*. NLD: IOS Press, 2008, p. 363–371.

[67] J. Gui, D. Gu, S. Wang, and H. Hu, "A review of visual inertial odometry from filtering and optimisation perspectives," *Advanced Robotics*, vol. 29, pp. 1289 – 1301, 2015.

[68] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 298–304.

[69] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[70] 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)*, 2020, pp. 4666–4672.

[71] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Found. Trends Robotics*, vol. 6, pp. 1–139, 2017.

[72] F. Dellaert and Others, "Georgia tech smoothing and mapping (gtsam)," 2019. [Online]. Available: https://gtsam.org/

[73] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012. [Online]. Available: https://doi.org/10.1177/0278364911430419

[74] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

[75] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[76] V. Usenko, N. Demmel, D. Schubert, J. Stückler, and D. Cremers, "Visual-inertial mapping with non-linear factor recovery," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 422–429, 2020.

[77] 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.

[78] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015. [Online]. Available: https://doi.org/10.1177/0278364914554813

[79] D. Galvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[80] H. Wang, J. Li, M. Ran, and L. Xie, "Fast loop closure detection via binary content," in *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, 2019, pp. 1563–1568.

[81] F. Schenk and F. Fraundorfer, "Reslam: A real-time robust edge-based slam system," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 154–160.

[82] 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.

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

[84] T. Schöps, T. Sattler, and M. Pollefeys, "Surfelmeshing: Online surfel-based mesh reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 2494–2507, 2020.

[85] S. Dinas and J. Bañón, "A review on delaunay triangulation with application on computer vision," *IJCSE - International Journal of Computer Science and Engineering*, vol. 3, pp. 9–18, 03 2014.

[86] A. Rosinol, T. Sattler, M. Pollefeys, and L. Carlone, "Incremental visual-inertial 3d mesh generation with structural regularities," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8220–8226, 2019.

[87] L. Teixeira and M. Chli, "Real-time mesh-based scene estimation for aerial inspection," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4863–4869.

[88] M. Yokozuka, S. Oishi, S. Thompson, and A. Banno, "Vitamin-e: Visual tracking and mapping with extremely dense feature points," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9633–9642, 2019.

[89] E. Piazza, A. Romanoni, and M. Matteucci, "Real-time cpu-based large-scale three-dimensional mesh reconstruction," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1584–1591, 2018.

[90] J. R. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Computational Geometry*, vol. 22, no. 1, pp. 21 – 74, 2002, 16th ACM Symposium on Computational Geometry. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925772101000475

[91] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH 1987*. ACM Press, 1987.

[92] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 303–312. [Online]. Available: https://doi.org/10.1145/237170.237269

[93] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[94] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao, "Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields," in *Robotics: Science and Systems*, 2015.

[95] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan, "Interactive ray tracing for isosurface rendering," in *Proceedings Visualization '98 (Cat. No.98CB36276)*, 1998, pp. 233–238.

[96] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[97] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon, "Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera," in *UIST '11 Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, October 2011, pp. 559–568.

[98] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, Nov. 2013. [Online]. Available: https://doi.org/10.1145/2508363.2508374

[99] O. Kähler, V. Prisacariu, C. Ren, X. Sun, P. Torr, and D. Murray, "Very high frame rate volumetric integration of depth images on mobile devices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, pp. 1241–1250, 2015.

[100] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

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

[102] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2758–2766.

[103] R. Clark, S. Wang, H. Wen, A. Markham, and A. Trigoni, "Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem," in *AAAI*, 2017.

[104] R. Li, S. Wang, and D. Gu, "Deepslam: A robust monocular slam system with unsupervised deep learning," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 4, pp. 3577–3587, 2021.

[105] C. Tang and P. Tan, "BA-net: Dense bundle adjustment networks," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=B1gabhRcYX

[106] J. Czarnowski, T. Laidlow, R. Clark, and A. J. Davison, "Deepfactors: Real-time probabilistic dense monocular slam," *IEEE Robotics and Automation Letters*, vol. 5, pp. 721–728, 2020.

[107] S. Arshad and G.-W. Kim, "Role of deep learning in loop closure detection for visual and lidar slam: A survey," *Sensors*, vol. 21, no. 4, p. 1243, Feb 2021. [Online]. Available: http://dx.doi.org/10.3390/s21041243

[108] X. Zhang, Y. Su, and X. Zhu, "Loop closure detection for visual slam systems using convolutional neural network," in *2017 23rd International Conference on Automation and Computing (ICAC)*, 2017, pp. 1–6.

[109] D. Li, X. Shi, Q. Long, S. Liu, W. Yang, F. Wang, Q. Wei, and F. Qiao, "Dxslam: A robust and efficient visual slam system with deep features," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4958–4965.

[110] M. Sons, C. Kinzig, D. Zanker, and C. Stiller, "An approach for cnn-based feature matching towards real-time slam," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 1305–1310.

[111] Z. Xu, J. Yu, C. Yu, H. Shen, Y. Wang, and H. Yang, "Cnn-based feature-point extraction for real-time visual slam on embedded fpga," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2020, pp. 33–37.

[112] Q. Zhou, T. Sattler, and L. Leal-Taixe, "Patch2pix: Epipolar-guided pixel-level correspondences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4669–4678.

[113] J. Tang, L. Ericson, J. Folkesson, and P. Jensfelt, "Gcnv2: Efficient correspondence prediction for real-time slam," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3505–3512, 2019.

[114] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[115] L. Xia, J. Cui, R. Shen, X. Xu, Y. Gao, and X. Li, "A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots," *International Journal of Advanced Robotic Systems*, vol. 17, no. 3, p. 1729881420919185, 2020. [Online]. Available: https://doi.org/10.1177/1729881420919185

[116] K.-N. Lianos, J. L. Schönberger, M. Pollefeys, and T. Sattler, "Vso: Visual semantic odometry," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 246–263.

[117] A. Gawel, C. D. Don, R. Siegwart, J. Nieto, and C. Cadena, "X-view: Graph-based semantic multi-view localization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1687–1694, 2018.

[118] C. Zhang, L. Chen, and S. Yuan, "St-vio: Visual-inertial odometry combined with image segmentation and tracking," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 10, pp. 8562–8570, 2020.

[119] S. Wen, P. Li, Y. Zhao, H. Zhang, F. Sun, and Z. Wang, "Semantic visual slam in dynamic environment," in *Autonomous Robots*, 2021.

[120] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1168–1174.

[121] K. Tateno, F. Tombari, and N. Navab, "Real-time and scalable incremental segmentation on dense slam," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 4465–4472.

[122] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4628–4635.

[123] J. Wald, K. Tateno, J. Sturm, N. Navab, and F. Tombari, "Real-time fully incremental scene understanding on mobile platforms," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3402–3409, 2018.

[124] R. A. Rosu, J. Quenzel, and S. Behnke, "Semi-supervised semantic mapping through label propagation with semantic texture meshes," *International Journal of Computer Vision*, vol. 128, no. 5, pp. 1220–1238, jun 2019.

[125] M. Grinvald, F. Furrer, T. Novkovic, J. J. Chung, C. Cadena, R. Siegwart, and J. Nieto, "Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 3037–3044, July 2019.

[126] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, "Panopticfusion: Online volumetric semantic mapping at the level of stuff and things," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4205–4212, 2019.

[127] Z. Landgraf, F. Falck, M. Bloesch, S. Leutenegger, and A. J. Davison, "Comparing view-based and map-based semantic labelling in real-time slam," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 6884–6890.

[128] K. Tateno, F. Tombari, I. Laina, and N. Navab, "Cnn-slam: Real-time dense monocular slam with learned depth prediction," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6565–6574.

[129] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam - learning a compact, optimisable representation for dense visual slam," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2560–2568.

[130] X. Yin, X. Wang, X. Du, and Q. Chen, "Scale recovery for monocular visual odometry using depth estimated with deep convolutional neural fields," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5871–5879.

[131] Y. Li, C. Xie, H. Lu, X. Chen, J. Xiao, and H. Zhang, "Scale-aware monocular slam based on convolutional neural network," in *2018 IEEE International Conference on Information and Automation (ICIA)*, 2018, pp. 51–56.

[132] D. Martins, K. Van Hecke, and G. De Croon, "Fusion of stereo and still monocular depth estimates in a self-supervised learning context," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 849–856.

[133] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[134] J. Yu, F. Gao, J. Cao, C. Yu, Z. Zhang, Z. Huang, Y. Wang, and H. Yang, "Cnn-based monocular decentralized slam on embedded fpga," in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2020, pp. 66–73.

[135] J. Yu, G. Ge, Y. Hu, X. Ning, J. Qiu, K. Guo, Y. Wang, and H. Yang, "Instruction driven cross-layer cnn accelerator for fast detection on fpga," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 11, no. 3, Dec. 2018. [Online]. Available: https://doi.org/10.1145/3283452

[136] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. Reid, "Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[137] W. Wen, Y. Zhou, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, and L. Hsu, "Urbanloco: A full sensor suite dataset for mapping and localization in urban scenes," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2310–2316, 2020.

[138] L.-T. Hsu, N. Kubo, W. Chen, Z. Liu, T. Suzuki, and J. Meguro, "Urbannav: An open-sourced multisensory dataset for benchmarking positioning algorithms designed for urban areas," in *In Proceedings of the ION GNSS+ 2021*, 2021.

[139] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *International Journal of Robotics Research*, vol. 38, no. 6, pp. 642–657, 2019.

[140] 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," in *2020 International Conference on Robotics and Automation (ICRA)*, 2020, pp. 3139–3145.

[141] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016. [Online]. Available: https://doi.org/10.1177/0278364915614638

[142] A. Pronobis and B. Caputo, "COLD: COsy Localization Database," *The International Journal of Robotics Research (IJRR)*, vol. 28, no. 5, pp. 588–594, May 2009. [Online]. Available: http://www.pronobis.pro/publications/pronobis2009ijrr

[143] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

[144] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

[145] A. L. Majdik, C. Till, and D. Scaramuzza, "The zurich urban micro aerial vehicle dataset," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 269–273, 2017. [Online]. Available: https://doi.org/10.1177/0278364917702237

[146] E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss, "ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals," 2019. [Online]. Available: https://www.ipb.uni-bonn.de/pdfs/palazzolo2019iros.pdf

[147] S. Park, T. Schöps, and M. Pollefeys, "Illumination change robustness in direct visual slam," in *ICRA*, 2017.

[148] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The tum vi benchmark for evaluating visual-inertial odometry," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1680–1687.

[149] J. Engel, V. C. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," *ArXiv*, vol. abs/1607.02555, 2016.

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

[151] S. Wang, J. Yue, Y. Dong, S. He, H. Wang, and S. Ning, "A synthetic dataset for visual slam evaluation," *Robotics and Autonomous Systems*, vol. 124, p. 103336, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889019301009

[152] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam," in *2014 IEEE*

International Conference on Robotics and Automation (ICRA), 2014, pp. 1524–1531.

[153] A. Rosinol, A. Violette, M. Abate, N. Hughes, Y. Chang, J. Shi, A. Gupta, and L. Carlone, "Kimera: from SLAM to spatial perception with 3D dynamic scene graphs," in *arxiv*, 2021.

[154] S. Aldegheri, N. Bombieri, D. D. Bloisi, and A. Farinelli, "Data flow orb-slam for real-time performance on embedded gpu boards," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5370–5375.

[155] O. C. B. Silveira, J. G. O. C. de Melo, L. A. S. Moreira, J. B. N. G. Pinto, L. R. L. Rodrigues, and P. F. F. Rosa, "Evaluating a visual simultaneous localization and mapping solution on embedded platforms," in *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*, 2020, pp. 530–535.

[156] N. Merrill, P. Geneva, and G. Huang, "Robust monocular visual-inertial depth completion for embedded systems," in *Proc. of the IEEE International Conference on Robotics and Automation*, Xi'an, China, 2021.

[157] Y. Verdie, F. Lafarge, and P. Alliez, "Lod generation for urban scenes," *ACM Transactions on Graphics (TOG)*, vol. 34, pp. 1 – 14, 2015.

[158] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1328–1338.

[159] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2502–2509.

[160] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[161] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017. [Online]. Available: https://github.com/daniilidis-group/msckf_mono

[162] 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.

[163] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[164] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. Yuille, "Towards unified depth and semantic prediction from a single image," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2800–2809.

[165] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 239–248, 2016.

[166] T. J. Steiner, R. D. Truax, and K. Frey, "A vision-aided inertial navigation system for agile high-speed flight in unmapped environments: Distribution statement a: Approved for public release, distribution unlimited," in *2017 IEEE Aerospace Conference*, 2017, pp. 1–10.

[167] H. Oleynikova, C. Lanegger, Z. Taylor, M. Pantic, A. Millane, R. Siegwart, and J. Nieto, "An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments," *Journal of Field Robotics*, vol. 37, no. 4, pp. 642–666, 2020.

[168] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration," *ACM Trans. Graph.*, vol. 36, no. 3, May 2017. [Online]. Available: https://doi.org/10.1145/3054739

[169] T. Whelan, S. Leutenegger, R. F. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph," in *Robotics: Science and Systems*, 2015.

[170] R. W. Sumner, J. Schmid, and M. Pauly, "Embedded deformation for shape manipulation," *ACM Trans. Graph.*, vol. 26, no. 3, p. 80–es, 2007. [Online]. Available: https://doi.org/10.1145/1276377.1276478

[171] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan, "Estimation of imu and marg orientation using a gradient descent algorithm," in *2011 IEEE International Conference on Rehabilitation Robotics*, 2011, pp. 1–7.

[172] Q. Gautier, A. Althoff, and R. Kastner, "Fpga architectures for real-time dense slam," in *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, vol. 2160-052X, 2019, pp. 83–90.

[173] K. Boikos and C.-S. Bouganis, "Semi-dense slam on an fpga soc," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1–4.

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

[175] M. R. Gkeka, A. Patras, C. D. Antonopoulos, S. Lalis, and N. Bellas, "Fpga architectures for approximate dense slam computing," in *Design, Automation and Test in Europe Conference and Exhibition (DATE)*, Grenoble, France, Feb. 2021.

[176] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, "S-ptam: Stereo parallel tracking and mapping," *Robotics and Autonomous Systems*, vol. 93, pp. 27–42, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015302955

[177] A. Suleiman, Z. Zhang, L. Carlone, S. Karaman, and V. Sze, "Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1106–1119, 2019.

[178] Z. Li, Y. Chen, L. Gong, L. Liu, D. Sylvester, D. Blaauw, and H. Kim, "An 879gops 243mw 80fps vga fully visual cnn-slam processor for wide-range autonomous exploration," *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*, pp. 134–136, 2019.

[179] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3923–3929.

[180] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3d reconstruction in real-time," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 963–968.

[181] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6101–6108.

[182] "Skydio 2." [Online]. Available: https://www.skydio.com/skydio-2

[183] D. Palossi, A. Loquercio, F. Conti, E. Flamand, D. Scaramuzza, and L. Benini, "A 64-mw dnn-based visual navigation engine for autonomous nano-drones," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8357–8371, 2019.

[184] E. Terry, "Silicon at the heart of hololens 2," in *2019 IEEE Hot Chips 31 Symposium (HCS)*, 2019, pp. 1–26.

[185] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 431–437.

[186] M. Lepecq and M. Darouich, "A stream hardware architecture for keypoint matching based on a speculative approach," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[187] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle," *J. Field Robot.*, vol. 33, no. 4, p. 431–450, Jun. 2016. [Online]. Available: https://doi.org/10.1002/rob.21581

[188] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard, "Kintinuous: Spatially extended kinectfusion," July 2012.

[189] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at https://octomap.github.io. [Online]. Available: https://octomap.github.io

[190] F. Steinbrücker, J. Sturm, and D. Cremers, "Volumetric 3d mapping in real-time on a cpu," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 2021–2028.

[191] M. Muglikar, Z. Zhang, and D. Scaramuzza, "Voxel map for visual slam," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4181–4187, 2020.

[192] Z. Zhang, A. Suleiman, L. Carlone, V. Sze, and S. Karaman, "Visual-inertial odometry on chip: An algorithm-and-hardware co-design approach," in *Robotics: Science and Systems*, 2017.

[193] Microsoft, "Microsoft hololens v1 and hololens v2," 2016, 2019. [Online]. Available: https://www.microsoft.com/en-us/hololens

[194] J. H. Yoon and A. Raychowdhury, "Neuroslam: A 65-nm 7.25-to-8.79-tops/w mixed-signal oscillator-based slam accelerator for edge

robotics," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 66–78, 2021.

[195] L. Millet, S. Chevobbe, C. Andriamisaina, L. Benaissa, E. Deschaseaux, E. Beigne, K. Ben Chehida, M. Lepecq, M. Darouich, F. Guellec, T. Dombek, and M. Duranton, "A 5500-frames/s 85-gops/w 3-d stacked bsi vision chip based on parallel in-focal-plane acquisition and processing," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1096–1105, 2019.

[196] P. Dudek and P. J. Hicks, "A general-purpose processor-per-pixel analog simd vision chip," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 1, pp. 13–20, 2005.

[197] R. Murai, S. Saeedi, and P. Kelly, "Bit-vo: Visual odometry at 300 fps using binary features from the focal plane," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8579–8586, 2020.

[198] J. Chen, S. Carey, and P. Dudek, "Feature extraction using a portable vision system," in *Vision-based Agile Autonomous Navigation of UAVs-Workshop, IEEE/RSJ International Conference on IntelligentRobots and Systems (IROS)*, 2017.

[199] M. J. Ebstyne, F. Schafalitzky, D. Steedly, C. Chan, E. Eade, A. Kipman, and G. Klein, "Pose tracking an augmented reality device," U.S. Patent 9 495 801, 2016.

[200] P. Hübner, K. Clintworth, Q. Liu, M. Weinmann, and S. Wursthorn, "Evaluation of hololens tracking and depth sensing for indoor mapping applications," *Sensors*, vol. 20, no. 4, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/4/1021

[201] K. Khoshelham, H. Tran, and D. Acharya, "Indoor mapping eyewear: Geometric evaluation of spatial mapping capability of hololens," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLII-2/W13, pp. 805–810, 2019. [Online]. Available: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-2-W13/805/2019/

Quentin Picard received the M.S. degree in mobile autonomous systems from the University of Paris-Saclay, France in 2019. He is currently pursuing the Ph.D. degree at the CEA LIST Institute, Saclay, France in collaboration with the IBISC (Computer Science, Bio-Informatics and Complex Systems) laboratory of the University of Paris-Saclay, France. His research interests involve the generation of a semantic and dynamic 3D scene reconstruction in real-time for embedded systems.

Stephane Chevobbe received the Ph.D. degree in microelectronic and signal processing from the University of Rennes 1, Rennes, France, in 2005. From 2006 to 2009, he participated in several national and European Research Projects that lead to the realizations of the application specified integrated circuit and reconfigurable architectures for embedded systems. Since 2009, he has participated in the design of computing architectures for embedded vision applications. He is currently an Expert and a Research Engineer with the CEA LIST Institute, Saclay, France, where he is involved in the domain of embedded computing architecture. His research interests include reconfigurable, programmable and dedicated embedded architectures, and embedded architectures for image processing.

Mehdi Darouich received the Ph.D. degree in embedded systems from the University of Rennes 1, Rennes, France, in 2010. He joined the IC Design and Embedded Software Division, CEA LIST (French Atomic Energy Commision), Saclay, France, where he works in the field of embedded processing architectures and real-time machine vision applications for embedded purposes. His current research interests include smart sensors architecture design, stereo vision perception, localization, and navigation.

Jean-Yves Didier is an Associate Professor at Université d'Evry-val d'Essonne (Université Paris-Saclay) since September 2006. He received, in 2002, two M.S. degrees, both in computer science, from ENSIIE of Evry (public school of engineers) and Université d'Evry. He defended his Phd degree in robotics in 2005 at the same university. His research interests are focused on software architectures for mixed reality applications and their requirements such as localization and 3D environment reconstruction. He is also currently co-head of the Interaction, Virtual and Augmented Reality, Ambiant Robotics research team of the IBISC (Computer Science, Bio-Informatics and Complex Systems) laboratory of the University of Paris-Saclay, France.